КЛАССИЧЕСКИЕ
И
КВАНТОВЫЕ
ВЫЧИСЛЕНИЯ

А. Китаев, А. Шень, М. Вялый

МЦНМО
ЧеРо
Москва 1999

ББК 22.12
22.134
К53

А. Китаев, А. Шень, М. Вялый

К53 Классические и квантовые вычисления. — М.: МЦНМО, ЧеРо, 1999. — 192 с.

ISBN 5-900916-35-9

Эта книга предназначена для первоначального знакомства с новой быстро развивающейся и популярной областью исследований — теорией квантовых вычислений. Вначале приводится краткое введение в классическую теорию сложности вычислений. Затем подробно излагаются основы теории квантовых вычислений, включая описание основных известных к настоящему времени эффективных квантовых алгоритмов.

Для студентов физико-математических специальностей (начиная со второго года обучения), аспирантов, научных работников: математиков и физиков.

ББК 22.12
22.134

РФФИ *Издание осуществлено при финансовой поддержке РФФИ (издательский проект №99-01-14054)*

ISBN 5-900916-35-9

Редактор В. В. Ященко
Технический редактор М. Н. Вялый

Лицензия ЛР №071150 от 11.04.95 г. Подписано в печать 16.09.99 г.
Формат 60 × 90/16. Бумага офсетная №1. Печать офсетная.
Печ. л. 12,0. Тираж 1500.

Издательство Московского Центра непрерывного математического образования. 121002, Москва, Большой Власьевский пер., д. 11.

Издательство «ЧеРо». *Редакционно-издательский отдел*: 121002, Москва, Б. Власьевский пер., д. 11, комн. 208. Тел. 241 33 90, 241 18 69.
Отдел реализации. 118899, Москва, ул. Акад. Хохлова, д. 11.
Тел. 939 47 09, 939 34 93.

Оглавление

Предисловие

В последние годы интерес к тому, что называется «квантовые компьютеры», необычайно возрос. Идея использования возможностей квантовой механики при организации вычислений выглядит всё более привлекательной, начаты экспериментальные работы в этой области.

Однако перспективы физической реализации квантовых компьютеров пока совершенно неясны. Скорее всего, это дело нескольких десятилетий. Основные достижения в этой области носят пока чисто математический характер.

Эта книга предназначена для первоначального знакомства с математической теорией квантовых вычислений. Для удобства читателя вначале даётся краткое введение в классическую теорию сложности вычислений. Затем подробно излагаются основы теории квантовых вычислений, включая описание основных известных к настоящему времени эффективных квантовых алгоритмов.

Основу книги составили материалы курса «Классическое и квантовое вычисление», прочитанного А. Шенем (классические вычисления) и А. Китаевым (квантовые вычисления) в Высшем колледже математики Независимого Московского университета в весеннем семестре 1998 г. При подготовке книги также использовались материалы курса *Physics 229 – Advanced Mathematical Methods of Physics* (*Quantum computation*), который вели Дж. Прескилл (John Preskill) и А. Китаев (при участии А. Ландала (Andrew Landahl)) в Калифорнийском технологическом институте в 1998–1999 уч. г.

В основном эта книга рассчитана на старшекурсников и аспирантов. Впрочем, требуемые для чтения этой книги знания невелики, так что она доступна и младшекурсникам; с другой стороны, некоторые сюжеты могут представлять интерес и для профессионалов.

Мы старались учесть то обстоятельство, что этой книгой могут заинтересоваться люди с совершенно разной подготовкой: чистые математики, физики, специалисты в computer science. Поэтому мы не предполагали, что наш читатель имеет хорошую математическую подготовку. Важно понимать теоретико-множественный язык, основы

линейной алгебры и теории вероятностей; иметь минимальные представления о понятии алгоритма (навыки практического программирования нетривиальных алгоритмов вполне достаточны). В последних параграфах появляются и более сильные средства, например, группы гомологий, хотя никаких знаний из гомологической алгебры при этом не потребуется.

Обозначения

$\vee$	дизъюнкция (логическое ИЛИ)
$\wedge$	конъюнкция (логическое И)
$\neg$	отрицание
$\oplus$	сложение по модулю 2 (а также прямая сумма линейных пространств)
$\Longrightarrow$	импликация (логическое следование)
$\Longleftrightarrow$	логическая эквивалентность
$\mathcal{A}^*$	множество конечных слов в алфавите $\mathcal{A}$
$\sqcup$	пустой символ (пробел) в алфавите машины Тьюринга
$\delta(\cdot,\cdot,\cdot)$	функция переходов машины Тьюринга
$L_1 \propto L_2$	сводимость предикатов по Карпу (L_1 сводится к L_2) (с. 32)
$f(n) = O(g(n))$	существует такое число C, что $f(n) \leqslant Cg(n)$
$f(n) = \Omega(g(n))$	существует такое число C, что $f(n) \geqslant Cg(n)$
$f(n) = \mathrm{poly}(n)$	то же самое, что $f(n) = O(n^{O(1)})$
$\mathbb{F}_q$	конечное поле из q элементов
$\mathbb{Z}/n\mathbb{Z}$	кольцо вычетов по модулю n
$\mathbb{Z}_n$	аддитивная группа кольца $\mathbb{Z}/n\mathbb{Z}$
$(\mathbb{Z}/n\mathbb{Z})^*$	мультипликативная группа обратимых элементов $\mathbb{Z}/n\mathbb{Z}$
E^*	$(= \mathrm{Hom}(E, \mathbf{U}(1)))$ — группа характеров абелевой группы E
$\mathrm{Sp}_2(n)$	симплектическая группа над полем $\mathbb{F}_2$ размерности n (с. 133)
$\mathrm{ESp}_2(n)$	расширенная симплектическая группа над полем $\mathbb{F}_2$ размерности n (с. 132)
$\mathbb{C}$	множество комплексных чисел
z^*	комплексное сопряжение
$\mathbf{U}(\mathcal{M})$	группа унитарных операторов на пространстве $\mathcal{M}$
$\mathbf{SU}(\mathcal{M})$	специальная унитарная группа на пространстве $\mathcal{M}$
$\mathbf{SO}(\mathcal{M})$	специальная ортогональная группа на евклидовом пространстве $\mathcal{M}$
$\mathbb{C}(a, b, \dots)$	пространство, порождённое векторами $a, b, \dots$
$\mathcal{M}^*$	пространство линейных функционалов на пространстве $\mathcal{M}$
$\mathcal{M}^{\otimes n}$	n-я тензорная степень
$\mathbf{L}(\mathcal{M})$	пространство линейных операторов на $\mathcal{M}$
$\mathbf{L}(\mathcal{N}, \mathcal{M})$	пространство линейных отображений из $\mathcal{N}$ в $\mathcal{M}$
$\mathbb{B}$	классический бит (множество $\{0,1\}$)
$\mathcal{B}$	квантовый бит (q-бит, пространство $\mathbb{C}^2$ — с. 50)
$\langle\xi\vert$	бра-вектор (с. 53)
$\vert\xi\rangle$	кет-вектор (с. 52)
$\langle\xi\vert\eta\rangle$	скалярное произведение
$A^\dagger$	эрмитово сопряжённый оператор
$\bigoplus$	обратимое копирование бита (Controlled NOT) (с. 57)
$f_\oplus$	обратимая функция, соответствующая булевой функции f (с. 56)
$I_\mathcal{L}$	тождественный оператор на пространстве $\mathcal{L}$
$\hat{G}$	унитарный оператор, соответствующий перестановке G (с. 56)
$\Lambda(U)$	оператор U с квантовым управлением (с. 60)
$\Pi_\mathcal{M}$	проектор (оператор проектирования) на подпространство $\mathcal{M}$
$\sigma(\alpha_1, \beta_1, \dots, \alpha_n, \beta_n)$	базисные операторы на пространстве $\mathcal{B}^{\otimes n}$ (с. 130)
$A \cdot B$	преобразование матриц плотности $\rho \mapsto A\rho B$ (с. 125)
$U[A]$	оператор, действующий на квантовый регистр (множество q-битов) A (с. 54)
$\mathrm{Tr}_\mathcal{F} A$	частичный след от оператора A по пространству $\mathcal{F}$ (с. 79)
$\Vert\cdot\Vert$	норма вектора (с. 64) или операторная норма оператора (с. 64)
$\Vert\cdot\Vert_{\mathrm{tr}}$	следовая норма (с. 123)
$\Vert\cdot\Vert_\diamond$	норма для преобразований матриц плотности (с. 125)
$\vert\cdot\vert$	мощность множества или модуль числа

δ_{jk}	символ Кронекера
$\chi_S(\cdot)$	характеристическая функция множества S
(x,y)	наибольший общий делитель x и y
$a \equiv b \pmod q$	сравнение по модулю q
$a \bmod q$	остаток по модулю q
$\boxed{\frac{a}{b}}$	представление рационального числа a/b в виде несократимой дроби
$\mathbf{Pr}[A]$	вероятность события A
$\mathbf{P}(\cdot\vert\cdot)$	условная вероятность (в различных контекстах)
$\mathbf{P}(\rho,\mathcal{M})$	квантовая вероятность (с. 78)

Обозначения матриц:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad K = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix},$$

матрицы Паули: $\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Обозначения сложностных классов:

P	(с. 22)	MA	(с. 106)	BQNP	(с. 106)
P/poly	(с. 25)	Π_k	(с. 44)	PP	(с. 74)
BPP	(с. 37)	Σ_k	(с. 44)	PSPACE	(с. 22)
NP	(с. 30)	BQP	(с. 74)		

Введение

Все компьютеры, начиная от так и не построенной «аналитической машины» Чарльза Бэббиджа[1)] и кончая Cray'ем, основаны на одних и тех же принципах. С логической точки зрения компьютер состоит из битов (переменных, принимающих значения 0 или 1), а программа — это последовательность операций, каждая из которых использует небольшое число битов. Конечно, новые компьютеры работают быстрее старых, но прогресс в этом направлении имеет предел. Трудно предположить, что размер транзистора или аналогичного элемента будет меньше 10^{-8} см (диаметр атома водорода), а рабочая частота — больше 10^{15} Гц (частота атомных переходов). Так что даже суперкомпьютеры будущего не смогут решать вычислительные задачи, имеющие *экспоненциальную* сложность. Рассмотрим, например, задачу о разложении целого числа x на простые множители. Очевидный способ — это попробовать разделить x на числа от 2 до $\sqrt{x}$. Если число x имеет n знаков в двоичной записи, то придётся перебрать $\sim \sqrt{x} \sim 2^{n/2}$ вариантов. Существует хитроумный алгоритм, решающий ту же задачу примерно за $\exp(cn^{1/3})$ шагов ($c = \mathrm{const}$). Даже в этом случае, чтобы разложить на множители число из миллиона знаков, не хватит времени жизни Вселенной. (Возможно, есть и более эффективные алгоритмы, но от экспоненты, по-видимому, избавиться не удастся.)

Существует, однако, другой способ ускорить процесс вычисления для некоторых специальных классов задач. Дело в том, что обычные компьютеры не используют всех возможностей, предоставляемых природой. Это утверждение может показаться слишком очевидным: в природе есть множество процессов, совершенно непохожих на операции с нулями и единицами. Можно попытаться использовать эти процессы

[1)] Бэббидж начал работу над проектом «аналитической машины» в 1833 году. Преполагалось, что, в отличие от уже существовавших в то время вычислительных устройств, это будет *универсальный* компьютер. Бэббидж посвятил разработке компьютера всю жизнь, но ему так и не удалось осуществить свою мечту. (Более простая, но неуниверсальная «разностная машина» была построена частично, но проект был вполне реалистичен — в 1991 году машина была полностью воспроизведена по чертежам Бэббиджа.)

для создания аналоговой вычислительной машины. Например, интерференция света может использоваться для вычисления преобразования Фурье. Однако в большинстве случаев выигрыш в скорости не является принципиальным, т. е. слабо зависит от размера устройства. Причина заключается в том, что уравнения *классической* физики (например, уравнения Максвелла) эффективно решаются на обычном цифровом компьютере. Что значит эффективно? Вычисление интерференционной картины может занять в миллионы раз больше времени, чем реальный эксперимент, потому что скорость света велика, а длина волны мала. Однако с увеличением размера моделируемой физической системы количество необходимых вычислительных операций растёт не слишком быстро — степенным, или, как принято говорить в теории сложности, *полиномиальным* образом. (Как правило, число операций пропорционально величине Vt, где V — объём, а t — время.) Таким образом, классическая физика слишком «проста» с вычислительной точки зрения.

Квантовая механика устроена в этом смысле гораздо интереснее. Рассмотрим, например, систему из n спинов. Каждый спин обладает двумя базисными состояниями ($0 =$ «спин вверх» и $1 =$ «спин вниз»), а вся система имеет 2^n *базисных* состояний $|x_1,\dots,x_n\rangle$ (каждая из переменных $x_1,\dots,x_n$ принимает значение 0 или 1). Согласно общим принципам квантовой механики, возможными состояниями системы являются также *суперпозиции* вида $\sum_{x_1,\dots,x_n} c_{x_1,\dots,x_n}|x_1,\dots,x_n\rangle$, где $c_{x_1,\dots,x_n}$ — комплексные числа, называемые *амплитудами*. Знак суммы здесь нужно понимать чисто формально. Суперпозиция является новым математическим объектом — вектором в 2^n-мерном комплексном пространстве. Квадрат модуля амплитуды, $|c_{x_1,\dots,x_n}|^2$, равен вероятности обнаружить систему в базисном состоянии $|x_1,\dots,x_n\rangle$ при измерении значений переменных x_j. (Отметим, что такое измерение разрушает суперпозицию.) Следовательно, должно выполняться условие $\sum_{x_1,\dots,x_n}|c_{x_1,\dots,x_n}|^2 = 1$. Итак, общее состояние системы (т. е. суперпозиция) — это вектор единичной длины в 2^n-мерном комплексном пространстве. Изменение состояния за определённый промежуток времени описывается унитарной матрицей размера $2^n \times 2^n$. Если промежуток времени очень мал ($\ll \hbar/J$, где J — энергия взаимодействия), то эта матрица устроена достаточно просто; каждый из её элементов можно легко вычислить, зная взаимодействие между спинами. Если же мы хотим узнать изменение состояния системы за большой промежуток времени, то придётся перемножать такие матрицы. Для этого требуется экспоненциально большое число операций. В настоящее время неизвестно никакого способа упростить данное вычисление и, скорее всего, моделирование квантовой механики является экспоненциально

сложной вычислительной задачей. Однако то же самое утверждение можно сформулировать иначе: квантовая система эффективно «решает» сложную вычислительную задачу — моделирует саму себя.

Можно ли использовать квантовые системы для решения других вычислительных задач? Какова должна быть математическая модель квантового компьютера, в той же степени не зависящая от физической реализации, что и модели классических вычислений[2)]? Эти вопросы, по-видимому, впервые были поставлены в книге Ю. И. Манина «Вычислимое и невычислимое» (1980 г.). Они обсуждались также в работах Р. Фейнмана и других авторов. В 1985 году Д. Дойч [27] предложил конкретную математическую модель — квантовую машину Тьюринга, а в 1989 году — эквивалентную, но более удобную модель — квантовые схемы [28, 47].

Что такое квантовая схема? Пусть в нашем распоряжении имеется N спинов, каждый из которых находится в отдельном ящичке и идеально изолирован от окружающего мира. В каждый момент времени мы можем выбрать, по нашему усмотрению, любые два спина и подействовать на них любой унитарной матрицей 4×4. Последовательность таких операций называется *квантовой схемой*. Каждая операция определяется парой номеров спинов и шестнадцатью комплексными числами, поэтому квантовую схему можно записать на бумаге. Это своего рода программа для квантового компьютера.

Чтобы использовать квантовую схему для вычисления функции[3)] $F\colon \mathbb{B}^n \to \mathbb{B}^m$, нужно уметь вводить входные данные, проделывать вычисления и считывать результат. Ввести в квантовый компьютер последовательность $(x_1,\dots,x_n)$ нулей и единиц — значит приготовить начальное состояние $|x_1,\dots,x_n,0,\dots,0\rangle$. (Объём входных данных n обычно меньше общего количества «ячеек памяти», т. е. спинов, N. Оставшиеся ячейки заполняются нулями.) К начальному состоянию применяется квантовая схема, зависящая от *решаемой задачи*, но не от конкретных входных данных. В итоге возникает квантовое состояние

$$|\psi(x_1,\dots,x_n)\rangle = \sum_{y_1,\dots,y_N} c_{y_1,\dots,y_N}(x_1,\dots,x_n)\,|y_1,\dots,y_N\rangle,$$

2) Наиболее известной математической моделью обычного компьютера является машина Тьюринга. Большинство моделей *полиномиально эквивалентны* друг другу, т. е. задача, разрешимая за L шагов в одной модели, разрешима за cL^k шагов в другой модели, где c и k — константы.

3) Любая вычислительная задача может быть представлена в таком виде. Например, если мы хотим решать задачу о разложении целого числа x на простые множители, то $(x_1,\dots,x_n) = x$ (в двоичной записи), а $F(x)$ — список простых множителей (в некоторой двоичной кодировке).

зависящее от $(x_1, \ldots, x_n)$. Теперь нужно считать результат. Предполагаем, что ответ должен содержаться в первых m битах строки $(y_1, \ldots, y_N)$, т.е. мы ищем такие $(y_1, \ldots, y_m)$, что $(y_1, \ldots, y_m) = F(x_1, \ldots, x_n)$. Для получения ответа производится измерение значений всех спинов. Результатом измерения может быть *любая* последовательность нулей и единиц $(y_1, \ldots, y_N)$, *вероятность* получить её равна $|c_{y_1,\ldots,y_N}(x_1, \ldots, x_n)|^2$. Таким образом, квантовый компьютер может, с некоторой вероятностью, дать любой ответ. Квантовая схема является «правильной» для данной функции F, если правильный ответ $(y_1, \ldots, y_m) = F(x_1, \ldots, x_n)$ получается с вероятностью, достаточно близкой к единице. Повторив всё вычисление несколько раз и выбрав тот ответ, который встречается чаще, можно снизить вероятность ошибки до сколь угодно малой величины.

Мы только что сформулировали (опуская некоторые подробности) математическую модель квантового вычисления. Теперь естественно задать два вопроса.

1. Для каких задач квантовое вычисление дает выигрыш по сравнению с классическим?
2. Какую систему можно использовать для физической реализации квантового компьютера? (Это не обязательно должна быть система спинов.)

По поводу первого вопроса сейчас известно следующее. Во-первых, на квантовом компьютере можно моделировать любую квантовую систему за *полиномиальное* число шагов. Это позволит (при наличии квантового компьютера) предсказывать свойства молекул и кристаллов, проектировать микроскопические электронные устройства размером в несколько десятков ангстрем. (Сейчас такие устройства находятся на пределе технологических возможностей, но в будущем они, вероятно, будут применяться в обычных компьютерах.) Второй пример — разложение на множители и аналогичные теоретико-числовые задачи, связанные с абелевыми группами. В 1994 году П. Шор (P. Shor) придумал квантовый алгоритм[4)], позволяющий разложить на простые множители число из n знаков за $n^3(\log n)^k$ шагов ($k = \mathrm{const}$). Этот красивый результат может иметь скорее вредное, чем полезное применение: разлагая числа на множители, можно подбирать ключи к шифрам, подделывать электронные подписи и т. д. (Впрочем, трудности на пути создания квантового компьютера столь велики, что в течение

[4)] Если не вдаваться в тонкости, квантовый алгоритм — это то же самое, что квантовая схема. Отличие состоит в том, что схема определена для задачи фиксированного размера ($n = \mathrm{const}$), а алгоритм определён для всех n сразу.

ближайших 10 лет пользователи шифров могут спать спокойно.) Третий пример — это поиск нужной записи в неупорядоченной базе данных. Здесь выигрыш не столь значителен: для нахождения одной записи из N требуется порядка $\sqrt{N}$ операций на квантовом компьютере вместо N операций на классическом. На этом список известных примеров заканчивается — не потому что квантовые компьютеры бесполезны для других задач, а потому что теория квантовых вычислений пока не разработана. Будем надеяться, что скоро появятся новые математические идеи, которые позволят придумать новые примеры.

Физическая реализация квантового компьютера — чрезвычайно интересная, но сложная задача. Ещё несколько лет назад высказывались сомнения в её принципиальной разрешимости. Дело в том, что любое унитарное преобразование можно реализовать лишь с некоторой точностью. Кроме того, систему спинов или аналогичную квантовую систему нельзя полностью защитить от возмущений со стороны окружающей среды. Всё это должно приводить к погрешностям, которые будут накапливаться в процессе вычисления. Через $L \sim \delta^{-1}$ шагов (где δ — точность каждого унитарного преобразования) вероятность ошибки станет порядка единицы. К счастью, эту трудность можно преодолеть, используя *квантовые коды, исправляющие ошибки*. В 1996 году П. Шор предложил схему коррекции ошибок в процессе квантового вычисления (fault-tolerant quantum computation), которая была вскоре усовершенствована. Окончательный результат состоит в следующем. Существует некоторое пороговое значение точности δ_0, такое что при $\delta < \delta_0$ возможно сколь угодно длинное квантовое вычисление. Однако при $\delta > \delta_0$ ошибки накапливаются быстрее, чем их удаётся исправлять. По разным оценкам, δ_0 лежит в интервале от 10^{-2} до 10^{-6} (точное значение зависит от характера возмущений и используемой схемы коррекции ошибок).

Итак, *принципиальных* препятствий для реализации квантового компьютера нет. Однако задача столь трудна, что её можно сравнить с задачей об управляемом термоядерном синтезе. В самом деле, необходимо удовлетворить нескольким почти несовместимым требованиям.

1. Элементы квантового компьютера — квантовые биты (спины или что-то подобное) — должны быть изолированы друг от друга и от окружающей среды.
2. Необходимо иметь возможность избирательного воздействия на пару квантовых битов. (Возможно, потребуется несколько типов воздействия на одну и ту же пару, описываемых различными унитарными операторами.)
3. Каждый из унитарных операторов должен быть реализован с точностью $\delta < \delta_0$ (см. выше).

4. Реализуемые операторы должны быть достаточно нетривиальны. Точнее говоря, они должны образовывать *полный базис*, чтобы любой другой оператор в определенном смысле выражался через них.

В настоящее время существует несколько подходов к проблеме реализации квантового компьютера.

1. Отдельные атомы или ионы. Это первая и наиболее хорошо разработанная идея, она существует в нескольких вариантах. Для представления квантового бита можно использовать как обычные электронные уровни, так и уровни тонкой и сверхтонкой структуры. Имеется экспериментальная техника, позволяющая удерживать отдельный ион или атом в ловушке из постоянного магнитного или переменного электрического поля в течение длительного времени (порядка 1 часа). Ион можно «охладить» (т. е. погасить колебательное движение) при помощи лазерного луча. Подбирая длительность и частоту лазерных импульсов, можно приготовить произвольную суперпозицию основного и возбуждённого состояний. Таким образом, управлять отдельным ионом достаточно легко. В ловушку можно также поместить два или большее число ионов на расстоянии несколько микрон друг от друга и управлять каждым из них в отдельности. Однако организовать взаимодействие между ионами достаточно трудно. Для этой цели предложено использовать коллективные колебательные моды ионов (обычные механические колебания с частотой в несколько мегагерц). Другой способ (для нейтральных атомов): поместить атомы в отдельные электромагнитные резонаторы, связанные друг с другом (пока непонятно, как это реализовать технически). Наконец, третий способ: при помощи нескольких лазерных лучей можно создать периодический потенциал («оптическую решётку»), удерживающий невозбуждённые атомы. При этом возможна ситуация, когда возбуждённые атомы могут свободно двигаться. Таким образом, возбуждая на короткое время один из атомов, мы заставляем его взаимодействовать с соседями. Это направление экспериментальной физики сейчас быстро развивается и, по-видимому, имеет большие перспективы.

2. Ядерный магнитный резонанс. В молекуле с несколькими *различными* ядерными спинами произвольное унитарное преобразование можно реализовать при помощи последовательности импульсов магнитного поля. Это было проверено экспериментально при комнатной температуре. Однако для приготовления начального состояния необходима температура $< 10^{-3}$K. Помимо трудностей с охлаждением, при такой температуре возрастают нежелательные взаимодействия молекул друг с другом. Кроме того, непонятно, как избирательно воздействовать на данный спин, если в молекуле есть несколько одинаковых спинов.

3. Системы сверхпроводящих гранул. При сверхнизких температурах единственной степенью свободы микроскопической сверхпроводящей гранулы (диаметром в несколько сотен ангстрем) является её заряд. Он может изменяться на величину, кратную двум зарядам электрона (поскольку электроны в сверхпроводнике связаны в пары). Меняя внешний электрический потенциал, можно добиться такой ситуации, когда два зарядовых состояния будут иметь почти одинаковую энергию. Эти два состояния можно использовать в качестве базисных состояний квантового бита. Гранулы взаимодействуют между собой посредством джозефсоновских контактов и взаимной электрической ёмкости. Этим взаимодействием можно управлять. Основная трудность состоит в том, что нужно управлять каждой гранулой в отдельности, причем с высокой точностью. По-видимому, этот подход перспективен, но для его реализации потребуется создание новой технологии.

4. Анионы. Анионы — это особые возбуждения в двумерных квантовых системах, в частности, в двумерной электронной жидкости в магнитном поле. Один из авторов (А. К.) считает этот подход наиболее интересным (поскольку он же его и придумал [32]), поэтому опишем его более подробно.

Основной проблемой при создании квантового компьютера является необходимость реализации унитарных преобразований с точностью $\delta < \delta_0 \sim 10^{-2} \div 10^{-6}$. Для этого, как правило, требуется контролировать параметры системы с ещё большей точностью. Однако можно представить ситуацию, когда высокая точность достигается автоматически, т. е. исправление ошибок происходит на физическом уровне. Примером являются двумерные системы с анионными возбуждениями.

Все частицы в трёхмерном пространстве являются либо бозонами, либо фермионами. Волновая функция бозонов не меняется при перестановке двух частиц, а волновая функция фермионов умножается на -1. В любом случае при возвращении каждой из частиц на прежнее место состояние системы не меняется. В двумерных системах возможно более сложное поведение. Прежде всего заметим, что речь пойдёт не об элементарных частицах типа электрона, а о возбуждениях, или дефектах в двумерной электронной жидкости. Такие возбуждения похожи на «настоящие» (т. е. элементарные) частицы, но обладают некоторыми необычными свойствами. Возбуждение может иметь дробный электрический заряд (например, 1/3 от заряда электрона). При движении одного возбуждения *вокруг другого* состояние окружающей их электронной жидкости меняется строго определённым образом, зависящим от типа возбуждений и от *топологии* пути, но не от конкретной траектории. В простейшем случае волновая функция домножается на число ($e^{2\pi i/3}$

для анионов в двумерной электронной жидкости в магнитном поле при факторе заполнения 1/3). Возбуждения с таким свойством называются *абелевыми анионами*. Другой пример абелевых анионов описан (на математическом языке) в разделе 14.11.

Более интересны *неабелевы анионы*, которые пока не наблюдались экспериментально. (Теория предсказывает существование неабелевых анионов в двумерной электронной жидкости в магнитном поле при факторе заполнения 5/2.) При наличии нескольких неабелевых анионов состояние электронной жидкости является вырожденным, причем кратность вырождения экспоненциально зависит от числа анионов. Другими словами, существует не одно, а много состояний, которые могут образовывать произвольные квантовые суперпозиции. На такую суперпозицию нельзя никак воздействовать, не перемещая анионы, поэтому она идеально защищена от возмущений. Если обвести один анион вокруг другого, суперпозиция подвергнется определённому унитарному преобразованию. Это преобразование является *абсолютно точным*. (Ошибка может возникнуть, только если анион «вырвется у нас из рук» вследствие квантового туннелирования).

На первый взгляд, проект с использованием анионов выглядит наименее реалистично. Прежде всего, абелевы анионы не годятся для квантовых вычислений, а неабелевы ещё только предстоит найти в эксперименте. Для реализации квантового компьютера нужно контролировать *каждую* из частиц, которые будут двигаться на расстояниях порядка долей микрона друг от друга. Это чрезвычайно сложная техническая задача. Однако, с учётом высоких требований к точности, осуществить любой из перечисленных выше подходов ничуть не легче. Кроме того, идея *топологического квантового вычисления*, лежащая в основе подхода с анионами, может воплотиться каким-либо другим способом. Например, защищённая от возмущений квантовая степень свободы может возникнуть на конце «квантовой проволоки» (одномерного проводника с нечётным числом распространяющихся электронных мод, находящегося в контакте с трёхмерным сверхпроводником).

Итак, идея квантового компьютера выглядит столь же заманчиво, сколь нереалистично. Наверное, так же воспринимался проект обычного компьютера во времена Чарльза Бэббиджа, изобретение которого было реализовано лишь сто лет спустя. Будем надеяться, что в наше время научно-технический прогресс идет быстрее, поэтому не придётся ждать так долго. Возможно, достаточно одной свежей идеи плюс несколько лет на разработку новой технологии...

Часть I

Классические вычисления

1. Что такое алгоритм?

Неформально *алгоритм* — это однозначно определённая совокупность инструкций по преобразованию исходных данных в результат, причём все инструкции *элементарны*, т. е. при их выполнении «нам придётся только механически следовать предписаниям, как если бы мы были роботами: от нас не потребуется ни понимания, ни искусства, ни изобретательности» [5, с. 270].

Обычно подразумевается, что каждый алгоритм решает какую-то *вычислительную задачу*. С формальной точки зрения, вычислительная задача — это функция

$$F : \text{входные данные} \mapsto \text{результат},$$

как правило, частично определенная.

Что такое «входные данные» и «результат»? Рассмотрим, например, задачу об умножении двух многочленов с целыми коэффициентами. Тогда входные данные — это пара многочленов. Проблема в том, как записать эти многочлены, чтобы их можно было ввести в компьютер. Машины Тьюринга, которые мы рассматриваем ниже, понимают лишь конечные последовательности символов (слова) из некоторого конечного множества $\mathcal{A}$, называемого *внешним алфавитом*. Поэтому строгая формулировка вычислительной задачи должна включать в себя алфавит и способ кодировки входных данных. Например, можно записывать пару многочленов с использованием 10 цифр, символа переменной x, знаков $+, -, *$ и скобок: $(x**2-5)(-4*x+1)$. В другой кодировке, коэффициенты записываются в двоичной системе и перечисляются через запятую; два многочлена разделяются звездочкой: $1, 0, -101*-100, 1$. Таким образом, мы имеем две различные вычислительные задачи. С практической точки зрения, обе задачи эквивалентны, поскольку перевод из одной кодировки в другую осуществляется с помощью *полиномиального алгоритма*. Пока определения полиномиального алгоритма и нас нет (оно появится в разделе 1.4), давайте будем формалистами: вычислительная задача — это частичная[5)] функция $F\colon \mathcal{A}^* \to \mathcal{A}^*$ (где $\mathcal{A}^*$ обозначает множество конечных слов в алфавите $\mathcal{A}$). Потом мы позволим себе некоторую вольность и будем говорить о задаче умножения многочленов или разложения целого числа на множители, имея ввиду, что все разумные кодировки эквивалентны (т. е. переводятся друг в друга при помощи полиномиальных алгоритмов). Однако нужно помнить, что не всякая кодировка является «разумной». Нехорошо, например, представлять натуральное число n набором из n звездочек, потому что длина такой записи экспоненциально велика по сравнению с двоичной или десятичной записью. Заметим также, что в некоторых задачах нет хорошего выбора «разумной» кодировки, в таких случаях (они нам не встретятся) необходимо указывать кодировку явно всякий раз, когда формулируется задача.

Дадим теперь формальное определение алгоритма.

1.1. Машины Тьюринга. Машина Тьюринга (сокращённо МТ) однозначно задаётся указанием набора $(\mathcal{S}, \sqcup, \mathcal{A}, \mathcal{Q}, q_0, \delta)$, где $\mathcal{S}$, $\mathcal{A}$, $\mathcal{Q}$ — конечные множества, причём $\mathcal{A} \subset \mathcal{S}$; $\sqcup$ — некоторый элемент $\mathcal{S} \setminus \mathcal{A}$; q_0 — некоторый элемент $\mathcal{Q}$, а δ — некоторая (частичная, вообще говоря) функция из $\mathcal{Q} \times \mathcal{S}$ в $\mathcal{Q} \times \mathcal{S} \times \{-1, 0, 1\}$.

Составляющие части МТ называются так:

$\mathcal{S}$ — алфавит,
$\sqcup$ — пустой символ (или пробел),
$\mathcal{A}$ — внешний алфавит,
$\mathcal{Q}$ — множество состояний управляющего устройства,
q_0 — начальное состояние,
δ — функция переходов.

Состояние МТ задаётся тройкой (σ, p, q), где σ — бесконечное слово в алфавите $\mathcal{S}$, т. е. произвольная последовательность $s_0, \ldots, s_n, \ldots$ элементов $\mathcal{S}$; p — неотрицательное целое число; $q \in \mathcal{Q}$. Символы слова σ будем, как это принято, представлять записанными на *ленте*, разбитой на *ячейки*, по ячейке на символ. На ленте также имеется *головка*, которая расположена над ячейкой с номером p. Наглядно это изображается так:

[5)] Под частичной функцией на множестве X здесь и далее понимается функция, область определения которой содержится в X.

Положение головки				∇	
Ячейки	s_0	s_1	$\ldots$	s_p	$\ldots$
Номера ячеек	0	1		p	

Помимо ленты машина Тьюринга имеет *управляющее устройство*, состояние которого задаётся элементом q множества $\mathcal{Q}$.

Состояния МТ меняются дискретно. За один такт работы управляющее устройство выполняет следующие действия (полагаем, что МТ находится в состоянии (σ, p, q)):

а) *читает* символ, находящийся под головкой (т. е. определяет s_p);

б) *вычисляет* значение функции переходов: $\delta(q, s_p) = (q', s, \Delta p)$ (если функция переходов на паре (q, s_p) не определена, то останавливает машину Тьюринга);

в) *записывает* на ленту в ячейку p символ s, сдвигает головку на Δp и переходит в состояние q' (другими словами, новое состояние машины задаётся тройкой $((s_0, \ldots, s_{p-1}, s, s_{p+1}, \ldots), p + \Delta p, q')$);

г) если $p + \Delta p < 0$, то останавливает машину.

Пожалуй, всякий согласится, что эти действия не требуют ни понимания, ни искусства, ни изобретательности.

Работа машины Тьюринга будет всегда начинаться из состояния $(\alpha\sqcup\ldots, 0, q_0)$, где за конечным словом α, состоящим из символов внешнего алфавита, следует бесконечное слово, целиком состоящее из пустых символов. Слово α будем называть *входом* МТ. В любой момент времени слово, записанное на ленте, однозначно записывается в виде $\sigma\sqcup\ldots$, где последний символ слова σ — не пустой, а за ним идут только пустые символы. Будем называть слово σ *используемой частью ленты*.

Выполняя один такт работы за другим, машина Тьюринга порождает последовательность состояний

$$(\sigma_0, 0, q_0),\ (\sigma_1, p_1, q_1),\ (\sigma_2, p_2, q_2),\ \ldots$$

Если МТ останавливается, используемая часть ленты в достигнутом перед остановкой состоянии называется *результатом* работы МТ.

1.2. Вычислимые функции и разрешимые предикаты. Каждая машина Тьюринга M *вычисляет* частичную функцию φ_M из $\mathcal{A}^*$ в $\mathcal{A}^*$, отображающую вход α в результат работы МТ на входе α при условии, что результат работы является словом во внешнем алфавите. Для входов, на которых машина не останавливается или результат содержит символы из $\mathcal{S}\setminus\mathcal{A}$, функция φ_M не определена. Из определения ясно, что любая МТ вычисляет ровно одну функцию (быть может, нигде не определённую).

Определение 1.1. Частичная функция f из $\mathcal{A}^*$ в $\mathcal{A}^*$ называется *вычислимой*, если существует машина Тьюринга M, для которой $\varphi_M = f$. При этом будем говорить, что f *вычислима* на M.

Не все функции вычислимы. Это ясно из сравнения мощности множества функций (континуум) и мощности множества машин Тьюринга (счётное множество). Более интересные примеры см. в задачах 1.3–1.5.

Под *предикатом* будем понимать некоторое условие, которое выполняется (предикат истинен) или не выполняется (предикат ложен) для каждого слова из $\mathcal{A}^*$. Определённые таким образом, предикаты легко отождествляются с *языками* (подмножествами слов в $\mathcal{A}^*$) — предикату соответствует множество слов, на которых он истинен. Каждому предикату сопоставим *характеристическую функцию*, которая равна 1 на множестве слов, для которых предикат истинен, и равна 0 на множестве слов, для которых предикат ложен. Мы будем обозначать характеристическую функцию так же, как и сам предикат. Предикат *разрешим*, если его характеристическая функция вычислима. О машине Тьюринга, вычисляющей характеристическую функцию предиката, будем говорить, что она даёт ответ («да» или «нет») на вопрос «истинно ли значение предиката на входе α?»

Понятия вычислимой функции и разрешимого предиката будут использоваться и для функций (предикатов) от многих переменных.

Пусть $\mathcal{A}$ — некоторый алфавит, а n — натуральное число. Пусть M — машина Тьюринга с внешним алфавитом $\mathcal{A} \cup \{\#\}$. Построим частичную функцию $\varphi_{M,n}$ из $(\mathcal{A}^*)^n$ в $\mathcal{A}^*$ так: $\varphi_{M,n}(\alpha_1, \ldots, \alpha_n) = y$, если результат работы машины M на входе $\alpha_1\#\alpha_2\#\ldots\#\alpha_n\#$ совпадает со словом y. Если машина не останавливается или на ленте записано что-нибудь не то (например, символы не из алфавита $\mathcal{A}$ и т. п.), то $\varphi_{M,n}(\alpha_1, \ldots, \alpha_n)$ не определена.

Определение 1.2. Частичная функция f из $(\mathcal{A}^*)^n$ в $\mathcal{A}^*$ называется *вычислимой*, если существует машина Тьюринга M, для которой $\varphi_{M,n} = f$.

Предикат от нескольких переменных *разрешим*, если его характеристическая функция вычислима.

Нас будут интересовать *ресурсы*, требующиеся для вычислений. Два важнейших ресурса — *время* и *память*. Будем говорить, что машина Тьюринга M работает за время $T_M(n)$, если максимальное (по всем входам длины n) количество тактов, которое проработает M до остановки, равно $T_M(n)$. Аналогично, машина Тьюринга M работает на памяти $S_M(n)$, если наиболее удалённое от начала ленты положение головки при вычислениях на входах длины n равно $S_M(n)$.

1.3. Вычисления на машинах Тьюринга. Очевидно, что МТ задаёт алгоритм в смысле приведенного выше неформального определения. Обратное утверждение называется *тезисом Чёрча*:

«любой алгоритм может быть реализован машиной Тьюринга».

Не вдаваясь в обсуждение тезиса Чёрча, заметим, что в настоящее время нет серьёзных оснований подвергать его сомнению. Все известные в настоящее время алгоритмы реализуются машинами Тьюринга. Подробное изложение теории алгоритмов читатель может найти в книгах [1, 5, 6, 10, 13, 16, 17]. Мы же ограничимся краткими неформальными пояснениями приёмов программирования на машинах Тьюринга.

Возможности машины Тьюринга при таком неформальном обсуждении — это возможности человека с ограниченной памятью, карандашом и ластиком, которому вручили неограниченной (бесконечной) толщины тетрадь. Страницы тетради имеют ограниченный размер (все возможные варианты заполнения страницы образуют алфавит МТ при строгом описании). На первых страницах тетради записано входное слово — по одному символу (из внешнего алфавита) на страницу. Человек может листать тетрадь, стирать символы, записывать новые. Заканчивается эта его деятельность тем, что он закрывает тетрадь (сдвиг головки влево в положении 0) и возвращает результат своей работы.

Представив себя в такой ситуации, легко сообразить, что можно, запоминая несколько символов, выполнять любые действия на ограниченном количестве подряд идущих страниц; можно ставить на страницах дополнительные пометки (ограниченное количество на каждой); можно листать тетрадь, пока не найдётся нужная пометка; можно копировать символы на свободные страницы тетради. Свободные места на страницах можно также использовать для хранения вспомогательных слов произвольной длины (как и входное слово, они записываются по одному символу на страницу), над которыми может поработать другой человек (это называется «вызов подпрограммы»). В частности, эти вспомогательные слова позволяют поддерживать *счётчики* (целочисленные переменные). Используя счётчики, можно адресоваться к ячейкам памяти по их номеру.

Поскольку описание любой машины Тьюринга является конечным объектом, его можно закодировать словом в некотором алфавите. В нашей неформальной ситуации легко понять, что существует *универсальная машина Тьюринга* U, которая, получая на вход пару $([M], x)$, даёт выход $\varphi_M(x)$. Здесь через $[M]$ обозначено описание некоторой машины Тьюринга M. Действительно, предположим, что тетрадь начинается со страниц, где записаны инструкции по работе. Тогда выполнять эти инструкции можно следующим образом: пометим текущую страницу; пролистаем тетрадь до начального раздела, содержащего инструкции; найдём нужную инструкцию; вернёмся назад и выполним её.

1.4. Сложностные классы. Не для всякой вычислимой функции можно реально осуществить вычисление её значения. Существование алгоритма не означает, что нам по силам проделать все предписанные действия. Препятствие может заключаться в том, что требуемые для этого вычисления ресурсы слишком велики. Поэтому нас интересует не столько существование алгоритма для решения задачи, сколько существование *эффективного алгоритма.*

Формализовать это понятие непросто. Принятый сейчас способ состоит в том, что выделяются классы тех функций или предикатов, вычисление которых возможно при задаваемых ограничениях на потребляемые ресурсы. Принадлежность функции тому или иному *сложностному классу* служит удобной характеристикой возможностей её эффективного вычисления, не связанной с конкретными реализациями алгоритмов.

Наиболее важные классы получаются, если накладывать ограничения на рост времени работы и/или используемой памяти в зависимости от длины входного слова. А наиболее важное различие между эффективными и неэффективными вычислениями задаётся функциями *полиномиального роста.* Функция $f(n)$ — полиномиального роста, если для некоторой константы d при достаточно больших n выполняется неравенство $f(n) \leqslant n^d$. В этом случае будем использовать обозначение $f(n) = \mathrm{poly}(n)$.

Определение 1.3. Предикат f на множестве $\mathbb{B}^*$ принадлежит классу P (и называется *полиномиально вычислимым*), если его характеристическая функция вычислима на машине Тьюринга M, для которой $T_M(n) = \mathrm{poly}(n)$.

Определение 1.4. Предикат f на множестве $\mathbb{B}^*$ принадлежит классу PSPACE, если его характеристическая функция вычислима на машине Тьюринга M, для которой $S_M(n) = \mathrm{poly}(n)$.

По аналогии с предикатами можно определить и *функции, вычислимые за полиномиальное время*, и *функции, вычислимые на полиномиальной памяти*. Для классов таких функций также используются обозначения P и PSPACE, так что точный смысл этих обозначений нужно восстанавливать из контекста.

1.5. Схемы. Схема (булева) — это способ вычислить функцию $f\colon \mathbb{B}^n \to \mathbb{B}^m$. Помимо исходных переменных $x_1, \ldots, x_n$, для которых вычисляется значение f, схема использует некоторое количество вспомогательных переменных $y_1, \ldots, y_s$ и некоторый набор (*базис*) булевых (т.е. принимающих значения 0 или 1) функций $\mathcal{F}$. Схема S в базисе $\mathcal{F}$ определяется последовательностью *присваиваний* $Y_1, \ldots, Y_s$. Каждое присваивание Y_i имеет вид $y_i := f_j(u_{k_1}, \ldots, u_{k_r})$, где $f_j(\cdot) \in \mathcal{F}$, а переменная u_{k_p} $(1 \leqslant p \leqslant r)$ — это либо одна из исходных переменных x_t $(1 \leqslant t \leqslant n)$, либо вспомогательная переменная y_l с меньшим номером $(1 \leqslant l < i)$. Таким образом, для каждого набора значений исходных переменных последовательное выполнение присваиваний, входящих в схему, однозначно определяет значения всех вспомогательных переменных. *Результатом вычисления* считаются значения последних m переменных $y_{s-m+1}, \ldots, y_s$.

Схема *вычисляет* функцию f, если для любых значений $x_1, \ldots, x_n$ результатом вычисления является $f(x_1, \ldots, x_n)$.

Схема называется *формулой*, если каждая вспомогательная переменная используется в правой части присваиваний только один раз. (Обычные математические формулы именно так задают последовательность присваиваний: «внутри» формул не принято использовать ссылки на их части или другие формулы.)

Схему можно также представлять в виде ориентированного ациклического графа, у которого вершины входной степени 0 (*входы*) помечены исходными переменными; остальные вершины (*функциональные элементы*) помечены функциями из базиса (при этом входная степень вершины должна совпадать с количеством аргументов её пометки); рёбра помечены числами, указывающими номера аргументов; вершины выходной степени 0 (*выходы*) помечены переменными, описывающими результат работы схемы. Вычисление на графе определяется индуктивно: как только известны значения всех вершин $y_1, \ldots, y_{k_v}$, из которых ведут рёбра в данную вершину v, вершина v получает значение $y_v = f_v(y_1, \ldots, y_{k_v})$, где f_v — базисная функция, которой помечена вершина. При переходе к графу схемы мы опускаем *несущественные присваивания*, которые ни разу не используются на пути к выходным вершинам, так что они никак не влияют на результат вычисления.

Базис называется *полным*, если для любой булевой функции f есть схема в этом базисе, вычисляющая f. Ясно, что в полном базисе можно вычислить произвольную функцию $f\colon \mathbb{B}^n \to \mathbb{B}^m$ (такую функцию можно представить как упорядоченный набор из m булевых функций).

Булева функция может быть задана таблицей значений. Приведём таблицы значений для трёх функций

$$NOT(x) = \neg x,\ \ OR(x_1, x_2) = x_1 \vee x_2,\ \ AND(x_1, x_2) = x_1 \wedge x_2$$

(*отрицание*, *дизъюнкция*, *конъюнкция*), образующих полный базис, который будем считать стандартным. В дальнейшем имеются в виду схемы именно в этом базисе, если явно не указано что-либо иное.

x	NOT
0	1
1	0

x_1	x_2	OR
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	AND
0	0	0
0	1	0
1	0	0
1	1	1

Конъюнкция и дизъюнкция определяются для произвольного числа булевых переменных аналогичным образом: конъюнкция равна 1 только тогда, когда все аргументы равны 1, а дизъюнкция равна 0 только тогда, когда все аргументы равны 0. В стандартном базисе они очевидным образом вычисляются схемами (и даже формулами) размера $n-1$.

Теорема 1.1. *Базис* $\{NOT, OR, AND\}$ *— полный.*

Доказательство. *Литералом* будем называть переменную или её отрицание. Конъюнкцией литералов (это схема и даже формула) легко представить функцию $\chi_u(x)$, которая принимает значение 1 ровно один раз: при $x = u$. Если $u_i = 1$, включаем в конъюнкцию переменную x_i, если $u_i = 0$, то включаем в конъюнкцию $\neg x_i$. Произвольная функция f может быть представлена в виде

$$f(x) = \bigvee_{u:f(u)=1} \chi_u(x). \tag{1.1}$$

В таком случае говорят, что f представлена в *дизъюнктивной нормальной форме* (ДНФ), т. е. как дизъюнкция конъюнкций литералов.[6)] Как уже говорилось, дизъюнкция нескольких переменных выражается формулой в стандартном базисе. □

Размером схемы называется количество присваиваний в схеме. Минимальный размер схемы в базисе $\mathcal{F}$, вычисляющей функцию f, называется *схемной сложностью* функции f в базисе $\mathcal{F}$ и обозначается $c_{\mathcal{F}}(f)$. Переход от одного полного конечного базиса к другому полному конечному базису меняет схемную сложность функций на множитель $O(1)$.

[6)] Далее нам ещё потребуется и *конъюнктивная нормальная форма* (КНФ) — конъюнкция дизъюнкций литералов.

Так что в асимптотических оценках выбор конкретного полного базиса неважен и поэтому будем использовать обозначение $c(f)$ для схемной сложности f в конечном полном базисе.

Каждый предикат f на множестве $\mathbb{B}^*$ определяет последовательность булевых функций $f_n\colon \mathbb{B}^n \to \mathbb{B}$ следующим образом:

$$f_n(x_1, x_2, \ldots, x_n) = f(x_1 x_2 \ldots x_n),$$

где справа стоит характеристическая функция предиката f.

Определение 1.5. Предикат f принадлежит классу P/poly, если

$$c(f_n) = \mathrm{poly}(n).$$

Теорема 1.2. P $\subset$ P/poly.

Доказательство. Если МТ работает за полиномиальное время, то и память, которую она использует, ограничена полиномом. Поэтому весь процесс вычисления на входном слове x длины n можно представить *таблицей вычисления* размера $T \times S$, где $T = \mathrm{poly}(n)$, $S = \mathrm{poly}(n)$.

$t = 0$		$\Gamma_{0,1}$			
$t = 1$					
	...				
$t = j$		$\Gamma'_л$	Γ'	$\Gamma'_п$	
$t = j+1$			Γ		
	...				
$t = T$	...				

$\underbrace{\hspace{10em}}_{S\ \text{клеток}}$

Строка с номером j таблицы задаёт состояние МТ после j тактов работы. Символы $\Gamma_{j,k}$, записанные в таблице, принадлежат алфавиту $\mathcal{S} \times \{\varnothing \cup \mathcal{Q}\}$. Символ $\Gamma_{j,k}$ определяет пару (символ, записанный в k-й ячейке после j тактов работы; состояние управляющего устройства после j тактов работы, если головка находится над k-й ячейкой, в противном случае второй элемент пары — $\varnothing$). Для простоты также считаем, что если вычисление заканчивается при некотором входе за $T' < T$ тактов, то строки с номерами, бóльшими T', повторяют строку с номером T'.

Построить схему, вычисляющую значения предиката на словах длины n, можно следующим образом. *Состояние каждой клетки таблицы* можно закодировать конечным (не зависящим от n) числом булевых переменных. Имеются *локальные правила согласования*, т. е. состояние каждой клетки Γ в строке ниже нулевой однозначно определяется состояниями клеток в предыдущей строке, лежащих непосредственно над данной (Γ'), левее данной ($\Gamma'_л$) и правее данной ($\Gamma'_п$). Каждая переменная, кодирующая состояние клетки Γ, есть функция от переменных, кодирующих состояния клеток $\Gamma'_л$, Γ', $\Gamma'_п$. Все эти функции могут быть вычислены схемами конечного размера. Объединяя эти схемы, получим схему, вычисляющую все переменные, кодирующие состояния клеток таблицы; размер этой схемы будет $O(ST) = O(n^{O(1)})$.

Осталось заметить, что переменные, кодирующие часть клеток нулевой строки, определяются входным словом, а переменные, кодирующие остальные клетки нулевой строки, являются константами. Чтобы узнать результат вычисления, нужно определить символ, записанный в нулевой ячейке ленты в конце вычисления. Без ограничения общности можно считать, что состояния клеток таблицы кодируются так, что одна из кодирующих переменных равна 1 только в том случае, когда в ячейке записана 1. Тогда значение этой переменной для кода $\Gamma_{T,0}$ и будет результатом вычисления. □

Замечание 1.1. Класс P/poly шире класса P. Любой функции от натурального аргумента $\varphi(n)$ со значениями в $\mathbb{B}$ можно сопоставить предикат f_φ по правилу $f_\varphi(x) = \varphi(|x|)$, где $|x|$ обозначает длину слова x. Ограничение такого предиката на слова длины n тождественно равно 0 или 1 (в зависимости от n). Схемная сложность таких функций ограничена константой. Поэтому все такие предикаты по определению принадлежат P/poly, хотя среди них есть и неразрешимые предикаты.

Справедливо следующее усиление теоремы.

Теорема 1.3. *f принадлежит* P *тогда и только тогда, когда*

1) $f \in \mathrm{P/poly}$;
2) *существует МТ, которая по числу n за время* $\mathrm{poly}(n)$ *строит схему вычисления f_n.*

Доказательство. $\Longrightarrow$ Данное в доказательстве теоремы 1.2 описание нетрудно превратить в МТ, которая строит схему вычисления f_n за полиномиальное по n время (схема f_n имеет простую структуру: каждая переменная связана с предыдущими одними и теми же правилами согласования).

$\Longleftarrow$ Столь же просто. Вычисляем размер входного слова. Затем строим по этому размеру схему $S_{|x|}$ вычисления $f_{|x|}$, используя указанную в условии 2) машину. После этого вычисляем $S_{|x|}(x)$ на машине, которая по описанию схемы и значениям входных переменных вычисляет значение схемы за полиномиальное от длины входа время. □

Задачи

1.1. Постройте машину Тьюринга, которая записывает входное двоичное слово в обратном порядке.

1.2. Постройте машину Тьюринга, которая складывает два числа, записанные в двоичной системе. Для определённости считайте, что записи чисел разделены специальным символом алфавита «+».

1.3. Докажите, что не существует алгоритма, который по машине Тьюринга и входу определяет, остановится ли она на этом входе.

1.4. Докажите, что не существует алгоритма, который выписывает одну за другой все машины Тьюринга, которые не останавливаются, будучи запущенными на пустой ленте.

1.5. Пусть $T(n)$ — максимальное время, которое может пройти до остановки машины Тьюринга с n состояниями и n символами алфавита, если её запустить на пустой ленте. Докажите, что функция $T(n)$ растёт быстрее любой вычислимой всюду определённой функции $b(n)$, то есть $\lim[T(n)/b(n)] = +\infty$.

Набор элементарных инструкций у описанных выше машин Тьюринга крайне беден. Имеются разнообразные их обобщения, например, *многоленточные* машины Тьюринга. В отличие от описанной выше, такая МТ имеет несколько (конечное множество) лент, каждая со своей головкой, управляющему устройству доступны символы, находящиеся в ячейках, на которых расположены головки лент. Выделены две ленты: *входная*, с которой разрешается только читать символы, и *выходная*, на которую разрешается только писать символы. Остальные ленты называются *рабочими*. Многоленточная машина называется k-ленточной, если у неё k рабочих лент. Действие за такт работы состоит в изменении состояния управляющего устройства, изменении символов в ячейках под головками и изменении положений головок на лентах (каждая головка сдвигается не более, чем на одну позицию). Это действие однозначно определяется состоянием управляющего устройства и набором символов в ячейках под головками. Если действие выполнить нельзя, машина останавливается.

В начале работы многоленточной машины Тьюринга все ленты пусты, кроме входной, на которой записан вход. Результат работы машины — состояние выходной ленты в конце работы.

Если интересоваться сложностью алгоритмов с точностью до полиномиально ограниченного множителя, многоленточные машины ничего не добавляют по сравнению с описанными выше *машинами с единственной лентой.*

1.6. Докажите, что двухленточную машину Тьюринга, работающую за время $T(n) \geqslant n$ на входах длины n, можно моделировать на машине с единственной лентой за время $O(T^2(n))$.

1.7. Докажите, что трёхленточную машину Тьюринга, работающую за время $T(n) \geqslant n$ на входах длины n, можно моделировать на двухленточной за время $O(T(n) \log T(n))$.

1.8. Пусть M — машина Тьюринга с единственной лентой, которая копирует входное слово (приписывая его копию справа от самого слова). Пусть $T(n)$ — максимальное время её работы на входах длины n. Докажите, что $T(n) \geqslant \varepsilon n^2$ для некоторого ε и для всех n. Что можно сказать про $T'(n)$, которое есть минимальное время её работы на входах длины n?

1.9. Рассмотрим язык программирования, в котором есть всего 100 натуральных переменных, разрешённые операции — прибавление и вычитание 1, разрешённая проверка — не равна ли переменная нулю. Разрешено использовать `if-then-else` и `while`, но рекурсия не разрешена. Докажите, что с помощью программ на таком языке программирования можно вычислять любую вычислимую функцию.

1.10. Постройте алгоритм, определяющий, является ли данный базис полным. Базисные функции заданы таблицами значений.

1.11. Пусть c_n есть максимум сложности $c(f)$ по всем булевым функциям f от n переменных. Докажите, что $1{,}99^n < c_n < 2{,}01^n$ при достаточно больших n.

1.12. *Глубиной* схемы называется максимальное число элементов на пути от входов к выходу. Покажите, что любую функцию можно вычислить схемой глубины не более 3 из элементов NOT и из элементов AND и OR с произвольным числом входов.

1.13. Докажите, что если из схемы глубины $O(\log n)$, вычисляющей функцию $f\colon \mathbb{B}^n \to \mathbb{B}^m$, выбросить все несущественные присваивания, то полученная схема имеет полиномиальный по $n+m$ размер.

1.14. Постройте схему, которая сравнивает два n-битовых числа и имеет размер $O(n)$, а глубину $O(\log n)$.

1.15. а) Постройте схему сложения двух n-битовых чисел размера $O(n)$.

б) Тот же вопрос, если дополнительно потребовать, чтобы глубина схемы была $O(\log n)$.

1.16. Функция $MAJ\colon \mathbb{B}^n \to \mathbb{B}$ равна 1 на двоичных словах, в которых число единиц больше числа нулей, и 0 — на остальных словах. Постройте схему, вычисляющую эту функцию, размер схемы должен быть линеен по n, глубина — $O(\log n \log\log n)$.

1.17. Постройте схему размера $\mathrm{poly}(n)$ и глубины $O(\log^2 n)$, которая проверяет, связаны ли путём две вершины в графе. Граф на m вершинах, которые помечены числами от 1 до m, задаётся $n = m(m-1)/2$ булевыми переменными. Переменная x_{ij}, где $i < j$, определяет, есть ли в графе ребро, соединяющее вершины i и j.

1.18. Пусть схема глубины 3 из элементов NOT и из элементов AND и OR с произвольным числом входов вычисляет сложение n битов по модулю 2 (функция $PARITY$). Покажите, что размер схемы не меньше c^n для некоторого $c > 1$.

1.19. Пусть $f_1, f_2, \ldots$ — последовательность булевых функций от $1, 2, \ldots$ аргументов. Покажите, что следующие два свойства равносильны:

а) существует последовательность вычисляющих эти функции формул, размер которых не превосходит полинома от n;

б) существует последовательность вычисляющих эти функции схем глубины $O(\log n)$ из элементов NOT, AND и OR (с двумя входами).

1.20. Докажите, что существует разрешимый предикат, который принадлежит P/poly, но не принадлежит P.

2. Класс NP: сводимость и полнота

2.1. NP — класс предикатов, вычислимых за полиномиальное время недетерминированными машинами Тьюринга. Термин «недетерминированный» неудачный, но он уже стал стандартным.

Класс NP определён только для предикатов. Говорят, например, что «свойство графа „иметь гамильтонов цикл[7]“ принадлежит NP».

Мы дадим несколько определений этого класса. Первое использует понятие *недетерминированной машины Тьюринга.* Устройство недетерминированной машины (НМТ) такое же, как и обычной (детерминированной), за одним исключением. У управляющего устройства НМТ есть особые недетерминированные состояния. Находясь в таком состоянии, НМТ может выполнить одно из нескольких действий. Поэтому у НМТ вместо функции переходов есть таблица, в которой для каждой пары (состояние управляющего устройства, текущий символ на ленте)

[7] *Гамильтонов цикл* — замкнутый путь в графе, проходящий через все вершины графа ровно по одному разу. Граф, в котором есть хотя бы один гамильтонов цикл, называется *гамильтоновым.*

вместо единственно возможного варианта действий (изменение состояния управляющего устройства, запись символа на ленту, сдвиг головки) указан набор возможных действий. Обычные, детерминированные, состояния становятся частным случаем, когда такой набор состоит ровно из одного варианта действий.

Путь вычисления НМТ определяется выбором одного из возможных переходов на каждом такте работы. Недетерминированность приводит к тому, что путей вычисления для НМТ, работающей на данном входном слове, оказывается много.

Определение 2.1. Предикат L принадлежит классу NP, если существуют НМТ M и полином $p(n)$ такие, что

$L(x) = 1 \quad \Longrightarrow$ существует путь вычисления, дающий ответ «да» за время, не превосходящее $p(|x|)$;

$L(x) = 0 \quad \Longrightarrow$ (1 *вариант определения*) нет указанного выше пути; (2 *вариант определения*) на любом пути вычисления ответа «да» не получается.

Замечание 2.1. Приведенные варианты определения эквивалентны. Чтобы исключить возможность ответов «да» на длинных путях вычисления, достаточно взять НМТ, имитирующую исходную НМТ и подсчитывающую количество сделанных исходной НМТ шагов. Когда число шагов превышает $p(|x|)$, машина останавливается.

Замечание 2.2. В предыдущем рассуждении допущена одна тонкая ошибка. В определении ничего не сказано о полиноме $p(n)$, кроме факта его существования. Если коэффициенты полинома невычислимы, могут возникнуть проблемы с указанным выше способом сведения одного определения к другому (нужно вычислить значение полинома). Чтобы в дальнейшем избегать этой сложности, будем считать, что полином имеет целые коэффициенты.

Замечание 2.3. Непосредственно из определения 2.1 следует, что $\mathrm{P} \subseteq \mathrm{NP}$. Является ли это включение строгим? Довольно интенсивные, хотя и безуспешные, попытки ответить на этот вопрос продолжаются уже почти 30 лет. Недавно С. Смейл включил проблему $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ в число трёх важнейших математических проблем следующего столетия (две другие — гипотеза Римана и гипотеза Пуанкаре).

Второе определение класса NP представляется более естественным. Оно использует понятие полиномиально вычислимого предиката от двух переменных. Определение такого предиката получается комбинацией определений 1.2 и 1.3. Под размером входа для предиката $R(x, y)$ можно понимать также или $|x| + |y|$, или $\max(|x|, |y|)$ — получатся эквивалентные определения.

Определение 2.2. Предикат L принадлежит классу NP, если он представим в форме $L(x) = \exists\, y\, \big((|y| < q(|x|)) \wedge R(x, y)\big)$, где $q(\cdot)$ — полином, а $R(\cdot,\cdot) \in \mathrm{P}$.

Пример 2.1. Пусть $R(x, y) =$ «y есть гамильтонов цикл в графе x». Более точно нужно сказать так: «x есть двоичный код некоторого графа, а y — код гамильтонова цикла в этом графе (используем такое кодирование, при котором код цикла не длиннее кода графа), такие что...». Возьмём $q(n) = n$. Тогда $L(x)$ будет в точности означать, что в графе найдётся гамильтонов цикл.

Теорема 2.1. *Определения* 2.1 *и* 2.2 *эквивалентны.*

Доказательство. Опр. 2.1 $\Longrightarrow$ опр. 2.2. Пусть есть НМТ M и полином $p(n)$ из первого определения. Рассмотрим предикат $R(x, y) =$ = «y есть протокол работы M на возможном пути вычисления со входом x дающего ответ „да“ за время, не превосходящее $p(|x|)$». Длина такого протокола при разумном кодировании линейно зависит от времени вычисления (если использовать в качестве протокола описанную в доказательстве теоремы 1.2 таблицу вычисления, то квадратично), поэтому в качестве $q(n)$ для второго определения можно взять $p(n)$ $(p^2(n))$, умноженный на подходящую константу.

Чтобы закончить доказательство в этом случае, осталось проверить, что $R(\cdot,\cdot) \in \mathrm{P}$. Это почти очевидно. Мы должны проверить протокол некоторой НМТ, работающей за полиномиальное время. Это займёт нас примерно на то же время (совершенно незачем смотреть на одну ячейку экспоненциально долго).

Опр. 2.2 $\Longrightarrow$ опр. 2.1. Пусть есть R, q из определения 2.2. Построим M для определения 2.1. Она работает в два этапа.

Вначале M *недетерминированно пишет* y (который в силу определения 2.2 существует для любого слова x, для которого $L(x) = 1$). Говорят ещё, что M «отгадывает» (“guesses”) y. Более точно это означает, что M находит правый конец входного слова, сдвигается ещё на одну ячейку вправо, записывает в неё #, ещё раз сдвигается вправо и переходит в такое недетерминированное состояние, в котором она пишет один из символов на ленту, сдвигается вправо и выбирает между сохранением этого состояния и переходом в (уже детерминированное) состояние начала следующего этапа.

После завершения первого этапа на ленте, помимо входного слова x, записано ещё и слово y. Теперь M осталось вычислить значение предиката $R(x, y)$ за полиномиальное время. Это можно сделать, используя детерминированный алгоритм, существующий в силу определения 2.2. □

Ещё одно определение класса NP есть не более чем вариант определения 2.2, но именно в такой форме его удобно обобщать и получать определения других сложностных классов.

Определение 2.3. Имеются два персонажа: король **A**rthur (Артур), умственные способности которого полиномиально ограничены, и волшебник **M**erlin (Мерлин), который интеллектуально всемогущ и знает правильные ответы на все вопросы. Король **A** интересуется некоторым свойством $L(x)$ (например, «есть ли у графа гамильтонов цикл»), а волшебник **M** хочет, чтобы король признал наличие этого свойства (ну, скажем, граф стремится к званию гамильтонова и дал **M** взятку). **A** не доверяет своему волшебнику, зная его корыстолюбие, и хочет иметь возможность самостоятельно проверить предложенный **M** ответ.

Поэтому они действуют следующим образом. **A** и **M** оба смотрят на слово x, после чего **M** сообщает некоторую информацию (слово y), которая должна убедить **A**, что $L(x) = 1$. Используя эту информацию, **A** проверяет убедительность аргументов **M** некоторым полиномиальным способом.

В этих терминах определение класса NP можно сформулировать так: свойство L принадлежит классу NP, если у Артура есть полиномиальный способ проверять убедительность доводов Мерлина, причём:

$L(x) = 1 \quad \Longrightarrow \quad$ у **M** есть способ убедить **A** в этом;

$L(x) = 0 \quad \Longrightarrow \quad$ как бы **M** ни изощрялся, **A** не поверит, что $L(x) = 1$.

Эквивалентность этого определения определению 2.2 не вызывает сомнений. Действительно, из-за полиномиальной ограниченности короля **A**, сообщение волшебника **M** должно иметь полиномиальный размер. В остальном различия между определениями чисто внешние.

2.2. Сводимость и NP-полнота. Сложность вычисления предикатов можно сравнивать, пользуясь следующим определением.

Определение 2.4. Сводимость по Карпу. Предикат L_1 сводится к предикату L_2 (обозначение $L_1 \propto L_2$), если существует такая функция $f \in \mathrm{P}$, что $\forall\, x\ L_1(x) = L_2(f(x))$.

Сводимость по Карпу также называют полиномиальной сводимостью, а часто — просто сводимостью.

Лемма 2.1. *Пусть* $L_1 \propto L_2$. *Тогда*
а) $L_2 \in \mathrm{P} \Rightarrow L_1 \in \mathrm{P}$; б) $L_1 \notin \mathrm{P} \Rightarrow L_2 \notin \mathrm{P}$; в) $L_2 \in \mathrm{NP} \Rightarrow L_1 \in \mathrm{NP}$.

Доказательство. Пункт а) совершенно очевиден — чтобы вычислить значение предиката L_1, достаточно применить ко входу x функцию f, а к результату её работы — программу вычисления L_2.

Пункт б) следует из пункта а).

Пункт в) также прост. Его можно объяснять по-разному. Например, так: Мерлин сообщает Артуру $f(x)$ (длина которого ограничена некоторым полиномом h от длины x, поскольку $f \in \mathrm{P}$) и слово y, которое убеждает Артура в том, что $L_2(f(x)) = 1$. Артур также может проверить, что ему действительно сообщено $f(x)$. Используя определение 2.2, можно написать цепочку эквивалентностей

$$L_1(x) \Leftrightarrow L_2(f(x)) \Leftrightarrow \exists\, y\ \big((|y| < q(|f(x)|)) \wedge R(f(x), y)\big) \Leftrightarrow$$
$$\Leftrightarrow \exists\, y\ \big((|y| < r(|x|)) \wedge R(f(x), y)\big).$$

Полином $r(|x|)$ в последнем выражении — это композиция полиномов q и h. □

Определение 2.5. Предикат $L \in \mathrm{NP}$ называется NP-*полным*, если любой предикат из NP к нему сводится.

Если некоторый NP-полный предикат можно вычислять за время $T(n)$, то любой предикат из NP можно вычислять за время $T(n^c)$ для некоторого фиксированного числа c. Такая потеря эффективности считается в этой науке несущественной.

NP-полные предикаты существуют, приведём примеры.

Выполнимость. Задаётся предикатом

$SAT(x) \quad \Longleftrightarrow \quad$ x есть формула с булевыми переменными и символами ($\neg$, $\vee$, $\wedge$), которая истинна при некоторых значениях переменных.

Теорема 2.2 (Кук, Левин). 1) $SAT \in \mathrm{NP}$; 2) SAT — NP-*полна.*

Следствие. *Если* $SAT \in \mathrm{P}$, *то* $\mathrm{P} = \mathrm{NP}$.

Доказательство (теоремы 2.2). 1) Мерлину достаточно сообщить Артуру значения переменных, входящих в формулу, при которых она истинна. Артур справится с проверкой истинности полученного высказывания.

2) Пусть предикат из NP, который нужно свести к SAT, имеет вид $L(x) = \exists\, y\ \big((|y| < q(|x|)) \wedge R(x, y)\big)$.

Рассмотрим таблицу вычисления (см. доказательство теоремы 1.2) для МТ, вычисляющей $R(x, y)$ (входом является пара $x\#y$). Будем использовать те же переменные, что и в доказательстве теоремы 1.2 (коды состояний клеток таблицы вычисления). Чтобы таблица вычисления соответствовала правильно проведённому успешному (с ответом «да») вычислению, должны выполняться локальные правила согласования для каждой четвёрки клеток вида и результат должен быть «да». Каждое такое правило задаётся формулой от переменных, отвечающих либо рассматриваемой четвёрке, либо нулевой ячейке самой нижней строки таблицы. Определим формулу φ_x как конъюнкцию всех этих формул, в которые подставлены значения переменных, кодирующих вход $x\#y$, дополненный символами $\sqcup$ до длины $|x| + 1 + q(|x|)$. Значения, соответствующие x и $\#$, — константы, поэтому переменные, от которых зависит эта формула, отвечают y и кодам внутренних ячеек таблицы. Так что можно считать, что формула φ_x зависит от y и ещё от каких-то переменных, которые мы обозначим z.

Итак, мы сопоставили слову x формулу $\varphi_x(y, z)$, которая по построению обладает следующим свойством. Если выполняется $R(x, y)$, то найдётся такой набор значений $z(x, y)$, при котором $\varphi_x(y, z(x, y))$ истинна (эти значения описывают работу МТ на входе $x\#y$). А если $R(x, y)$ не выполняется, то $\varphi_x(y, z)$ всегда ложна (поскольку по сути утверждает, что вычисление на входе (x, y) даёт ответ «да»). Таким образом, при $L(x) = 1$ такая формула иногда (при некоторых значениях y) истинна, при $L(x) = 0$ — всегда ложна. □

Другие примеры NP-полных задач получаются с помощью следующей леммы.

Лемма 2.2. *Если $SAT \propto L$ и $L \in$ NP, то L — NP-полная. И вообще, если L_1 — NP-полная, $L_1 \propto L_2$ и $L_2 \in$ NP, то L_2 — NP-полная.*

Доказательство. Достаточно проверить транзитивность сводимости: если $L_1 \propto L_2$, $L_2 \propto L_3$, то $L_1 \propto L_3$. Она следует из того, что композиция двух полиномиально вычислимых функций полиномиально вычислима. □

3-КНФ. Задаётся предикатом

3-$SAT(x)$ $\iff$ x есть 3-КНФ, которая истинна истинна при некоторых значениях переменных. 3-КНФ — это конъюнкция дизъюнкций, каждая из которых содержит три литерала.

3-SAT также NP-полна. Это устанавливается сведением к ней SAT.

Теорема 2.3. *$SAT \propto$ 3-SAT.*

Доказательство. Сопоставим любой схеме в стандартном базисе такую 3-КНФ, выполнимость которой равносильна тому, что вычисляемая схемой функция не равна тождественно 0. В эту КНФ будут

входить все переменные схемы ($x_1, \ldots, x_n$ и $y_1, \ldots, y_s$ — используем те же обозначения, что и в определении схемы). Для этого построим вначале конъюнкцию K'' условий, означающих, что каждая из вспомогательных переменных y_j имеет значение, соответствующее правой части присваивания Y_j. Есть три типа таких условий (они соответстуют трём базисным функциям), и каждый можно записать в виде 3-КНФ:

$$(y \Leftrightarrow (x_1 \vee x_2)) = (x_1 \vee x_2 \vee \neg y) \wedge (\neg x_1 \vee x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee y) \wedge \wedge (\neg x_1 \vee \neg x_2 \vee y),$$

$$(y \Leftrightarrow (x_1 \wedge x_2)) = (x_1 \vee x_2 \vee \neg y) \wedge (\neg x_1 \vee x_2 \vee \neg y) \wedge (x_1 \vee \neg x_2 \vee \neg y) \wedge \wedge (\neg x_1 \vee \neg x_2 \vee y),$$

$$(y \Leftrightarrow \neg x) = (x \vee y) \wedge (\neg x \vee \neg y).$$

Подставляя эти 3-КНФ в K'', получим 3-КНФ K'. Искомая 3-КНФ имеет вид $K = K' \wedge y_s$. Действительно, истинность K равносильна утверждению: все присваивания выполнены правильно и в результате получилась 1 ($y_s = 1$). Значит, если при каких-то значениях переменных x_i схема выдаёт 1, то K выполнима, и наоборот. □

Ещё один простой пример свед*е*ния.

Задача ЦЛП (целочисленного линейного программирования). Дана система линейных неравенств с целыми коэффициентами. Есть ли у неё целочисленное решение? (Другими словами, *совместна* ли система?)

В этой задаче входом является матрица коэффициентов и вектор правых частей. То, что ЦЛП $\in$ NP, не совсем очевидно. Оказывается, что в качестве подсказки Мерлин может сообщить Артуру значения переменных, при которых выполнены все неравенства системы. По определению, длина этого сообщения должна быть полиномиально ограничена. Можно доказать, что из существования целочисленного решения следует существование целочисленного решения, размер записи которого ограничен полиномом от длины записи системы, см. [14, т. 2, §17.1].

Сведём теперь 3-КНФ к ЦЛП. Построим по 3-КНФ систему линейных неравенств. Целочисленных переменных возьмём столько же, сколько есть булевых переменных. Булевой переменной x_i сопоставим выражение p_i. Отрицанию переменной $\neg x_i$ сопоставим выражение $1-p_i$. Каждой дизъюнкции $X_j \vee X_k \vee X_m$ (X_* — литералы) сопоставим неравенство $P_j + P_k + P_m \geqslant 1$, в котором P_j, P_k, P_m — выражения, сопоставленные литералам дизъюнкции.

Искомая система содержит для всех i неравенства $0 \leqslant p_i \leqslant 1$, а также все неравенства, сопоставленные дизъюнкциям из КНФ. Очевидно, что выполнимость 3-КНФ равносильна совместности такой системы.

Замечание 2.4. Если не требовать целочисленности решений, то проверить совместность системы линейных неравенств можно за полиномиальное время. Примеры таких алгоритмов (Хачияна, Кармаркара) также см. в [14, т. 1, §13, §15.1].

Обширный список NP-полных задач содержится в книге Гэри и Джонсона [3]. Как правило, их NP-полнота доказывается с помощью свед́ений. Приведём несколько примеров NP-полных задач.

3-раскраска. Дан граф. Спрашивается, можно ли раскрасить его вершины в три цвета так, чтобы концы каждого ребра были покрашены в разные цвета. (Аналогичная задача 2-раскраска решается за полиномиальное время.)

Клика. Даны граф и число k. Спрашивается, есть ли k-элементное множество вершин графа, любые две вершины которого соединены ребром.

Задачи

2.1. Докажите, что задача проверки выполнимости 2-КНФ (конъюнкции дизъюнкций, каждая из которых содержит два литерала) принадлежит P.

2.2. Докажите, что задача об эйлеровом пути в неориентированном графе (существует ли путь, проходящий по всем рёбрам ровно по одному разу) принадлежит P.

2.3. Предположим, что класс NP совпадает с P. Докажите, что в этом случае за полиномиальное время можно не только проверить выполнимость формулы, но и найти значения переменных, при которых она истинна (аналогично для гамильтонова цикла и т. п.)

2.4. Докажите, что задача о паросочетаниях (есть n мальчиков и n девочек, известно, какие пары согласны друг с другом танцевать; надо определить, можно ли устроить танец, в котором все n мальчиков и n девочек соединены в пары) принадлежит NP и, более того, принадлежит P.

2.5. Постройте

а) полиномиальное сведение задачи 3-КНФ к задаче Клика;

б) тот же вопрос, но дополнительно требуется, чтобы количество решений сохранялось (если 3-КНФ F при сведении соответствует пара (граф H, размер k), то количество выполняющих наборов переменных для F равно количеству клик размера k в графе H).

2.6. Постройте

а) полиномиальное сведение задачи 3-КНФ к задаче 3-раскраска;

б) тот же вопрос, но дополнительно требуется, чтобы количество выполняющих наборов переменных для любой 3-КНФ равнялось бы шестикратному количеству 3-раскрасок соответствующего ей графа.

2.7. Покажите, что следующая задача является NP-полной: дан набор из не более чем n типов квадратиков 1×1, на сторонах которых написаны какие-то буквы; дан список допустимых пар букв и список граничных букв; спрашивается, можно ли корректно сложить из квадратиков набора большой квадрат размера $n \times n$ (так, чтобы на примыкающих сторонах квадратиков были только допустимые пары букв, а на границе квадрата — только граничные буквы).

2.8. Докажите, что предикат «x — двоичная запись составного числа» принадлежит NP.

2.9. Докажите, что предикат «x — двоичная запись простого числа» принадлежит NP.

3. Вероятностные алгоритмы и класс BPP. Проверка простоты числа

В вероятностных МТ (ВМТ), как и в недетерминированных, имеются состояния, из которых возможен переход в несколько (больше одного) состояний. Отличие состоит в том, что состояние, куда ВМТ делает переход, определяется результатом некоторого случайного процесса («подбрасывания монеты»). Нужно оговорить, какие монеты допускаются. Например, если взять монету, вероятность выпадения герба для которой равна невычислимому числу, то возможности ВМТ, использующей подбрасывание такой монеты, будут больше, чем хотелось бы; она сможет вычислить и некоторые неразрешимые предикаты.

Обычно считают, что вероятности выпадения одинаковы для обеих сторон монеты, а результат подбрасывания отождествляется с числом 0 или 1.

Хотя для ВМТ нельзя сказать, какой в точности ответ она выдаст, можно определить вероятность того или иного ответа.

Определение 3.1. Предикат L принадлежит классу BPP, если существуют такие ВМТ M и полином $p(n)$, что машина M заведомо остановится за время, не превосходящее $p(|x|)$, причём

$L(x) = 1 \quad \Longrightarrow \quad M$ с вероятностью большей 2/3 даёт ответ «да»;
$L(x) = 0 \quad \Longrightarrow \quad M$ с вероятностью большей 2/3 даёт ответ «нет».

Если в этом определении заменить число 2/3 на любое фиксированное число, бо́льшее 1/2, класс BPP не изменится. Есть простой способ добиться вероятности, сколь угодно близкой к 1. Возьмём несколько одинаковых машин, запустим их все, а окончательным результатом будем считать мнение большинства. Если вероятность неправильного ответа для каждого экземпляра машины равна $c < 1/2$, то вероятность неправильного ответа после голосования n машин

$$\sum_{\substack{S\subseteq\{1,\ldots,n\},\\ |S|\leqslant n/2}} (1-c)^{|S|}c^{n-|S|} = ((1-c)c)^{n/2} \sum_{\substack{S\subseteq\{1,\ldots,n\},\\ |S|\leqslant n/2}} \left(\frac{c}{1-c}\right)^{n/2-|S|} <$$
$$< \left(\sqrt{(1-c)c}\right)^n 2^n = \lambda^n, \quad \text{где } \lambda = 2\sqrt{c(1-c)} < 1. \tag{3.1}$$

Замечание 3.1. Представить физическую реализацию НМТ очень трудно (вспомним, что нам придётся поместить в неё всезнайку Мерлина). А вероятностные машины вполне могут мыслиться как реальные устройства. Поэтому предикаты из класса BPP вполне можно считать реально вычислимыми.

Класс BPP можно определить и с помощью предикатов от двух переменных, как это было сделано для класса NP.

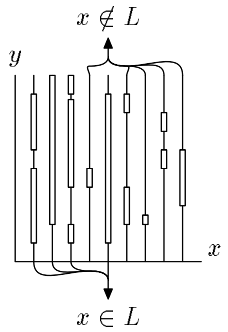
Определение 3.2. Предикат L принадлежит классу BPP, если существуют такие полином $q(\cdot)$ и предикат $R(\cdot,\cdot) \in \mathrm{P}$, что

$L(x) = 1 \quad \Longrightarrow \quad$ доля слов r длины $q(|x|)$, для которых выполнено $R(x,r)$, больше 2/3;
$L(x) = 0 \quad \Longrightarrow \quad$ доля слов r длины $q(|x|)$, для которых выполнено $R(x,r)$, меньше 1/3.

Теорема 3.1. *Определения* 3.1 *и* 3.2 *эквивалентны.*

Доказательство. Опр. 3.1 $\Longrightarrow$ опр. 3.2. Полагаем $q(n) = p(n)$ (количество подбрасываний монеты не превосходит общего числа действий). Определим предикат $R(x,r) =$ «машина M на входе x даёт ответ „да“ при указанной в r последовательности результатов подбрасываний монеты». Этот предикат по очевидным причинам удовлетворяет определению 3.2.

Опр. 3.2 $\Longrightarrow$ опр. 3.1. Случайно выбираем слово r длины $q(|x|)$ и подставляем в предикат, затем вычисляем значение предиката. Такая ВМТ удовлетворяет определению 3.1. □



Определение 3.2 допускает следующее наглядное толкование. Отметим на плоскости (x,y) точки из множества $\{(x,y) : R(x,y) \wedge (|y| = q(|x|))\}$. Для $x = x_0$ рассмотрим сечение $\{(x,y) : R(x,y) \wedge \wedge (|y| = q(|x|)) \wedge (x = x_0)\}$ этого множества. Предикат $R(\cdot,\cdot)$, участвующий в определении 3.2, обладает таким странным свойством, что мера этого сечения при любом x_0 либо больше 2/3, либо меньше 1/3. Это разделяет значения x на две категории: одна соответствует истинности предиката L, другая — ложности.

Классический пример задачи из BPP представляет ПРОВЕРКА ПРОСТОТЫ числа: дано число n, требуется определить, простое ли оно. Для этой задачи существует вероятностный алгоритм, работающий за полиномиальное время; он будет сейчас описан.

3.1. Необходимые сведения из теории чисел. Подробное изложение элементарной теории чисел содержится в книге [2]. Мы лишь напомним две теоремы, которые будут важны при анализе работы алгоритма проверки простоты числа.

Малая теорема Ферма. *Если n — простое и $n \nmid a$, то $a^{n-1} \equiv 1 \pmod{n}$.*

Китайская теорема об остатках. *Пусть $n = uv$ — разложение числа на взаимно простые множители. Тогда $\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}/u\mathbb{Z} \times \mathbb{Z}/v\mathbb{Z}$.*

Другими словами, существует взаимно однозначное соответствие между остатками от деления на n и парами остатков от деления на u и на v. (И это соответствие уважает операции сложения и умножения.)

Из малой теоремы Ферма следует, что $a^{n-1} \not\equiv 1 \pmod{n}$ позволяет утверждать, что n — составное (говорят, что a является *свидетелем* непростоты числа n). Это свидетельство косвенное — явного разложения n на множители мы не получаем — и сильное: часто достаточно проверки при $a = 2$!

Однако проверки малой теоремы Ферма даже при всех a может оказаться недостаточно. Алгоритм проверки будет использовать свидетелей ещё одного типа: если $b^2 \equiv 1 \pmod{n}$, а $b \not\equiv \pm 1 \pmod{n}$, то n — составное; n и $b-1$ имеют общий делитель, бо́льший 1. Поэтому свидетели такого вида (вообще говоря, гораздо более редко появляющиеся) позволяют сразу же указать разложение n (против простоты которого они свидетельствуют) на два множителя за полиномиальное время (наибольший общий делитель (x, y) двух чисел x и y можно найти за полиномиальное время, см. ниже).

3.2. Необходимые сведения из алгоритмической теории чисел. Арифметические операции над числами можно выполнять за полиномиальное время от длины их записи (число n записывается $\lceil \log n \rceil = O(\log n)$ знаками). Например, схема умножения чисел n, m столбиком имеет размер $O(\log n \log m)$, растущий квадратично от длины входа. Хотя квадрат от длины входа уже достаточно мал, чтобы быть полиномиальным, заметим, что для умножения n-значных чисел есть и более экономные схемы размера $O(n \log n \log\log n)$, см. [1, р. 7.5] или [7, т. 2, р. 4.3.3].

В модулярной арифметике (арифметике остатков по модулю заданного числа q) можно вычислить a^n по модулю q схемой полиномиального размера от длины входа (a, n, q) (в обычной арифметике даже результат возведения в степень может иметь экспоненциальный размер). Для этого нужно вычислить остатки от степеней a^{2^k} последовательным возведением в квадрат и затем перемножить те из полученных чисел, которым соответствуют 1 в двоичной записи числа n.

Наибольший общий делитель двух чисел можно найти, пользуясь алгоритмом Евклида, использующим рекурсивно равенство $(x, y) = (y, x \bmod y)$. Если делить бóльшее число на меньшее, то за каждые два шага длина записи меньшего числа уменьшается на константу.

Существует также алгоритм проверки того, что из числа извлекается нацело корень k-й степени. Это можно сделать, найдя достаточно точно приближённое значение корня и возведя ближайшее к нему целое число в k-ю степень. Найти приближённое значение $\sqrt[k]{x}$ можно с помощью рекуррентной последовательности

$$a_{n+1} = \frac{1}{k}\left((k-1)a_n + \frac{x}{a_n^{k-1}}\right),$$

если выбрать подходящее значение a_0. Детали этого (полиномиального) алгоритма оставляются читателю для самостоятельного обдумывания.

Заметим, что все схемы, о которых шла речь выше, могут быть построены МТ за полиномиальное время. Так что все перечисленные задачи принадлежат классу P.

3.3. Алгоритм проверки простоты. Вход: число n.

Шаг 1. Проверяем чётность n. Если $n = 2$, то ответ «n — простое», если n — чётное и больше 2, то ответ «n — составное», в противном случае переходим к шагу 2.

Шаг 2. Проверяем, извлекается ли из n нацело корень k-й степени при $k = 2, \ldots, \lfloor \log_2 n \rfloor$. Если извлекается, то ответ «n — составное», иначе переходим к шагу 3.

Шаг 3. Записываем $n - 1$ в виде $2^k \cdot l$, где $k > 0$, а l — нечётное.

Шаг 4. Выбираем случайное a среди чисел от 1 до n.

Шаг 5. Вычисляем a^l, a^{2l}, $\ldots$, a^{n-1} по модулю n.

Проверка 1. Если $a^{n-1} \not\equiv 1 \pmod{n}$, то ответ «$n$ — составное».

Проверка 2. Если найдено такое j, для которого $a^{2^j l} \not\equiv \pm 1 \pmod{n}$, а $a^{2^{j+1} l} \equiv 1 \pmod{n}$, то ответ «$n$ — составное».

В противном случае ответ «n — простое».

3.4. Анализ алгоритма.

Теорема 3.2. *Если n — простое, то описанный выше алгоритм всегда (с вероятностью 1) выдаёт ответ «n — простое».*

Если n — составное, то ответ «n — составное» будет получен с вероятностью $\geqslant 1/2$.

Замечание 3.2. Чтобы получить полиномиальный вероятностный алгоритм проверки простоты числа в смысле определения 3.1, нужно дважды применить приведённый алгоритм. Тогда вероятность ошибки станет меньше 1/4.

Доказательство (теоремы 3.2). Из сказанного выше следует, что в доказательстве нуждается только второе утверждение.

Пусть $n = uv \equiv 1 \pmod{2}$, где $(u, v) = 1$ (если такого представления нет, то на шаге 2 будет обнаружена непростота). Если $(a, n) > 1$, то проверка 1 для такого a заведомо обнаружит, что n — составное. Так что достаточно доказать, что не меньше половины тех a, которые взаимно просты с n, обнаруживают непростоту числа.

Обозначим группу $(\mathbb{Z}/u\mathbb{Z})^*$ через U, группу $(\mathbb{Z}/v\mathbb{Z})^*$ — через V. Из китайской теоремы об остатках следует, что группа $(\mathbb{Z}/n\mathbb{Z})^*$ изоморфна $U \times V$ (изоморфизм задаётся естественным образом — вычислением остатков по модулю u и по модулю v).

Рассмотрим множества $U^k = \{x^k : x \in U\}$, $V^k = \{x^k : x \in V\}$. Это подгруппы в U и V соответственно (произведение k-х степеней есть k-я степень, обратный к k-й степени есть k-я степень). Так что $|U|/|U^k|$ — целое число. Более того, отображение $x \mapsto x^k$ является гомоморфизмом групп, поэтому число прообразов при этом отображении одинаково для всех элементов U^k.

Очевидно, что $U^{2l} \subseteq U^l$ (степеней квадратов не больше, чем всех степеней). Поэтому получаем пару невозрастающих цепочек множеств

$$U^l \supseteq U^{2l} \supseteq \ldots \supseteq U^{n-1} \supseteq \{1\}, \qquad V^l \supseteq V^{2l} \supseteq \ldots \supseteq V^{n-1} \supseteq \{1\}.$$

Дальнейшее рассуждение разбивается на анализ нескольких случаев.

1. Пусть $U^{n-1} \neq \{1\}$ или $V^{n-1} \neq \{1\}$.
   Чтобы пройти проверку 1 в алгоритме, нужно после возведения в $(n-1)$-ю степень получить пару остатков $(1,1)$. Поскольку прообразов каждого числа из $U^{n-1} \neq \{1\}$ одинаковое количество, вероятность получения пары $(1,1)$ не выше $1/2$.
2. Пусть $U^{n-1} = V^{n-1} = \{1\}$.
   Поскольку l — нечётное, $-1 \in U^l \cap V^l$, т.е. найдётся такое $t = 2^s$, $0 \leqslant s < k$, что $U^{2t} = V^{2t} = \{1\}$, а $U^t \neq \{1\}$ или $V^t \neq \{1\}$.
   Будем доказывать, что с вероятностью не меньше $1/2$ в этом месте проверка 2 алгоритма обнаружит, что n — составное. В данном случае $a^{2t} \equiv 1 \pmod n$, так что нужно понять, с какой вероятностью a^t не равно ± 1. Рассмотрим два случая.
   а) Одно из множеств U^t, V^t равно $\{1\}$. Пусть, например, $U^t = \{1\}$. В этом случае можно утверждать, что $a^t \not\equiv -1 \pmod n$ (остатку -1 при делении на n соответствует пара остатков $(-1,-1)$ при делении на u, v).
   Можно рассуждать так же, как и в случае 1. С вероятностью не меньше $1/2$ получим пару остатков $(1, \alpha)$, $\alpha \neq 1$.
   б) Оба множества U^t, V^t содержат по крайней мере два элемента, пусть $|U^t| = c$, $|V^t| = d$. В этом случае может получиться как $a^t \equiv 1 \pmod n$, так и $a^t \equiv -1 \pmod n$, но вероятность этого не превосходит $2/(cd) \leqslant 1/2$. □

Замечание 3.3. Шаг 2 в алгоритме избыточен, дополнительными соображениями устанавливается, что и для чисел вида m^k проверки 1, 2 дают правильный ответ со значительной вероятностью.

3.5. BPP и схемная сложность.

Теорема 3.3. $\mathrm{BPP} \subseteq \mathrm{P/poly}$.

Доказательство. Идея доказательства состоит в том, чтобы усилить оценки вероятностей с $(1/3, 2/3)$ до $(\varepsilon, 1-\varepsilon)$. Число ε должно быть настолько мало, чтобы можно было выбрать такое случайное слово y_0, при котором рассматриваемый предикат из BPP совпадает с $R(x, y_0)$.

Суть здесь в том, что повторение опытов за полиномиальное время экспоненциально уменьшает оценку вероятности ошибки ε, но не меняет размер входа $|x|$. Поэтому можно добиться выполнения неравенства $\varepsilon 2^{|x|} < 1$. При таком соотношении между ε и $|x|$ всегда найдётся случайное слово, для которого нет ошибок ни при каких x.

Действительно, вспомним наглядное толкование BPP, данное после теоремы 3.1. Если доля слов r, на которых происходит ошибка, для каждого x не превосходит ε, то доля тех слов r, на которых происходит ошибка хотя бы для одного x, не превосходит $\varepsilon 2^{|x|} < 1$. Значит, найдутся и такие r, на которых нет ошибок ни при каком x. □

Это типичное неконструктивное доказательство существования. Мы доказываем, что вероятность того, что объекта с нужными свойствами не существует, меньше 1. Но это и означает, что хотя бы один такой объект существует.

4. Иерархия сложностных классов

Напомним, что мы отождествляем языки и предикаты, как описано на с. 20. В частности, запись $x \in L$ означает $L(x) = 1$.

Определение 4.1. Пусть A — некоторый класс языков. Класс дополнений co-A составляют дополнения ко всем языкам из A. Формально

$$L \in \mathrm{A} \iff (\mathbb{B}^* \setminus L) \in \text{co-A}.$$

Непосредственно из определений классов P, BPP, PSPACE следует, что P = co-P, BPP = co-BPP, PSPACE = co-PSPACE.

4.1. Игры, в которые играют машины. Рассмотрим игру, в которую играют два игрока, будем их называть белые (Б) и чёрные (Ч). Игрокам сообщается некоторое слово x, и они делают ходы по очереди (w_1 — первый ход белых, b_1 — первый ход чёрных и т. д.). Каждый ход может быть описан словом длины $p(|x|)$, где $p(\cdot)$ — некоторый полином. Игра завершается после некоторого, заранее заданного, числа ходов.[8] Результат игры описывается некоторым предикатом $W(x, w_1, b_1, \ldots) \in \mathrm{P}$, истинность которого означает, что выиграли белые (ничьих не бывает, так что ложность W означает, что выиграли чёрные). Предикат W зависит от исходного слова и ходов, сделанных игроками: $w_1, \ldots$ — белыми, $b_1, \ldots$ — чёрными. Поскольку P замкнут относительно дополнений, предикат $B(\cdot) = \neg W(\cdot)$, утверждающий выигрыш чёрных, также принадлежит P.

Отсутствие ничьих и конечность числа ходов гарантируют при заданном x существование выигрышной стратегии либо для белых, либо

[8] На самом деле правило завершения игры может быть и сложнее. Если есть верхняя оценка на число ходов, то можно всегда дополнить число ходов до этой оценки, никак не учитывая дополнительные ходы при подведении результата игры.

для чёрных. (Формальное доказательство легко получается индукцией по числу ходов.) Поэтому каждой игре можно сопоставить два взаимно дополнительных множества

$$L_w = \{x : \text{Б имеет выигрышную стратегию}\},$$
$$L_b = \{x : \text{Ч имеет выигрышную стратегию}\}.$$

Многие сложностные классы можно определить как множества L_w (или L_b), соответствующие тем или иным видам игр. Например, получаем следующие классы.

P: множества L_w (как и L_b, впрочем) для игр, в которых никто не делает ходов.

NP: множества L_w для игр, в которых белые делают 1 ход. Другими словами, это множества вида

$$\{x : \exists\, w_1\; W(x, w_1)\}.$$

co-NP: множества L_b для игр, в которых белые делают 1 ход. Другими словами, это множества вида

$$\{x : \forall\, w_1\; B(x, w_1)\}.$$

Σ_2: множества L_w для игр из 2 ходов: 1 ход белых, 1 ход чёрных. Другими словами, это множества вида

$$\{x : \exists\, w_1\, \forall\, b_1\; W(x, w_1, b_1)\}.$$

Словами это можно сказать так: есть такой ход белых, что, как бы ни сыграли чёрные, белые выигрывают.

Π_2: множества L_b для игр из 2 ходов. Другими словами, это множества вида

$$\{x : \forall\, w_1\, \exists\, b_1\; B(x, w_1, b_1)\}$$

...

Σ_k: множества L_w для игр из k ходов (в зависимости от чётности k последними ходят либо чёрные, либо белые). Другими словами, это множества вида

$$\{x : \exists\, w_1\, \forall\, b_1\; \ldots\; \mathsf{Q}_k\, y_k\; W(x, w_1, b_1, \ldots)\}$$

(если k чётное, то $\mathsf{Q}_k = \forall$, $y_k = b_{k/2}$, если k нечётное, то $\mathsf{Q}_k = \exists$, $y_k = w_{(k+1)/2}$).

Π_k: множества L_b для игр из k ходов (в зависимости от чётности k последними ходят либо чёрные, либо белые). Другими словами, это множества вида

$$\{x : \forall\, w_1\, \exists\, b_1\; \ldots\; \mathsf{Q}_k\, y_k\; B(x, w_1, b_1, \ldots)\}$$

(если k чётное, то $\mathsf{Q}_k = \exists$, $y_k = b_{k/2}$, если k нечётное, то $\mathsf{Q}_k = \forall$, $y_k = w_{(k+1)/2}$).

...

Классы Σ_k и Π_k взаимно дополнительны: $\Sigma_k = \text{co-}\Pi_k$, $\Pi_k = \text{co-}\Sigma_k$.

Теорема 4.1 (Лаутеман [35]). BPP $\subseteq \Sigma_2 \cap \Pi_2$.

Доказательство. Поскольку класс BPP замкнут относительно дополнений, достаточно показать, что BPP $\subseteq \Sigma_2$.

Для этого нужно научиться формулировать свойство «множество содержит много элементов» с использованием кванторов существования и всеобщности. Мы сделаем это, предполагая, что рассматриваются подмножества некоторой конечной группы. Пусть G — группа, а X — подмножество G. Свойство, которым мы будем отличать большие множества от малых, состоит в том, что некоторым количеством сдвигов множества X можно покрыть всю группу

$$\exists\, g_1, \ldots, g_k : \bigcup_i g_i X = G. \tag{4.1}$$

Чтобы выбрать подходящее значение k, нужно найти случаи, когда (4.1) заведомо выполняется, и когда заведомо не выполняется.

Если

$$k|X| < |G|, \tag{4.2}$$

условие (4.1) заведомо ложно.

Условие (4.1) заведомо истинно, если для случайных независимых $g_1, \ldots, g_k$ вероятность события $\bigcup_i g_i X = G$ больше 0. Другими словами, $\mathbf{Pr}[G \setminus \bigl(\bigcup_i g_i X\bigr) \neq \varnothing] < 1$. Вероятность того, что случайный сдвиг X не покрывает (не содержит) некоторый фиксированный элемент, равна по очевидным причинам $1 - |X|/|G|$. Вероятность того, что k случайных сдвигов не покрывают фиксированный элемент, равна $(1 - |X|/|G|)^k$ (покрытия разными сдвигами — независимые события). Поскольку покрывается $|G|$ элементов, вероятность события $\mathbf{Pr}[G \setminus \bigl(\bigcup_i g_i X\bigr) \neq \varnothing]$ не больше $|G|\,(1 - |X|/|G|)^k$ (вероятность объединения событий не больше суммы вероятностей этих событий). Итак, при

$$|G| \left(1 - \frac{|X|}{|G|}\right)^k < 1 \tag{4.3}$$

условие (4.1) заведомо выполняется.

Рассмотрим теперь некоторый язык $L \in$ BPP. Для него, как объяснялось выше, можно найти полиномиально вычислимый предикат $R(x, y)$ и полином $p(\cdot)$ такие, что число $|X_x|/2^{p(|x|)}$, где $X_x = \{y : (|y| = p(|x|)) \wedge R(x, y)\}$, различает слова, принадлежащие языку (для них оно больше $1 - \varepsilon$), и слова, языку не принадлежащие (для таких слов это число меньше ε). Параметр ε мы выберем позже, сейчас отметим, что его величина может быть экспоненциально мала, как объяснялось на с. 38 после определения 3.1.

Введём искусственно структуру группы на множестве слов y длины $p(|x|)$ так, чтобы произведение и обращение элемента в этой группе были полиномиально вычислимыми (например, в качестве групповой операции возьмём покомпонентное сложение по модулю два). Запишем следующее Σ_2-условие

$$\exists\, g_1, \ldots, g_k\, \forall\, y\, \big((|y| = p(|x|)) \Rightarrow \\ \Rightarrow ((y \in g_1 X_x) \vee (y \in g_2 X_x) \vee \cdots \vee (y \in g_k X_x))\big).$$

Другими словами, рассматриваем такую игру: белые своим ходом называют k слов (элементов группы), а чёрные — один элемент, который, по их мнению, не покрыт сдвигами множества X_x на слова g_i, названные белыми.

Условие (4.2) в этом случае имеет вид $\varepsilon k < 1$, а условие (4.3): $2^{p(|x|)}\varepsilon^k < 1$. Эти условия выполняются при подходящем выборе параметров. Можно взять k порядка $p(|x|)$ и ε порядка $p^{-1}(|x|)$. □

Замечание 4.1. Имеется рассуждение, которое «показывает», что время вычисления функций из класса BPP можно сделать, по-видимому, меньше 2^{n^ε} для любого ε. Идея состоит в том, чтобы использовать генераторы *псевдослучайных* чисел. Такой генератор по набору битов длины k строит набор длины k', где $k' \gg k$. Если при этом выбирать короткие наборы случайно, то длинные наборы будут распределены так, что вычислительное устройство с ограниченными ресурсами (например, полиномиальная машина Тьюринга) не сможет отличить их от по-настоящему случайных. Это пояснение заменяет точное определение псевдослучайного генератора, которое нам не понадобится.

Существуют ли псевдослучайные генераторы, неизвестно. Их заведомо нет, если $\mathrm{P} = \mathrm{NP}$. Но в этом случае $\mathrm{P} = \Sigma_2 = \mathrm{BPP}$.

Если же $\mathrm{P} \neq \mathrm{NP}$ и есть псевдослучайные генераторы, то можно находить за указанное выше время значение функции $f \in \mathrm{BPP}$, вычисляя значения предиката $R(x, y)$ из опр. 3.2 только для псевдослучайных y.

Остаётся «небольшой» дефект в этом рассуждении: оно не проходит при $\mathrm{P} \neq \mathrm{NP}$ и отсутствии псевдослучайных генераторов. Такая ситуация считается маловероятной.

4.2. Класс PSPACE.

Как уже говорилось, в этот класс попадают те функции, которые могут быть вычислены на МТ, использующей память, ограниченную полиномом от длины входного слова. Класс PSPACE также можно описать, используя игры.

Теорема 4.2. $L \in \mathrm{PSPACE}$ *тогда и только тогда, когда существует такая игра с полиномиальным от длины входного слова числом ходов и полиномиально вычислимым результатом, что* $L = \{x \mid$ *Б имеет выигрышную стратегию*$\}$.

Доказательство. $\Longleftarrow$ Покажем, что язык, определяемый игрой, принадлежит PSPACE. Пусть число ходов ограничено $p(|x|)$. Определим по индукции набор машин Тьюринга M_k для $k = 0, \ldots, p(|x|)$. Каждая M_k по заданному началу игры $x, w_1, b_1, \ldots$ длины k определяет наличие выигрышной стратегии у Б. Последней в этом ряду машине $M_{p(|x|)}$ нужно просто вычислить предикат $W(x, w_1, \ldots)$. Машина M_k перебирает все возможные варианты $(k+1)$-го хода и консультируется с M_{k+1} по поводу окончательных результатов игры. Её оценка игры составляется очень просто: если текущий ход у Б, то достаточно найти один ход, при котором M_{k+1} гарантирует выигрышную стратегию для Б. Если текущий ход у Ч, то при всех вариантах M_{k+1} должна обнаружить выигрышную стратегию для Б. Машина M_0 определяет наличие выигрышной стратегии для Б в самом начале игры и для её работы нам нужно использовать всю последовательность машин M_k. Но каждая из этих машин использует небольшую (полиномиально ограниченную) память, так что весь процесс потребует лишь полиномиально ограниченной памяти.

$\Longrightarrow$ Пусть есть машина M, распознающая вхождение слова в язык L на полиномиальной памяти. Во-первых, заметим, что вычисление на памяти S бессмысленно проводить дольше, чем время $2^{O(S)}$ (всё начнет повторяться после того, как мы исчерпаем все состояния нашей системы, а их не более чем $|\mathcal{A}|^S \cdot |\mathcal{Q}| \cdot S$, где $\mathcal{Q}, \mathcal{A}$ — соответственно множество состояний управляющего устройства и алфавит рассматриваемой МТ). Поэтому можно считать без ограничения общности, что время работы машины M ограничено 2^q, где $q = O(p(|x|))$.

Чтобы было проще описывать игру, потребуем, чтобы после завершения вычисления МТ сохраняла без изменений достигнутое состояние.

Игра заключается в следующем. Белые утверждают, что ответ на входном слове x равен «да», а чёрные хотят это проверить. Белые своим первым ходом декларируют состояние машины M (строка, записанная на ленте, положение читающей головки, состояние управляющего устройства) после 2^{q-1} тактов. Чёрные своим ходом выбирают один из промежутков: от начала до (2^{q-1})-го такта или от (2^{q-1})-го такта до конца. Белые декларируют состояние M в середине этого промежутка. Далее всё повторяется: Ч выбирает одну из половинок, Б декларирует состояние M в середине выбранной половинки и т. д.

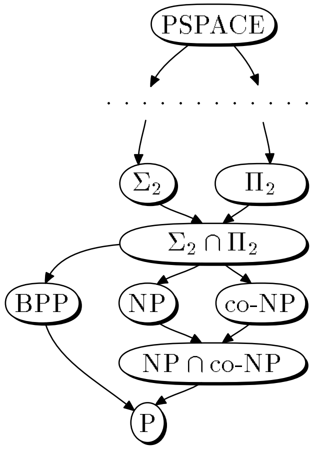
Игра заканчивается, когда длина промежутка становится равной 1. Результат определяется так: для обоих концов этого промежутка в процессе игры были названы состояния M. Если состояние правого конца получается из состояния левого конца за один такт работы M, то Б выиграли, иначе выиграли Ч.

Если ответ M на слове x действительно «да», то белым нужно всё время говорить правду, это гарантирует им выигрыш.

Если ответ M на слове x — «нет», то при любом ходе белых на одном из промежутков (или на обоих) будет содержаться ошибка. Ч должен указывать каждый раз именно этот промежуток. □

Задача 4.1. Докажите, что класс языков, распознаваемых недетерминированными машинами, работающими на памяти S, содержится в классе языков, распознаваемых детерминированными машинами, работающими на памяти $\mathrm{poly}(S)$.

В качестве следствия теоремы 4.2 получаем включения всех определённых выше классов Σ_k, Π_k в класс PSPACE. Взаимное соотношение этих классов можно изобразить диаграммой включений, показанной слева. На этой диаграмме от бóльшего класса к меньшему можно пройти, двигаясь по стрелкам. Внизу располагается класс P, отвечающий играм с 0 ходов, затем идут дополняющие друг друга классы, отвечающие играм с конечным числом ходов (для одного хода это NP и co-NP, для двух ходов — Σ_2 и Π_2 и т. д.). Завершается эта диаграмма классом PSPACE, который определяется произвольными играми с одним естественным условием — время игры должно быть полиномиально ограничено размером входного слова. Мы уже доказали все включения, изображённые на этой диаграмме. Ни про одно из включений, следующих из этой диаграммы, неизвестно, является ли оно строгим. Быть может, скажем, P = PSPACE. С другой стороны, возможно и так, что PSPACE = EXPTIME, где EXPTIME обозначает (не рассматривавшийся нами) класс языков, вычислимых за экспоненциальное время $2^{\mathrm{poly}(n)}$. Впрочем, наиболее популярна гипотеза о том, что все включения, изображённые на диаграмме — строгие.



Задача 4.2. *Машина Тьюринга с оракулом A* — это МТ с дополнительной *оракульной* лентой, куда она (машина) может записывать слова, а затем за один такт работы проверять, принадлежит ли записанное

на оракульной ленте слово языку A. По двум сложностным классам $\mathcal{X}$ и $\mathcal{Y}$ можно определить класс $\mathcal{X}^{\mathcal{Y}}$ таких языков, которые распознаются машинами из класса $\mathcal{X}$ с оракулами из $\mathcal{Y}$.

Докажите, что $\mathrm{P}^{\Sigma_k} = \mathrm{P}^{\Pi_k} \subseteq \Sigma_{k+1} \cap \Pi_{k+1}$.

В классе PSPACE существуют полные задачи (относительно полиномиальной сводимости). Простейший вариант получается применением предыдущей теоремы.

Задача TQBF. Задаётся предикатом

$TQBF(x) \Leftrightarrow$ x есть истинная булева формула с кванторами (True Quantified Boolean Formula), т. е. формула вида

$$\mathsf{Q}_1\, y_1 \ldots \mathsf{Q}_n\, y_n F(y_1, \ldots, y_n),$$

где $y_i \in \mathbb{B}$, F — некоторая логическая формула, а Q_i — либо $\forall$, либо $\exists$. По определению, $(\forall\, y_1\, A(y_1)) = (A(0) \wedge A(1))$, а $(\exists\, y_1\, A(y_1)) = (A(0) \vee A(1))$.

Теорема 4.3. TQBF PSPACE-*полна.*

Доказательство. Построим сведéние любого языка $L \in$ PSPACE к задаче TQBF. Для этого превратим МТ, вычисляющую результат игры (предикат $W(\cdot)$), в схему, а ходы игроков закодируем булевыми переменными. Тогда наличие выигрышной стратегии у белых задаётся условием

$$\exists\, w_1^1\, \exists\, w_1^2 \ldots \exists\, w_1^{p(|x|)}\, \forall\, b_1^1 \ldots \forall\, b_1^{p(|x|)}\, S(x, w_1^1, w_1^2, \ldots),$$

где $S(\cdot)$ обозначает результат вычисления по схеме.

Чтобы превратить S в булеву формулу, добавим новые переменные y_i (значение, вычисленное при i-м присваивании в схеме) и заменим $S(\cdot)$ на формулу вида

$$\exists\, y_1, \ldots, \exists\, y_{\text{размер схемы}}\ (y_1 \Leftrightarrow R_1) \wedge \cdots \wedge (y_s \Leftrightarrow R_s) \wedge y_s,$$

где s — размер схемы, R_i — правая часть i-го присваивания.

После этой подстановки получим квантифицированную булеву формулу, которая истинна в точности для $x \in L$. □

Часть II

Квантовые вычисления

Как уже говорилось во введении, обычные компьютеры не используют всех возможностей, предоставляемых природой. В них выполняются преобразования на конечных множествах состояний (действия с 0 и 1), а в природе есть возможность делать унитарные преобразования, т. е. действовать на бесконечном множестве.[9] Эта возможность описывается квантовой механикой. Устройства (реальные или воображаемые), использующие эту возможность, называются квантовыми компьютерами.

Заранее неясно, увеличиваются ли вычислительные возможности при переходе от преобразований конечных множеств к унитарным преобразованиям конечномерных пространств. Сейчас есть основания полагать, что такое увеличение действительно происходит. В качестве примера можно привести задачу о разложении числа на множители: для обычных компьютеров неизвестны полиномиальные алгоритмы её решения, а для квантовых компьютеров такие алгоритмы есть.

Обычный компьютер работает с состояниями из конечного числа битов. Каждый бит может находиться в одном из двух состояний 0 или 1. Состояние всей системы задаётся указанием значений всех битов. Поэтому множество состояний $\mathbb{B}^n = \{0,1\}^n$ конечно и имеет мощность 2^n.

Квантовый компьютер работает с конечными наборами элементарных состояний, называемых q-битами. Каждый q-бит имеет два выделенных состояния (если считать q-биты спинами, то это состояния «спин вверх» и «спин вниз»). Указание выделенных состояний для каждого q-бита системы задаёт не все возможные состояния системы, а только базисные. Возможны также любые линейные комбинации базисных состояний с комплексными коэффициентами. Базисные состояния

[9] Конечно, настоящей бесконечности в природе не бывает. В данном случае дело в том, что унитарное преобразование можно задать лишь с некоторой точностью — подробности см. в разделе 7.

мы будем обозначать $|x_1, \dots, x_n\rangle$, где $x_j \in \mathbb{B}$, или $|x\rangle$, где $x \in \mathbb{B}^n$. Произвольное состояние системы может быть представлено в виде[10]

$$|\psi\rangle = \sum_{(x_1,\dots,x_n)\in\mathbb{B}^n} c_{x_1,\dots,x_n}|x_1,\dots,x_n\rangle, \text{ где } \sum_{(x_1,\dots,x_n)\in\mathbb{B}^n} |c_{x_1,\dots,x_n}|^2 = 1.$$

Пространство состояний для такой системы — конечномерное (размерности 2^n) пространство над полем комплексных чисел.

Состояние

обычного компьютера	**квантового компьютера**
□ □ ··· □ биты	□ □ ··· □ q-биты
$x_1\ x_2\ \dots\ x_n$ $x_j \in \mathbb{B}$	базисное: $|x_1, x_2, \dots, x_n\rangle$, $x_j \in \mathbb{B}$
	произвольное: $\sum_{x\in\mathbb{B}^n} c_x|x\rangle$, где $\sum_{x\in\mathbb{B}^n} |c_x|^2 = 1$

Небольшое уточнение: если умножить вектор $\sum_x c_x|x\rangle$ на *фазовый множитель* $e^{i\varphi}$ (φ — вещественное), то получится физически неотличимое состояние. Таким образом, состояние квантового компьютера — это вектор единичной длины, заданный с точностью до фазового множителя.

Вычисление можно представлять как последовательность преобразований на множестве состояний системы. Опишем, какие преобразования возможны в классическом, а какие — в квантовом случае.

Классический случай:	**Квантовый случай:**
преобразования — это функции из $\mathbb{B}^n$ в $\mathbb{B}^n$.	преобразования — это унитарные операторы, то есть операторы, сохраняющие длину вектора $\sum_{x\in\mathbb{B}^n} |c_x|^2$.

Замечание. Всё сказанное относится только к замкнутым системам. Реальный квантовый компьютер — это часть большой системы (Вселенной), взаимодействующая с остальным миром. Квантовые состояния и преобразования открытых систем будут рассмотрены в разделах 9–10.

Теперь нужно дать формальное определение квантового вычисления. Как и в классическом случае, можно определить квантовые машины Тьюринга или квантовые схемы. Мы выбираем второй подход, который удобнее по ряду причин.

[10] Скобки $|\dots\rangle$ в записи $|\psi\rangle$ не обозначают никакой операции над объектом ψ — они просто указывают на то, что ψ является вектором.

5. Определения и обозначения

Пространство состояний системы из n q-битов $\mathbb{C}^{2^n}$ можно записать в виде тензорного произведения $\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2 = \left(\mathbb{C}^2\right)^{\otimes n}$. Сомножители соответствуют *пространству состояний одного q-бита*.

Тензорное произведение двух пространств L и M, в которых фиксированы базисы $\{e_1, \ldots, e_l\}$ и $\{f_1, \ldots, f_m\}$, можно определить как пространство с базисом из элементов $e_j \otimes f_k$. (В данном случае $e_j \otimes f_k$ — это то же самое, что (e_j, f_k), т. е. просто пара векторов.) Размерность тензорного произведения равна lm (произведению размерностей сомножителей).

Такое определение *неинвариантно*, т. е. зависит от выбора базисов в перемножаемых пространствах. Можно дать *инвариантное* определение. Для этого рассмотрим вначале пространство (бесконечномерное) с базисом $e \otimes f$, где $e \in L$, $f \in M$ — произвольные векторы из перемножаемых пространств. Тензорное произведение будет факторпространством этого пространства по подпространству, порожденному векторами вида

$$\begin{gathered}(e_1 + e_2) \otimes f - e_1 \otimes f - e_2 \otimes f,\\ e \otimes (f_1 + f_2) - e \otimes f_1 - e \otimes f_2,\\ (\lambda e) \otimes f - e \otimes (\lambda f),\\ \lambda(e \otimes f) - (\lambda e) \otimes f.\end{gathered}$$

Другими словами, указанные векторы считаются равными 0.

Можно доказать, что данные определения эквивалентны.

В нашем случае имеется естественный выделенный базис (соответствующий выделенным состояниям): для $\mathbb{C}^2$ — $\{|0\rangle, |1\rangle\}$, а для $\left(\mathbb{C}^2\right)^{\otimes n}$ — $\{|x_1, \ldots, x_n\rangle\}$, $x_j \in \mathbb{B}$. Пространство $\mathbb{C}^2$ с выделенным базисом обозначается через $\mathcal{B}$. Выделенный базис считается ортонормированным, это задаёт скалярное произведение на пространстве состояний. Коэффициенты $c_{x_1,\ldots,x_n}$ разложения вектора $|\psi\rangle$ по этому базису называются *амплитудами*. Их физический смысл состоит в том, что квадрат модуля амплитуды $|c_{x_1,\ldots,x_n}|^2$ интерпретируется как вероятность обнаружить систему в данном базисном состоянии. Как и должно быть, суммарная вероятность всех состояний равна 1, поскольку длина вектора предполагается единичной. (Вероятности будут подробно обсуждаться позже; до некоторых пор мы будем заниматься линейной алгеброй — изучать унитарные операторы на пространстве $\mathcal{B}^{\otimes n}$).

Мы будем использовать (и уже использовали) принятые в физике обозначения, относящиеся к векторам и скалярному произведению в гильбертовом пространстве (их ввёл Дирак). Векторы обозначаются

$|\xi\rangle$, скалярное произведение — $\langle\xi|\eta\rangle$. Если $|\xi\rangle = \sum_x a_x|x\rangle$ и $|\eta\rangle = \sum_x b_x|x\rangle$, то $\langle\xi|\eta\rangle = \sum_x a_x^* b_x$. (Здесь и далее a^* обозначает комплексное сопряжение.) В записи векторов скобки нужны лишь «для красоты» — они указывают на тип объекта и придают симметрию обозначениям (см. ниже). Вместо $|\xi\rangle$ можно было бы написать просто ξ, хотя это и не принято. Поэтому $|\xi_1+\xi_2\rangle = |\xi_1\rangle + |\xi_2\rangle$ — и то, и другое обозначает вектор $\xi_1 + \xi_2$.

Скалярное произведение антилинейно по *первому* аргументу[11] и линейно по второму, т. е.

$$\begin{aligned}&\langle\xi_1 + \xi_2|\eta\rangle = \langle\xi_1|\eta\rangle + \langle\xi_2|\eta\rangle, &\qquad &\langle\xi|\eta_1 + \eta_2\rangle = \langle\xi|\eta_1\rangle + \langle\xi|\eta_2\rangle,\\ &\langle c\xi|\eta\rangle = c^*\langle\xi|\eta\rangle, &&\langle\xi|c\eta\rangle = c\langle\xi|\eta\rangle.\end{aligned}$$

Если в обозначении скалярного произведения взять левую половину, то получим *бра-вектор* $\langle\xi|$, т. е. линейный функционал на *кет-векторах* (векторах нашего пространства). Бра- и кет-векторы находятся во взаимно однозначном соответствии. (Тем не менее, нужно их как-то различать — именно для этого и были введены угловые скобки.) Из-за антилинейности скалярного произведения по первому аргументу имеем равенство $\langle c\xi| = c^*\langle\xi|$. Бра-вектор можно записать в виде строки, а кет-вектор — в виде столбца (чтобы его можно было умножить слева на матрицу):

$$\langle\xi| = c_0^*\langle 0| + c_1^*\langle 1| = (c_0^*, c_1^*), \qquad |\xi\rangle = c_0|0\rangle + c_1|1\rangle = \begin{pmatrix} c_0\\ c_1\end{pmatrix}.$$

Запись $\langle\xi|A|\eta\rangle$ (A — линейный оператор) можно толковать двояко: либо как скалярное произведение вектора $\langle\xi|$ на вектор $A|\eta\rangle$, либо как — $\langle\xi|A$ на $|\eta\rangle$. Так появляется сопряженный оператор $A^\dagger$: по определению, $\langle A^\dagger\xi|$ (бра-вектор, соответствующий $A^\dagger|\xi\rangle$) равен линейному функционалу $\langle\xi|A$. Из определения сразу следует, что

$$\langle A^\dagger\xi|\eta\rangle = \langle\xi|A|\eta\rangle.$$

Унитарный оператор — это линейный оператор, сохраняющий скалярное произведение. Условие

$$\langle\eta|\xi\rangle = \langle U\eta|U|\xi\rangle = \langle\eta|U^\dagger U|\xi\rangle$$

эквивалентно тому, что $U^\dagger U = I$ (где I — тождественный оператор).

Наше определение скалярного произведения в $\mathcal{B}^{\otimes n}$ согласовано с тензорным произведением:

$$\Big(\langle\xi_1| \otimes \langle\xi_2|\Big)\Big(|\eta_1\rangle \otimes |\eta_2\rangle\Big) = \langle\xi_1|\eta_1\rangle\,\langle\xi_2|\eta_2\rangle\,.$$

[11] Обратите внимание, что математики обычно считают, что скалярное произведение в унитарном пространстве антилинейно по *второму* аргументу.

В дальнейшем будет использоваться *тензорное произведение операторов*. Оно действует в тензорном произведении пространств, на которых действуют сомножители, по правилу

$$(A \otimes B)|\xi\rangle \otimes |\eta\rangle = A|\xi\rangle \otimes B|\eta\rangle.$$

Если операторы заданы в матричном виде в некотором базисе, т. е.

$$A = \sum_{j,k} a_{jk} |j\rangle\langle k|, \qquad B = \sum_{j,k} b_{jk} |j\rangle\langle k|$$

(легко понять, что $|j\rangle\langle k|$ — линейный оператор: $|j\rangle\langle k|\,|\xi\rangle = \langle k|\xi\rangle |j\rangle$), то матричные элементы оператора $C = A \otimes B$ имеют вид $c_{(jk)(lm)} = a_{jl} b_{km}$.

Вычисление состоит из преобразований, считаемых элементарными (выполняемых за единицу времени).

Элементарное преобразование **в классическом случае**: такая функция из $\mathbb{B}^n$ в $\mathbb{B}^n$, которая зависит от небольшого (не зависящего от n) числа битов и изменяет также небольшое число битов.	Элементарное преобразование **в квантовом случае**: тензорное произведение произвольного унитарного оператора, действующего на части сомножителей $\mathcal{B}^{\otimes r}$, где r мало ($r = O(1)$), и тождественного оператора, действующего на остальных сомножителях.

Тензорное произведение некоторого оператора U, действующего на множестве q-битов A, и тождественного оператора, действующего на остальных q-битах, будем обозначать $U[A]$. (В частности, $U[1, \ldots, r] = U \otimes I$ обозначает действие на первых r q-битах.)

Пример 5.1. Приведём матрицу оператора $H[2]$, действующего в пространстве $\mathcal{B}^{\otimes 3}$. Оператор $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ действует на второй q-бит, на остальных q-битах действие тождественное. Базисные векторы расположены в лексикографическом порядке: от $|000\rangle$ до $|111\rangle$.

$$H[2] = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}.$$

С этого места начинается вычислительная сложность. Пусть $r = 2$, тогда U — некоторая матрица 4×4, $U \otimes I$ — матрица размерности $2^n \times 2^n$, у которой по диагонали стоят блоки из матриц U. Эта матрица представляет один элементарный шаг. Когда применяется несколько таких операторов к разным парам q-битов, результат будет выглядеть гораздо сложнее. Не видно способа определить этот результат, кроме прямого перемножения матриц. Поскольку размеры матриц экспоненциально велики, потребуется экспоненциальное время для их перемножения.

Заметим однако, что вычисление матричных элементов возможно на полиномиально ограниченной памяти. Пусть нужно найти матричный элемент U_{xy} оператора

$$U = U^{(l)}[j_l, k_l]\, U^{(l-1)}[j_{l-1}, k_{l-1}] \cdot \ldots \cdot U^{(2)}[j_2, k_2]\, U^{(1)}[j_1, k_1].$$

Очевидно, что

$$\left(U^{(l)} \cdot \ldots \cdot U^{(1)}\right)_{x_l x_0} = \sum_{x_{l-1}, \ldots, x_1} U^{(l)}_{x_l x_{l-1}} \cdot \ldots \cdot U^{(1)}_{x_1 x_0}. \tag{5.1}$$

(Здесь $x_0, \ldots, x_l$ — строки длиной n битов.) Для вычисления этой суммы достаточно $(l-1)$-го регистра для хранения текущих значений $x_{l-1}, \ldots, x_1$, ещё одного регистра для хранения частичной суммы и некоторого фиксированного количества регистров для вычисления промежуточных произведений.

Определение 5.1. ***Квантовая схема.*** *Пусть $\mathcal{A}$ — некоторое множество унитарных операторов (базис). Тогда квантовая схема в базисе $\mathcal{A}$ — это последовательность $U_1[A_1], \ldots, U_l[A_l]$, где A_j — множества q-битов, $U_j \in \mathcal{A}$.*

Оператор, реализуемый квантовой схемой. *Это оператор $U\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$, равный $U_l[A_l] \cdot \ldots \cdot U_1[A_1]$.*

Это определение не очень хорошо, так как не учитывает возможность использования дополнительной памяти в процессе вычисления. Поэтому дадим ещё одно определение.

Оператор $U\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$, реализуемый схемой в расширенном смысле. *Это такой оператор, что произведение*

$$W = U_l[A_l] \cdot \ldots \cdot U_1[A_1],$$

действующее на N q-битов, $N \geqslant n$, для любого вектора $|\xi\rangle \in \mathcal{B}^{\otimes n}$ удовлетворяет условию $W(|\xi\rangle \otimes |0^{N-n}\rangle) = (U|\xi\rangle) \otimes |0^{N-n}\rangle$.

Таким образом, мы «берём напрокат» дополнительную память, заполненную нулями, и должны возвратить ее в прежнем состоянии. Какой смысл имеет такое определение? Зачем нужно требовать, чтобы дополнительные q-биты вернулись в состояние $|0^{N-n}\rangle$? На самом деле это условие чисто техническое, однако важно, чтобы вектор состояния в конце вычисления был *разложим*, т. е. имел вид $|\xi'\rangle \otimes |\eta'\rangle$ (с произвольным $|\eta'\rangle$). Если это так, то первая подсистема находится в определённом состоянии $|\xi'\rangle$, поэтому про вторую подсистему (дополнительную память) можно забыть. В противном случае, совместное состояние двух подсистем оказывается «запутанным» (entangled), поэтому первую подсистему нельзя отделить от второй.

6. Соотношение между классическим и квантовым вычислением

Классический объект, соответствующий унитарному оператору, — перестановка. Любой перестановке $G\colon \mathbb{B}^k \to \mathbb{B}^k$ естественно сопоставляется унитарный оператор $\widehat{G}$ в пространстве $\mathcal{B}^{\otimes k}$, действующий по правилу $\widehat{G}|x\rangle \stackrel{\text{def}}{=} |Gx\rangle$.

Аналогично определению 5.1, можно определить обратимые классические схемы, реализующие перестановки.

Определение 6.1. ***Обратимая классическая схема.*** *Пусть $\mathcal{A}$ — некоторое множество перестановок вида $G\colon \mathbb{B}^k \to \mathbb{B}^k$ (базис). Обратимая классическая схема в базисе $\mathcal{A}$ — это последовательность перестановок $U_1[A_1], \ldots, U_l[A_l]$, где A_j — множества битов, $U_j \in \mathcal{A}$.*

Перестановка, реализуемая обратимой схемой. *Это произведение перестановок $U_l[A_l] \cdot \ldots \cdot U_1[A_1]$.*

Перестановка U, реализуемая схемой в расширенном смысле. *Это такая перестановка, что произведение перестановок*

$$W = U_l[A_l] \cdot \ldots \cdot U_1[A_1]$$

(действующее на N битов, $N \geqslant n$) для любого $x \in \mathbb{B}^n$ удовлетворяет условию $W(x, 0^{N-n}) = (Ux, 0^{N-n})$.

В каких случаях функцию, заданную булевой схемой, можно реализовать обратимой схемой? Обратимые схемы реализуют только перестановки. Преодолеть эту трудность можно так. Вместо вычисления функции $F\colon \mathbb{B}^n \to \mathbb{B}^m$ будем вычислять функцию $F_\oplus\colon \mathbb{B}^{n+m} \to \mathbb{B}^{n+m}$, заданную соотношением $F_\oplus(x, y) = (x, y \oplus F(x))$ (здесь $\oplus$ означает побитовое сложение по модулю 2). Тогда значение $F(x)$ можно получить так: $F_\oplus(x, 0) = (x, F(x))$.

Чтобы можно было вычислять функции, заданные булевыми схемами в полном базисе, недостаточно взять базис для обратимых схем из перестановок на двух битах. Оказывается, что любая перестановка на двух битах $g\colon \mathbb{B}^2 \to \mathbb{B}^2$ является линейной функцией (при естественном отождествлении множества $\mathbb{B}$ и поля из двух элементов $\mathbb{F}_2$): $g(x, y) = (ax \oplus by \oplus c, dx \oplus ey \oplus f)$, где a, b, c, d, e, $f \in \mathbb{F}_2$. Поэтому все

функции, вычисляемые обратимыми схемами в базисе из перестановок на двух битах, являются линейными.

А вот перестановок на трёх битах уже достаточно, чтобы реализовать любую функцию. При этом не обязательно использовать все перестановки, достаточно включить в базис лишь две функции — отрицание $\neg$ и *элемент Тоффоли* $\wedge_\oplus\colon (x, y, z) \mapsto (x, y, z \oplus xy)$. При этом имеется в виду реализуемость в расширенном смысле, т. е. можно брать напрокат биты в состоянии 0 и возвращать их после окончания вычислений в том же состоянии.

Задача 6.1. Докажите для обратимых схем полноту базиса, состоящего из отрицания и элемента Тоффоли.

Лемма 6.1. *Пусть функция $F\colon \mathbb{B}^n \to \mathbb{B}^m$ реализуется булевой схемой размера L в некотором базисе $\mathcal{A}$. Тогда можно реализовать функцию $(x, 0) \mapsto (F(x), G(x))$ обратимой схемой размера $O(L)$ в базисе $\mathcal{A}_\oplus$, состоящем из функций $f_\oplus$ ($f \in \mathcal{A}$), а также функции $\bigoplus\colon (x, y) \mapsto (x, x \oplus y)$.*

Замечание 6.1. Помимо «полезного» ответа $F(x)$ схема, указанная в формулировке леммы, производит «мусор» $G(x)$.

Замечание 6.2. Содержательный смысл операции $\bigoplus$ — обратимое копирование бита (если начальное значение y равно 0). В литературе эта операция обычно называется Controlled NOT по причинам, которые станут ясными из дальнейшего.

Замечание 6.3. Применяя функцию $(\leftrightarrow)\colon (a, b) \mapsto (b, a)$ можно менять биты местами в записи. Обратите внимание, что для перестановок битов достаточно также иметь в базисе $\bigoplus$, так как

$$(\leftrightarrow)[j, k] = \bigoplus[j, k] \bigoplus[k, j] \bigoplus[j, k].$$

Доказательство. Возьмём схему, вычисляющую F. Пусть входные переменные — это $x_1, \ldots, x_n$. Вспомогательные переменные схемы и биты результата — это $x_{n+1}, \ldots, x_{n+L}$; в обратимой схеме сопоставим им дополнительные биты, имеющие в начальном состоянии значение 0.

Каждое присваивание в схеме имеет вид $x_{n+k} := f_k(x_{j_k}, \ldots, x_{l_k})$, $f_k \in \mathcal{A}$, $j_k, \ldots, l_k < n + k$. В обратимой схеме аналогом присваивания будет действие перестановки $(x_{j_k}, \ldots, x_{l_k}, x_{n+k}) := (f_k)_\oplus(x_{j_k}, \ldots, x_{l_k}, x_{n+k})$, т. е. $x_{n+k} := x_{n+k} \oplus f_k(x_{j_k}, \ldots, x_{l_k})$.

Поскольку начальные значения дополнительных переменных были равны 0, их конечные значения будут такими же, как и в булевой схеме.

Осталось поменять местами биты, чтобы получить указанный в условии формат ответа.

Весь процесс вычисления удобно представить следующей схемой (над прямоугольниками подписано количество битов, внутри — их содержимое):

n	$L-m$	m	
x	0	0	
			присваивания по схеме
x	$x_{n+1}\ \ldots\ x_{L-m}$	$F(x)$	
			перестановки битов
$F(x)$	$G(x)$		

□

Лемма 6.2 (очистка мусора). *В условиях леммы 1 можно произвести вычисление функции $F_\oplus$ обратимой схемой размера $O(L+n+m)$ (с использованием дополнительных битов).*

Доказательство. Для очистки мусора будет использована обратимость. Изобразим процесс вычисления $F_\oplus$ схемой, аналогичной той, что приведена в доказательстве леммы 6.1.

n	L	m	
x	0	y	
			вычисление по схеме из доказательства леммы 6.1
m	$L+n-m$	m	
$F(x)$	$G(x)$	y	
			сложение $F(x)$ и y по модулю 2
$F(x)$	$G(x)$	$F(x)\oplus y$	
			обращение вычислений, сделанных на первом шаге
x	0	$F(x)\oplus y$	

□

Замечание 6.4. Обратимыми вычислениями заинтересовались при попытке ответить на вопрос, какая энергия необходима для вычислений (классических). Анализ показал, что потери энергии можно устремить к нулю для всех вычислительных операций, кроме необратимых. Когда производится необратимая операция (например, стирание бита), два различных логических значения (0 и 1) становятся одинаковыми (0). Однако физические законы на микроуровне являются обратимыми, поэтому отличие между старыми состояниями (0 и 1) должно сохраниться в каких-то неконтролируемых физических степенях свободы. Это можно интерпретировать как возрастание беспорядка (энтропии), которое

в конечном счете проявится в окружающей среде в виде тепла. Величина энергии, требуемой для стирания одного бита, очень мала ($\approx kT$), но конечна. Потеря энергии из-за необратимого стирания информации при форматировании жёсткого диска ёмкостью 1 Гб равна $3\cdot 10^{-11}$ Дж, что примерно соответствует затратам энергии на сдвиг головки диска на половину диаметра атома водорода. Это на много порядков меньше реального перемещения головки при форматировании.

С другой стороны, если ёмкость дисков будет расти столь же быстро, как в настоящее время, то к концу XXIII века для форматирования жёсткого диска потребуется энергия, соответствующая годовому излучению Солнца.

Лемма об очистке мусора показывает, что можно избежать таких потерь энергии, связанных с необратимостью вычислений.

Можно также показать, что любое вычисление, требующее памяти L, можно реализовать обратимым образом с использованием памяти, не превышающей $L^{O(1)}$. Приведём набросок доказательства.

Поскольку задача TQBF PSPACE-полна, то достаточно вычислять на небольшой памяти значение формулы

$$\exists\, x_1\, \forall\, y_1\ \ldots\ \exists\, x_M\, \forall\, y_M\ f(x_1, y_1, \ldots, x_M, y_M, z) \tag{6.1}$$

при условии, что $f(\cdot)$ вычислима булевой схемой небольшого размера L. Покажем, что в этом случае значение формулы 6.1 можно вычислить на памяти $O(L+M)$. Вычисление будет организовано рекурсивно, начиная с внутренних кванторов.

Чтобы вычислить $\forall\, x\ F(x,z)$, вычислим $F(0,z)$, занесём результат в одну дополнительную ячейку, затем вычислим $F(1,z)$ и занесём результат в другую ячейку. После вычислим $\forall\, x\ F(x,z) = F(0,z)\wedge F(1,z)$ и результат занесём в третью ячейку. Чтобы убрать мусор, прокрутим все вычисления, кроме последнего шага, в обратном направлении.

Разобравшись аналогичным образом с формулой $\exists\, x\ F(x,z)$, приходим к такому выводу: добавление квантора по булевой переменной увеличивает требуемую память не более чем на константу битов.

В заключение сформулируем теорему о вычислении обратимых функций, которая является прямым обобщением леммы 6.2.

Теорема 6.1. *Пусть F и F^{-1} вычислимы булевыми схемами размеров $\leqslant L$. Тогда F реализуется обратимой схемой размера $O(L+n)$.*

Доказательство. Даётся следующей схемой вычислений (для простоты не указаны биты, которые «берутся напрокат» при вычислении из леммы 6.2).

n	n	
x	0	
		вычисление $F_\oplus$ по схеме из доказательства леммы 6.2
x	$F(x)$	
		перестановки битов
$F(x)$	x	
		применение $(F^{-1})_\oplus$ (по схеме из доказательства леммы 6.2) даёт в правом регистре $x \oplus F^{-1}(F(x)) = 0$
$F(x)$	0	

$\square$

7. Базисы для квантовых схем

Как выбрать базис для вычислений в квантовых схемах? Унитарных операторов бесконечно много, поэтому либо полный базис должен содержать бесконечное количество элементов, либо мы должны ослабить условие точной реализуемости оператора схемой, заменив его на условие приближённой реализуемости. Мы рассмотрим обе возможности.

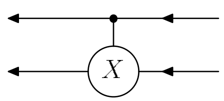
Точная реализация.

Теорема 7.1. *Базис, содержащий все унитарные операторы, действующие на парах q-битов, позволяет реализовать любой унитарный оператор в расширенном смысле.*

Для доказательства этой теоремы введем важный класс операторов: *операторы с квантовым управлением.*

Определение 7.1. *Определим по оператору* $U\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$ *оператор* $\Lambda(U)\colon \mathcal{B} \otimes \mathcal{B}^{\otimes n} \to \mathcal{B} \otimes \mathcal{B}^{\otimes n}$ ***с квантовым управляющим q-битом*** (*первый сомножитель*) *следующими соотношениями:*

$$\begin{aligned} \Lambda(U)\,|0\rangle \otimes |\xi\rangle &= |0\rangle \otimes |\xi\rangle, \\ \Lambda(U)\,|1\rangle \otimes |\xi\rangle &= |1\rangle \otimes U|\xi\rangle. \end{aligned} \tag{7.1}$$



Графически будем изображать оператор $\Lambda(X)$ с квантовым управлением как показано на рисунке. Верхняя линия соответствует первому сомножителю, нижняя линия — второму. Направление стрелок соответствует направлению перемножения операторов (справа налево).

Нам потребуются также операторы с несколькими управляющими q-битами:

$$\Lambda^k(U)\,|x_1,\ldots,x_k\rangle \otimes |\xi\rangle = \begin{cases} |x_1,\ldots,x_k\rangle \otimes |\xi\rangle, & \text{если } x_1 \cdot \ldots \cdot x_k = 0, \\ |x_1,\ldots,x_k\rangle \otimes U|\xi\rangle, & \text{если } x_1 \cdot \ldots \cdot x_k = 1. \end{cases} \tag{7.2}$$

Пример 7.1. Пусть $\sigma^x \overset{\text{def}}{=} \widehat{\neg} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Тогда $\Lambda(\sigma^x) = \widehat{\oplus}$, а $\Lambda^2(\sigma^x) = \widehat{\wedge_\oplus}$ (элемент Тоффоли).

Теперь построим элемент Тоффоли, используя преобразования двух q-битов. Для начала найдем пару операторов, удовлетворяющих следующему соотношению $XYX^{-1}Y^{-1} = i\sigma^x$. Например, годится такая пара:

$$X = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}; \quad Y = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Поясним геометрический смысл этой конструкции. Унитарная группа $\mathbf{U}(2)$ действует на трёхмерном евклидовом пространстве. Чтобы описать это действие, заметим, что эрмитовы матрицы 2×2 с нулевым следом образуют трёхмерное евклидово пространство: скалярное произведение задаётся формулой $\frac{1}{2}\operatorname{Tr}(XY)$, ортонормированный базис образуют *матрицы Паули*

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Унитарный оператор $U \in \mathbf{U}(2)$ действует на этом пространстве так: $U\colon E \mapsto UEU^{-1}$. Можно доказать (см. [8, §11.12]), что описанное действие задаёт изоморфизм $\mathbf{U}(2)/\mathbf{U}(1) \cong \mathbf{SO}(3)$, где $\mathbf{U}(1)$ — подгруппа фазовых сдвигов, а $\mathbf{SO}(3)$ — группа поворотов в трёхмерном пространстве (т. е. группа ортогональных преобразований с детерминантом, равным 1).

При этом действии σ^x соответствует поворот вокруг оси x на $180°$, X соответствует поворот вокруг x на $90°$, а Y соответствует поворот вокруг z на $180°$.

На рис. 1 изображено графическое представление схемы, вычисляющей элемент Тоффоли с помощью операторов $\Lambda(X)$, $\Lambda(Y)$ и $\Lambda^2(-i)$. Последний — это управляемый двумя битами фазовый сдвиг (умножение на $-i$). Проверим эту схему. Пусть на вход подаётся вектор $|a\rangle \otimes |b\rangle \otimes |\xi\rangle$, где $a, b \in \mathbb{B}$, $|\xi\rangle \in \mathcal{B}$. Если $a = b = 1$, то к $|\xi\rangle$ будет применён оператор $-iXYX^{-1}Y^{-1} = \sigma^x$, т. е. $|0\rangle$ и $|1\rangle$ в третьем q-бите переставляются. Если же хотя бы один из управляющих битов равен 0, то к $|\xi\rangle$ будет применён тождественный оператор. Это и есть действие

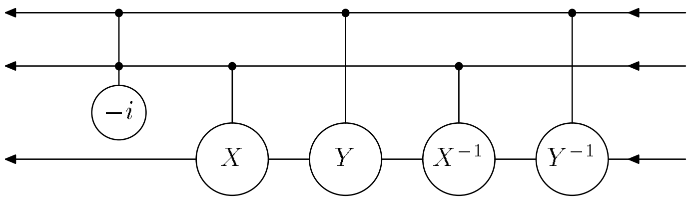


Рис. 1.

элемента Тоффоли.

С помощью элемента Тоффоли можно реализовать любую перестановку базисных векторов (с использованием дополнительной памяти).

Покажем, как реализовать оператор $\Lambda^k(U)$ для любого k, действуя только на пары q-битов. Для этого также потребуется дополнительная память. Будем строить оператор W, действующий в пространстве N q-битов $\mathcal{B}^{\otimes N}$ и удовлетворяющий условию

$$W\left(|\eta\rangle \otimes |0^{N-k-1}\rangle\right) = \Lambda(U)|\eta\rangle \otimes |0^{N-k-1}\rangle.$$

(Предостережение: это условие **не означает**, что $W = \Lambda(U) \otimes I$.)

Существует обратимая схема P размера $O(k)$, вычисляющая произведение входных битов (с мусором); графически она представлена на рис. 2 (сверху обозначено число битов в каждом из выделенных фрагментов памяти).

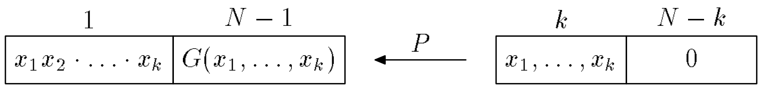


Рис. 2.

На рис. 3 показано, как с помощью схемы P и оператора с одним управляющим q-битом построить оператор $\Lambda^k(U)$. Мы применяем схему P, а затем — обратную схему P^{-1}, после чего все вспомогательные биты возвращаются в исходное состояние. В промежутке самой верхней линии соответствует бит со значением $x_1\cdot\ldots\cdot x_k$. Его мы и используем для управления оператором U, действующим на самой нижней линии. Другим способом $\Lambda^k(U)$ можно записать как $P^{-1}\Lambda(U)P$.

Действие $\Lambda^k(U)$ можно описать так: на подпространстве, порождённом векторами $|1,\ldots,1,0\rangle$ и $|1,\ldots,1,1\rangle$, действует оператор U, а на ортогональном дополнении к этому подпространству — тождественный оператор. Наша следующая задача: реализовать оператор, который устроен так же, но нетривиальное действие осуществляется на под-

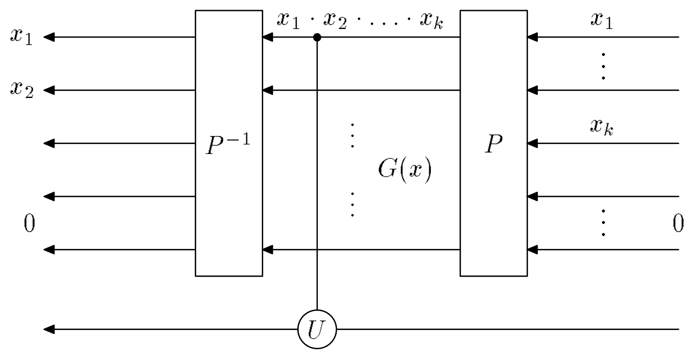


Рис. 3.

пространстве, натянутом на произвольную пару базисных векторов. Пусть мы хотим реализовать произвольный оператор в подпространстве, натянутом на базисные векторы $|x\rangle$ и $|y\rangle$, где $x = (x_1,\ldots,x_n)$, $y = (y_1,\ldots,y_n)$, x_j, $y_j \in \mathbb{B}$. Пусть f — такая перестановка, что $f(x) = (1,\ldots,1,0)$, $f(y) = (1,\ldots,1,1)$. Тогда нужный нам оператор представляется в виде $\widehat{f}^{-1}\Lambda^{n-1}(U)\widehat{f}$. (Напомним, что $\widehat{f}$ — оператор, соответствующий перестановке f.)

Итак, на парах базисных векторов мы можем действовать произвольно. Пока все использованные схемы имели размер $O(n)$, так что построенные действия реализуются эффективно. Следующая часть неэффективна. Имеет место следующая лемма.

Лемма 7.1. *Любая унитарная матрица U в пространстве $\mathbb{C}^M$ может быть представлена в виде произведения $M(M-1)/2$ матриц вида*

$$\begin{pmatrix} 1 & 0 & \ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & & \\ \vdots & \ddots & 0 & \ldots\ldots\ldots\ldots\ldots\ldots & & & \\ 0 & \ldots & 1 & 0 & \ldots\ldots\ldots & & \\ 0 & \ldots\ldots & & \begin{pmatrix} a & b \\ c & d \end{pmatrix} & 0 & \ldots\ldots & \\ 0 & \ldots\ldots\ldots\ldots\ldots & & & 1 & 0 & 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & & & \ddots & 0 \\ 0 & \ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & & & 1 \end{pmatrix}, \text{ где } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbf{U}(2).$$

Заметим, что в нашем случае $M = 2^n$, так что получаем представление в виде произведения экспоненциального большого числа базисных операторов.

Приближённая реализация. Теперь перейдём к конечным базисам. В этом случае возможно только приближённое представление операторов произведениями базисных. Чтобы определить приближённую реализацию, нам потребуется норма на пространстве операторов.

На пространстве состояний есть норма $\||\xi\rangle\| = \sqrt{\langle\xi|\xi\rangle}$. Она, как и любая норма, по определению удовлетворяет следующим условиям:

$$\||\xi\rangle\| \begin{cases} = 0, & \text{если } |\xi\rangle = 0, \\ > 0, & \text{если } |\xi\rangle \neq 0; \end{cases} \tag{7.3}$$

$$\||\xi\rangle + |\eta\rangle\| \leqslant \||\xi\rangle\| + \||\eta\rangle\|; \tag{7.4}$$

$$\|c|\xi\rangle\| = |c|\||\xi\rangle\|. \tag{7.5}$$

Введём теперь норму на пространстве операторов. Пусть $\mathcal{N}$ — пространство с нормой. Пространство операторов, действующих на нём, можно представить как $\mathbf{L}(\mathcal{N}) = \mathcal{N} \otimes \mathcal{N}^*$ (изоморфизм задается матричным представлением $\sum a_{jk}|j\rangle\langle k|$).

Определение 7.2. *Норма оператора X (так называемая операторная норма, вообще говоря, есть и другие) равна*

$$\|X\| = \sup_{|\xi\rangle\neq 0} \frac{\|X|\xi\rangle\|}{\||\xi\rangle\|}.$$

Заметим, что $\|X\|^2$ — наибольшее собственное число оператора $X^\dagger X$.

Эта норма обладает всеми перечисленными выше свойствами нормы, а кроме того, еще несколькими специфическими:

$$\|XY\| \leqslant \|X\|\,\|Y\|; \tag{7.6}$$

$$\|X^\dagger\| = \|X\|; \tag{7.7}$$

$$\|X \otimes Y\| = \|X\|\,\|Y\|, \qquad \text{где } X \in \mathbf{L}(\mathcal{N}),\ Y \in \mathbf{L}(\mathcal{M}). \tag{7.8}$$

Доказательства этих свойств нормы получаются непосредственно из определения и оставляются в качестве упражнения.

Дадим теперь определение приближенной реализуемости. Если искомый оператор — U, то его приближённая реализация будет обозначаться $\tilde{U}$.

Определение 7.3. *Оператор $\tilde{U}$ представляет оператор U с точностью δ, если $\|\tilde{U} - U\| \leqslant \delta$.*

У этого определения есть два замечательных свойства. Во-первых, если мы имеем произведение нескольких операторов $U = U_L \cdot \ldots \cdot U_2 U_1$, каждый из которых имеет свое приближение $\tilde{U}_k$ с точностью δ_k, то

произведение этих приближений $\tilde{U} = \tilde{U}_L \cdot \ldots \cdot \tilde{U}_2\tilde{U}_1$ приближает U с точностью $\sum \delta_k$ (***ошибки накапливаются линейно***):

$$\|\tilde{U}_L \cdot \ldots \cdot \tilde{U}_1 - U_L \cdot \ldots \cdot U_1\| \leqslant \sum_j \delta_j.$$

Достаточно рассмотреть пример с двумя операторами:

$$\begin{aligned} &\|\tilde{U}_2\tilde{U}_1 - U_2U_1\| = \|\tilde{U}_2(\tilde{U}_1 - U_1) + (\tilde{U}_2 - U_2)U_1\| \leqslant \\ &\leqslant \|\tilde{U}_2(\tilde{U}_1 - U_1)\| + \|(\tilde{U}_2 - U_2)U_1\| \leqslant \|\tilde{U}_2\|\,\|(\tilde{U}_1 - U_1)\| + \|(\tilde{U}_2 - U_2)\|\,\|U_1\| = \\ &\qquad = \|(\tilde{U}_1 - U_1)\| + \|(\tilde{U}_2 - U_2)\|. \end{aligned}$$

В этой выкладке последнее равенство справедливо благодаря унитарности операторов. (Если рассматривать неунитарные операторы, то ошибки приближения могут накапливаться гораздо быстрее, например, экспоненциально.)

Замечание 7.1. Всякая модель, претендующая на решение сложных задач какими-то реальными физическими процессами, должна обязательно изучаться на предмет устойчивости к ошибкам приближения. (В реальной жизни параметры любого физического процесса можно задать лишь с некоторой точностью.) В частности, вычисление с экспоненциальным накоплением ошибок почти заведомо бесполезно с практической точки зрения.

Второе свойство понятия «$\tilde{U}$ представляет U с точностью δ» мы сформулируем в более общем контексте.

Определение 7.4. *Оператор $U\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$* ***приближается в расширенном смысле*** *оператором $\tilde{U}\colon \mathcal{B}^{\otimes N} \to \mathcal{B}^{\otimes N}$ с точностью δ, если для любого $|\xi\rangle$ из $\mathcal{B}^{\otimes n}$ выполнено*

$$\|\tilde{U}\left(|\xi\rangle \otimes |0^{N-n}\rangle\right) - U|\xi\rangle \otimes |0^{N-n}\rangle\| \leqslant \delta\||\xi\rangle\|. \tag{7.9}$$

Сформулируем это определение ещё одним способом. Введём оператор $V\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes N}$, который действует по правилу $V\colon |\xi\rangle \mapsto |\xi\rangle \otimes |0^{N-n}\rangle$. Оператор V не унитарный, но изометричный. Условие из последнего определения можно переписать так

$$\|\tilde{U}V - VU\| \leqslant \delta. \tag{7.10}$$

Рассуждение про накопление ошибок проходит и в этом случае (что, конечно, следует проверить).

Справедливо следующее утверждение: если $\tilde{U}$ приближает (в расширенном смысле) U с точностью δ, то $\tilde{U}^{-1}$ приближает U^{-1} с той же точностью δ. Это следует из того, что $\|W_1XW_2\| = \|X\|$ для унитарных операторов W_1, W_2. Умножая выражение под нормой в (7.10) слева на $\tilde{U}^{-1}$, а справа — на U^{-1}, получим следствие из неравенства (7.10): $\|\tilde{U}^{-1}V - VU^{-1}\| \leqslant \delta$.

Определение 7.5. *Будем называть базис $\mathcal{A}$ полным, если любой унитарный оператор U можно с любой точностью представить в расширенном смысле квантовой схемой в базисе $\mathcal{A}$.*

Теорема 7.2 (см. [4]). *Базис $\mathcal{Q} = \{H, K, \Lambda(\sigma^x), \Lambda^2(\sigma^x)\}$, где*

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad K = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix},$$

является полным. (Такой базис будем называть **стандартным**.)

Доказательство этой теоремы следует из решения задач 7.5–7.9.

Замечание 7.2. Если убрать из базиса $\mathcal{Q}$ квантовый элемент Тоффоли, он перестает быть полным. Однако многие важные вычисления можно делать и в таком усеченном базисе. В частности, как будет видно в дальнейшем, схемы, исправляющие ошибки, можно реализовать без элемента Тоффоли.

Можно оценить сложность реализации оператора U в этом базисе. Если $U : \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$, то можно реализовать этот оператор с точностью δ квантовой схемой в базисе $\mathcal{Q}$ размера $L = \exp(O(n)) \cdot \mathrm{poly}(\log(1/\delta))$. Если матричные элементы U заданы в двоичной записи, то эта схема строится по U с помощью некоторого алгоритма примерно за то же время (множители и степени полинома могут отличаться). Идея построения такого алгоритма легко усматривается из задач 7.1 и 7.11.

Задачи

7.1. Докажите, что все операторы на одном q-бите в сочетании с оператором $\Lambda(\sigma^x)$ образуют полный базис. Решение должно быть достаточно эффективным: должен существовать алгоритм, который строит схему, реализующую произвольный оператор U на n q-битах, за время $\exp(O(n)) \cdot \mathrm{poly}(\log(1/\delta))$.

7.2. Докажите свойства операторной нормы (7.6–7.8).

7.3. Пусть операторы $\tilde{U}_k$ приближают в расширенном смысле операторы U_k с точностью δ_k, $1 \leqslant k \leqslant L$. Докажите, что оператор $\tilde{U}_L \cdot \ldots \cdot \tilde{U}_1$ приближает в расширенном смысле оператор $U_L \cdot \ldots \cdot U_1$ с точностью $\sum \delta_k$.

7.4. Пусть оператор $\tilde{U}$ приближает в расширенном смысле оператор U с точностью δ. Докажите, что существует оператор W, точно представляющий U в расширенном смысле, т. е. выполняется равенство

$$W\left(|\xi\rangle \otimes |0^{N-n}\rangle\right) = (U|\xi\rangle) \otimes |0^{N-n}\rangle,$$

и такой, что $\|W - \tilde{U}\| \leqslant O(\delta)$.

7.5. Пусть унитарный оператор $U : \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$ удовлетворяет условию $U|0\rangle = |0\rangle$. Постройте реализующую $\Lambda(U)$ схему размера $6n + 1$ в базисе $\mathcal{Q} \cup \{U\}$, использующую оператор U один раз.

7.6. Пусть X, Y — некоммутирующие элементы группы $\mathbf{SO}(3)$ — повороты на углы, несоизмеримые с π. Докажите, что группа, порождённая X и Y, образует всюду плотное подмножество в $\mathbf{SO}(3)$.

7.7. Пусть $\mathcal{M}$ — унитарное пространство размерности $\geqslant 3$. Рассмотрим подгруппу $H \subset \mathbf{U}(\mathcal{M})$ — стабилизатор одномерного подпространства, порождённого некоторым единичным вектором $|\xi\rangle \in \mathcal{M}$. Пусть V — произвольный унитарный оператор, не сохраняющий подпространство $\mathbb{C}(|\xi\rangle)$. Докажите, что множество операторов $H \cup V^{-1}HV$ порождает всю группу $\mathbf{U}(\mathcal{M})$.

(Заметим, что в условии этой задачи $\mathbf{U}(\mathcal{M})$ и H можно профакторизовать по подгруппе фазовых сдвигов $\mathbf{U}(1)$).

7.8. Докажите, что операторы из стандартного базиса порождают всюду плотное множество в $\mathbf{U}(\mathcal{B}^{\otimes 2})/\mathbf{U}(1)$.

7.9. Докажите, что фазовые сдвиги можно реализовать в стандартном базисе, используя напрокат дополнительные q-биты.

7.10. Докажите, что отрицание σ^x и *элемент Дойча* $\Lambda^2(R)$, где $R = -i\exp(\pi i \alpha \sigma^x)$, α — иррациональное, образуют полный базис для квантового вычисления.

7.11. Докажите, что любой оператор U, действующий на одном q-бите, может быть приближённо реализован в расширенном смысле с точностью δ схемой размера $O(\log^3(1/\delta))$ в стандартном базисе, и есть полиномиальный алгоритм построения этой схемы по описанию U.

Эта задача довольно сложна, к её решению лучше приступать после знакомства с разделами 11 и 12 и решения задачи 12.3 (квантовое преобразование Фурье). Предлагаемый путь решения является достаточно изощрённым. В статье [4] был использован более прямой (но тоже неочевидный) подход, при котором получается схема размера $\mathrm{poly}(\log(1/\delta))$.

8. Определение квантового вычисления. Примеры

Пока мы описали работу квантового компьютера. Теперь пора определить, когда эта работа приводит к решению интересующей нас задачи. Определение будет похоже на определение вероятностного вычисления.

Пусть есть функция $F\colon \mathbb{B}^n \to \mathbb{B}^m$. Рассмотрим квантовую схему, работающую с N битами: $U = U_L \cdot \ldots \cdot U_2 U_1 : \mathcal{B}^{\otimes N} \to \mathcal{B}^{\otimes N}$. Неформально говоря, эта схема вычисляет F, если после применения U к начальному состоянию $|x, 0^{N-n}\rangle$, мы, «посмотрев» на первые m битов, *с большой вероятностью* «увидим» $F(x)$. (Остальные q-биты могут содержать произвольный мусор.)

Нужно только оговорить, что такое эта вероятность. Слова «посмотрев» и «увидим» в точном смысле означают, что производится *измерение* значений соответствующих q-битов. В результате измерения могут получаться разные ответы, каждому соответствует своя вероятность. Ниже (раздел 9) этот вопрос рассматривается подробно. Для того, чтобы дать определение квантового вычисления функции F, достаточно (не вдаваясь в обсуждение физических объяснений этого факта) принять следующее: *вероятность получения базисного состояния x при измерении состояния $|\psi\rangle = \sum_x c_x |x\rangle$ равна*

$$\mathbf{P}(|\psi\rangle, x) = |c_x|^2. \tag{8.1}$$

Нас интересует вероятность того, что компьютер закончит работу в состоянии вида $(F(x), z)$, где z — любое.

Определение 8.1. *Схема $U = U_L \cdot \ldots \cdot U_2 U_1$ вычисляет F, если для любого x выполнено*

$$\sum_z \left|\langle F(x), z|\, U\, |x, 0^{N-n}\rangle\right|^2 \geqslant 1 - \varepsilon,$$

где ε — некоторое фиксированное число, меньшее $1/2$. (Обратите внимание, что $F(x)$ и x состоят из разного количества битов, хотя суммарная длина $(F(x), z)$ и $(x, 0^{N-n})$ одинакова и равна N.)

Как и для вероятностных вычислений, выбор ε несущественен, поскольку можно запустить несколько экземпляров схемы независимо и выбрать тот результат, который получается чаще всего. Из оценки, приведённой на с. 38, следует, что для уменьшения вероятности неудачи в N раз нужно взять $O(\log N)$ экземпляров схемы U. Выбор самого частого результата реализуется классической схемой, использующей функцию голосования $MAJ(x_1, \ldots, x_n)$ (она равна 1, когда более половины её аргументов равны 1, и равна 0 в противном случае). Функция

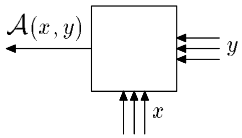
$MAJ(x_1, \ldots, x_n)$ реализуется в полном базисе схемой размера $O(n \log n)$, так что потеря эффективности при уменьшении вероятности неудачи в N раз задаётся множителем $O(m \log N \log\log N)$.

Задача 8.1. Докажите, что приведенное рассуждение является корректным в квантовом случае: функция $MAJ_\oplus$ реализована в виде обратимой схемы, на вход которой подаются выходные q-биты n копий схемы U.

8.1. Квантовый поиск: алгоритм Гровера.

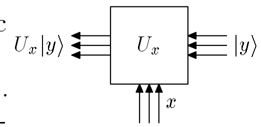
Итак, мы имеем определение квантового вычисления. Теперь можно заняться сравнением эффективности классического и квантового вычисления. Во введении упоминались три основных примера, для которых квантовое вычисление оказывается, по-видимому, эффективнее классического. Мы начнём с того из них, в котором квантовое вычисление заведомо эффективнее (хотя ускорение лишь «полиномиальное»).

Дадим определение *универсальной переборной задачи* в классической и квантовой постановке.



Пусть имеется устройство (см. рисунок), которое по входам x и y определяет значение некоторого предиката $\mathcal{A}(x, y)$. Нас интересует предикат $F(x) = \exists y\, \mathcal{A}(x, y)$. Это похоже на определение класса NP, но сейчас нам недоступна внутренняя структура устройства, вычисляющего предикат $\mathcal{A}$. В таких условиях на классическом компьютере значение предиката $F(x)$ нельзя вычислить быстрее, чем за $N = 2^n$ шагов, где n — количество битов в записи y.

Оказывается, что на квантовом компьютере можно вычислить значение предиката $F(x)$ и даже найти y, на котором выполнено $\mathcal{A}(x, y)$, за время $O(\sqrt{N})$. Получены также и нижние оценки, показывающие, что в этой постановке квантовые устройства дают лишь полиномиальное ускорение по сравнению с классическими.



В квантовой постановке задача выглядит так. Вход x по-прежнему классический, но сам «чёрный ящик» — квантовое устройство, и вход y (варианты ответа) мы будем считать квантовым. Поэтому наш оракул (или «чёрный ящик») задаёт оператор U_x, действующий по правилу

$$U_x\,|y\rangle = \begin{cases} |y\rangle, & \text{если } \mathcal{A}(x,y) = 0,\\ -|y\rangle, & \text{если } \mathcal{A}(x,y) = 1. \end{cases}$$

Нужно вычислить значение $F(x)$ и найти «ответ» y (при котором выполнен $\mathcal{A}(x,y)$).

Результаты, о которых уже упоминалось, формулируются так (см. [31, 48]): *существуют две константы C_1 и C_2 такие, что есть схема размера $\leqslant C_1\sqrt{N}$, решающая задачу для любого предиката $\mathcal{A}(x,y)$; а для любой схемы размера $\leqslant C_2\sqrt{N}$ существует предикат $\mathcal{A}(x,y)$, при котором задача не решается на этой схеме (т. е. схема даёт неправильный ответ с вероятностью $>1/3$).*

Мы разберём упрощённую постановку: считаем, что «ответ» существует и единствен, обозначим его через y_0; нужно найти y_0. Схема, которую мы для этого построим, будет примером «прямого» квантового вычисления; она будет описана в терминах преобразований базисных векторов.

Рассмотрим два оператора:

$$U = I - 2|y_0\rangle\langle y_0|$$

$$\text{и} \quad V = I - 2|\xi\rangle\langle\xi|, \quad \text{где } |\xi\rangle = \frac{1}{\sqrt{N}}\sum_y |y\rangle.$$

Оператор V в матричной форме может быть записан так (напомним, что $N=2^n$):

$$V = \begin{pmatrix} 1-\frac{2}{N} & \dots & -\frac{2}{N} \\ \vdots & \ddots & \vdots \\ -\frac{2}{N} & \dots & 1-\frac{2}{N} \end{pmatrix}.$$

Оператор U нам задан (это оракул). Построим квантовую схему, вычисляющую V. Действовать будем так: переведем $|\xi\rangle$ в $|0^n\rangle$ некоторым оператором W, затем применим оператор $Y = I - 2|0^n\rangle\langle 0^n|$, после чего применим W^{-1}.

Построить оператор W, который переводит $|\xi\rangle$ в $|0^n\rangle$, просто. Это $W = H^{\otimes n}$, где оператор H — из стандартного базиса (см. с. 66). Действительно, $|\xi\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle+|1\rangle\right)^{\otimes n}$, а $H\colon \frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right) \mapsto |0\rangle$.

Теперь построим реализацию оператора Y. Используем обратимую классическую схему, реализующую оператор $Z\colon \mathbb{B}^{n+1}\to\mathbb{B}^{n+1}$,

$$Z|a_0,\dots,a_n\rangle = |a_0\oplus f(a_1,a_2,\dots,a_n), a_1,\dots,a_n\rangle;$$

$$f(a_1,\dots,a_n) = \begin{cases} 1, & \text{если } a_1=\ldots=a_n=0,\\ 0, & \text{если } \exists j : a_j\neq 0.\end{cases}$$

(С точностью до перестановки аргументов, $Z = \widehat{f_\oplus}$.) Поскольку f имеет малую схемную сложность (в классическом смысле), по лемме 6.2 для вычисления Z существует небольшая схема (в которой «берутся напрокат» дополнительные q-биты).

Схема, реализующая оператор V, изображена на рис. 4. Центральная часть, включающая в себя Z, σ^z и Z, реализует оператор Y. В схеме используется оператор $\sigma^z = K^2$ (K из стандартного базиса).

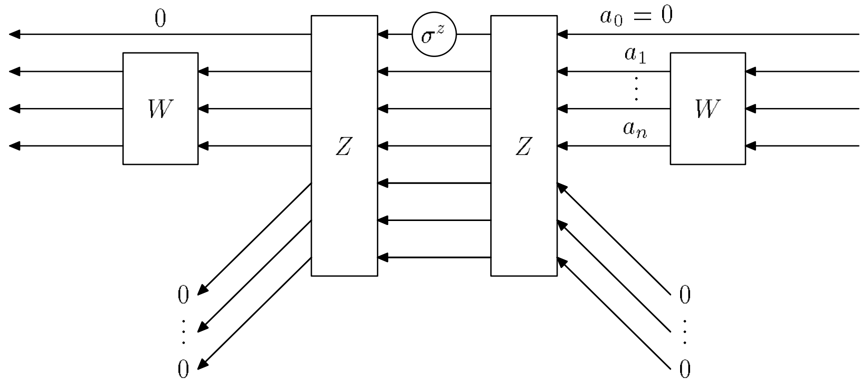


Рис. 4.

Заметим, что W^2 и Z^2 действуют тождественно на векторах с нулевыми значениями q-битов, взятых напрокат. Поэтому решающую роль играет оператор σ^z, действующий на вспомогательный q-бит, который также не меняется после всего вычисления.

Пусть вас не смущает то, что σ^z действует только на «управляемый» q-бит, а меняется в результате весь вектор. Вообще, различие между «чтением» и «записью» в квантовом случае неабсолютно и зависит от выбора базиса. Приведём соответствующий пример.

Напишем матрицу $\Lambda(\sigma^x)\colon |a,b\rangle \mapsto |a,a\oplus b\rangle$ в базисе $\frac{1}{\sqrt{2}}\left(|0\rangle \pm |1\rangle\right)$ для каждого из q-бит. Другими словами, запишем матрицу для оператора $X = (H\otimes H)\,\Lambda(\sigma^x)\,(H\otimes H)$. Схема для этого оператора изображена

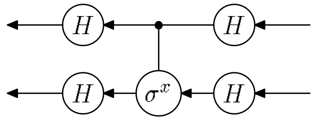


Рис. 5.

на рис. 5. Используя равенство $\Lambda(\sigma^x)|c,d\rangle = |c, c\oplus d\rangle$, найдём действие X на базисном векторе:

$$X|a,b\rangle = \frac{1}{2}\,(H\otimes H)\,\Lambda(\sigma^x)\sum_{c,d}(-1)^{ac+bd}|c,d\rangle =$$

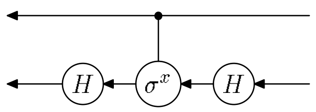
$$= \frac{1}{2}\,(H\otimes H)\sum_{c,d}(-1)^{ac+bd}|c,c\oplus d\rangle =$$

$$= \frac{1}{4}\sum_{a',b',c,d}(-1)^{a'c+b'(c+d)}(-1)^{ac+bd}|a',b'\rangle =$$

$$= \frac{1}{4}\sum_{a',b'}2\delta_{b,b'}\cdot 2\delta_{a,(a'+b')}|a',b'\rangle = |a\oplus b, b\rangle.$$

Итак, в базисе $\frac{1}{\sqrt{2}}\,(|0\rangle \pm |1\rangle)$ управляющий и управляемый биты поменялись местами. Какой бит «управляющий», или какой «читается», зависит от выбора базиса. Разумеется, такое положение дел противоречит нашей классической интуиции. Трудно представить, как при переходе к другому базису квантовый принтер вдруг становится квантовым сканнером.

Задача 8.2. Что будет, если изменить базис только в одном бите?



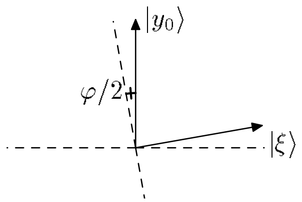
Например, как будет выглядеть матрица оператора, схема которого изображена на рисунке? Попробуйте также поменять базис в другом бите.

Вернёмся к построению схемы для универсальной переборной задачи. Оракул $U = I - 2|y_0\rangle\langle y_0|$ нам задан, и мы реализовали оператор $V = I - 2|\xi\rangle\langle\xi|$. Из вектора $|0^n\rangle$ можно получить вектор $|\xi\rangle$ применением оператора W ($W^2 = I$). Теперь с помощью операторов U и V построим из вектора $|\xi\rangle$ искомый вектор $|y_0\rangle$. Для этого будем поочередно действовать операторами V, U:

$$\ldots V\,U\,V\,U\,|\xi\rangle = (VU)^s|\xi\rangle.$$

Что при этом получается? Геометрически оба оператора есть отражения относительно гиперплоскости. Подпространство $\mathcal{L} = \mathbb{C}(|\xi\rangle, |y_0\rangle)$ инвариантно относительно обоих операторов, а, значит, и относительно VU. Поскольку вектор $|\xi\rangle$ принадлежит этому подпространству, достаточно рассмотреть действие VU на нём.

Композиция двух отражений относительно двух прямых есть поворот на удвоенный угол между этими прямыми. Угол легко вы-



числить, $\langle\xi|y_0\rangle = \frac{1}{\sqrt{N}} = \sin\frac{\varphi}{2}$, т.е. эти прямые почти перпендикулярны. Поэтому можно написать $VU = -R$, где R — поворот на малый угол φ. Но тогда $(VU)^s = (-1)^s R^s$, где R^s — поворот на угол $s\varphi$. Знак нас не интересует (фазовые множители не меняют вероятностей). При больших N имеем $\varphi \approx 2/\sqrt{N}$. Тогда после $s \approx \pi/4\sqrt{N}$ итераций исходный вектор повернется на угол $s\varphi \approx \pi/2$ и станет близок к искомому вектору. Это и означает, что система окажется в состоянии $|y_0\rangle$ с вероятностью, близкой к единице.

Если решается переборная задача в общей постановке (т.е. ответов может быть несколько, а может не быть вообще), требуются дополнительные технические ухищрения. Число шагов для поворота от исходного вектора к какому-нибудь вектору из подпространства, порожденного векторами ответов, обратно пропорционально корню из числа решений.

8.2. Универсальная квантовая схема. Второй из примеров, упомянутых во введении, — моделирование квантовомеханических систем. Это нечётко поставленная задача, так как большую роль в ней играет выбор конкретной системы и выделение «существенных» степеней свободы. С математической точки зрения, более корректно говорить о моделировании квантовых схем.

Квантовые схемы имеют конструктивное описание, если указать точность, с которой известны матричные элементы операторов, входящих в схему. Пусть есть описание квантовой схемы Z, размера $\leqslant L$ и точности δ. Элементами этой схемы могут быть любые унитарные операторы на $r = O(\log L)$ q-битах (так чтобы полная длина описания схемы не превышала $\mathrm{poly}(L, \log(1/\delta))$). Обозначим оператор, реализуемый этой схемой, через $Op(Z)$.

Из результатов задач 7.1 и 7.11 следует, что можно построить *универсальную квантовую схему* U размера $\mathrm{poly}(L, \log 1/\delta)$, которая моделирует работу произвольной квантовой схемы следующим образом. Если задано описание некоторой схемы $Op(Z)$ размера L и её вход $|\xi\rangle$, то

$$\Big\| U(|Z\rangle\otimes|\xi\rangle) - |Z\rangle\otimes Op(Z)|\xi\rangle \Big\| = O(L\delta).$$

8.3. Квантовые алгоритмы и класс BQP. До сих пор мы рассматривали неоднородные вычисления (вычислялись булевы функции). Алгоритмы вычисляют функции на словах произвольной длины. Определение квантового алгоритма можно дать, используя уже введённые квантовые схемы. Пусть есть функция $F\colon \mathbb{B}^* \to \mathbb{B}^*$, длина результата — полином от длины входа. Ей сопоставляется последовательность булевых функций (ограничения на входы длины n) $F_n\colon \mathbb{B}^n \to \mathbb{B}^{m(n)}$. *Квантовый алгоритм* для вычисления F — это однородная последовательность схем, вычисляющих F_n. «Однородная» означает, что по n можно построить описание соответствующей схемы на обычной полиномиально ограниченной машине Тьюринга. Будем говорить, что алгоритм работает за время $T(n)$, если размер схемы, вычисляющей F_n, равен $T(n)$.

Замечание 8.1. Можно определить квантовую машину Тьюринга и непосредственно через суперпозиции различных состояний ленты МТ (первоначальное определение Д. Дойча было именно таким). Наше определение оказывается эквивалентным.

Определение 8.2. *Функция $F\colon \mathbb{B}^* \to \mathbb{B}^*$ принадлежит классу* BQP, *если есть квантовый алгоритм её вычисления, работающий за время $O(n^d)$ для некоторой константы d.*

Как соотносится класс BQP с сложностными классами, введёнными ранее?

Задача 8.3. Докажите, что

$$\mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PP} \subseteq \mathrm{PSPACE}.$$

Класс PP состоит из предикатов вида

$$Q(x) = \Big(|\{y : R_0(x,y)\}| < |\{y : R_1(x,y)\}| \Big),$$

где $R_0, R_1 \in \mathrm{P}$, и учитываются только y с длиной меньше некоторого полинома $q(x)$.

Это почти всё, что известно о соотношениях между BQP и другими сложностными классами. Косвенное свидетельство в пользу строгого включения $\mathrm{BPP} \subset \mathrm{BQP}$ даёт существование эффективных квантовых алгоритмов для некоторых теоретико-числовых задач, традиционно считаемых трудными (см. раздел 12).

Заметим также, что в последнее время появились интересные результаты о квантовых аналогах некоторых более сильных сложностных классов (не описанных в части I).

Задача 8.4. Постройте квантовые схемы полиномиального размера в базисе из операторов на двух q-битах, которые выполняют следующие действия:

а) для заданного числа q, $1 \leqslant q \leqslant 2^n$, записанного n двоичными цифрами, преобразовать состояние $|0^n\rangle$ в состояние $|\psi_n(q)\rangle = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} |j\rangle$;

б) преобразовать $|q-1, 0^n\rangle$ в $|q-1\rangle \otimes |\psi_n(q)\rangle$, считая, что q записано n двоичными цифрами;

в) выполнить преобразование Фурье на группе $\mathbb{Z}_k$ при $k = 2^n$:

$$U_k |x\rangle = \frac{1}{\sqrt{k}} \sum_{y=0}^{k-1} \exp\left(2\pi i \frac{xy}{k}\right) |y\rangle,$$

считая, что x и y записаны n двоичными цифрами; (в этом случае найдите схему размера $O(n^2)$).

9. Квантовые вероятности

Перейдём теперь к обсуждению некоторых «физических» аспектов квантовых вычислений. Пусть система из n q-битов находится в состоянии $|\psi\rangle = \sum_x c_x |x\rangle$. Коэффициенты разложения по выделенному базису (классических состояний) называются амплитудами. Квадрат модуля амплитуды $|c_x|^2$ равен вероятности обнаружить систему в состоянии x (сравните с (8.1)). Другими словами, при *измерении* состояния этой квантовой системы будут получаться классические состояния, распределённые как $|c_x|^2$.

Определённая формулой (8.1) величина обладает основными свойствами обычной вероятности. Тот факт, что квадрат модуля амплитуды — это вероятность наблюдения системы в состоянии x, согласуется с тем, что физические состояния в квантовой механике соответствуют векторам единичной длины, а преобразования этих состояний не меняют длины, т. е. унитарны. Действительно, $\langle\psi|\psi\rangle = \sum_x |c_x|^2 = 1$ (сумма вероятностей равна 1), а применение физически реализуемого оператора должно сохранять это соотношение, т. е. должно быть унитарным.

Формулы (8.1) достаточно для определения квантового вычисления и класса BQP. Однако есть вопросы, в которых это определение оказывается неудобным или неприменимым. Два основных примера: измеряющие операторы и алгоритмы, построенные на их основе, и задача построения надёжных квантовых схем из ненадёжных элементов (исправление ошибок).

Поэтому мы построим определение квантовой вероятности, в котором обобщается как то, что мы наблюдаем (состояние системы), так и результат наблюдения. К этому общему определению мы придём, рассматривая ряд примеров. Для начала перепишем уже полученное выражение для вероятности в виде

$$|c_x|^2 = |\langle\psi|x\rangle|^2 = \langle\psi\,\overbrace{|x\rangle\langle x|}^{\Pi_x}\,\psi\rangle,$$

где через Π_x обозначен проектор на подпространство, порождённое $|x\rangle$.

Чтобы сделать следующий шаг к общему определению квантовой вероятности, подсчитаем вероятность того, что первые m битов имеют заданное значение $y = (y_1, \ldots, y_m)$. Для этого представим состояние в виде двух блоков $x = \overset{m}{\boxed{y}}\overset{n-m}{\boxed{z}}$. Имеем

$$\mathbf{P}(|\psi\rangle, y) = \sum_z \mathbf{P}(|\psi\rangle, (y,z)) = \sum_z \langle\psi|\,|y,z\rangle\langle y,z|\,|\psi\rangle = \langle\psi|\,|y\rangle\langle y|\otimes I\,|\psi\rangle = \\ = \langle\psi|\Pi_{\mathcal{M}}|\psi\rangle. \quad (9.1)$$

$\Pi_{\mathcal{M}}$ здесь обозначает проектор на подпространство $\mathcal{M} = |y\rangle\otimes\mathcal{B}^{\otimes(n-m)}$. Формула (9.1) задаёт определение квантовой вероятности и в том случае, когда $\mathcal{M}$ — произвольное подпространство.

В этом случае проектор на подпространство $\mathcal{M} \subseteq \mathcal{N}$ определяется как $\Pi_{\mathcal{M}} = \sum_j |e_j\rangle\langle e_j|$, где e_j пробегают любой ортонормированный базис $\mathcal{M}$.

Замечание 9.1. Величина $\sum_z \left|\langle F(x), z|\; U\; |x, 0^{N-n}\rangle\right|^2$, которая используется в определении вычисления функции из $\mathbb{B}^n$ в $\mathbb{B}^m$ квантовой схемой (см. с. 68), равна $\mathbf{P}\Big(U|x,0^{N-n}\rangle, \mathcal{M}\Big)$, где $\mathcal{M} = |F(x)\rangle\otimes\mathcal{B}^{N-m}$. Ещё раз напомним смысл этого определения: схема $U = U_L \cdot \ldots \cdot U_2U_1$ вычисляет $F\colon \mathbb{B}^n \to \mathbb{B}^m$, если для любого x вероятность наблюдения правильного ответа $F(x)$ после применения схемы к начальному состоянию $U|x,0^{N-n}\rangle$ больше $1-\varepsilon$.

Проекторы не являются физически реализуемыми операторами, точнее говоря, они не описывают переход от одного состояния системы к другому за определённый промежуток времени. Такой переход описывается унитарными операторами. Всё же, допуская некоторую вольность, проекторам можно придать физический смысл. Проектор выделяет часть состояний системы из всех возможных. Представьте фильтр (устройство, а не теоретико-множественное понятие), который пропускает только системы в состояниях из $\mathcal{M}$. Если на такой фильтр подать систему в состоянии $|\psi\rangle$, то сквозь фильтр пройдёт система в состоянии $|\xi\rangle = \Pi_{\mathcal{M}}|\psi\rangle$. Суммарная вероятность выделенных состояний, вообще говоря, меньше 1, она равна $p = \langle\xi|\xi\rangle = \langle\psi|\Pi_{\mathcal{M}}|\psi\rangle$. Число $1-p$ определяет вероятность того, что система сквозь фильтр не пройдёт.

Сравним свойства классической и квантовой вероятности.

Классическая вероятность	**Квантовая вероятность**
Определение	
Событие $M \subseteq N$ — подмножество некоторого конечного множества.	Событие $\mathcal{M}$ — подпространство некоторого конечномерного унитарного пространства $\mathcal{N}$.
Распределение вероятностей задаётся функцией $w\colon N \to \mathbb{R}$ со свойствами а) $\sum w_j = 1$; б) $w_j \geqslant 0$.	Распределение вероятностей задаётся вектором состояния $|\psi\rangle$, $\langle\psi|\psi\rangle = 1$.
Вероятность: $\mathbf{Pr}(w, M) = \sum_{j\in M} w_j$.	Вероятность: $\mathbf{P}(|\psi\rangle, \mathcal{M}) = \langle\psi|\Pi_{\mathcal{M}}|\psi\rangle$.
Свойства	
1. Если $M_1 \cap M_2 = \varnothing$, то $\mathbf{Pr}(w, M_1 \cup M_2) =$ $= \mathbf{Pr}(w, M_1) + \mathbf{Pr}(w, M_2)$.	1^{q}. Если $\mathcal{M}_1 \perp \mathcal{M}_2$, то $\mathbf{P}(|\psi\rangle, \mathcal{M}_1 \oplus \mathcal{M}_2) =$ $\mathbf{P}(|\psi\rangle, \mathcal{M}_1) + \mathbf{P}(|\psi\rangle, \mathcal{M}_2)$.
2. (в общем случае) $\mathbf{Pr}(w, M_1 \cup M_2) = \mathbf{Pr}(w, M_1) +$ $+ \mathbf{Pr}(w, M_2) - \mathbf{Pr}(w, M_1 \cap M_2)$.	2^{q}. Если $\Pi_{\mathcal{M}_1}\Pi_{\mathcal{M}_2} = \Pi_{\mathcal{M}_2}\Pi_{\mathcal{M}_1}$, то $\mathbf{P}(|\psi\rangle, \mathcal{M}_1 + \mathcal{M}_2) = \mathbf{P}(|\psi\rangle, \mathcal{M}_1) +$ $+ \mathbf{P}(|\psi\rangle, \mathcal{M}_2) - \mathbf{P}(|\psi\rangle, \mathcal{M}_1 \cap \mathcal{M}_2)$.

Заметим, что условие $\mathcal{M}_1 \perp \mathcal{M}_2$ эквивалентно условию $\Pi_{\mathcal{M}_1}\Pi_{\mathcal{M}_2} = \Pi_{\mathcal{M}_2}\Pi_{\mathcal{M}_1} = 0$.

Если есть два неортогональных подпространства с пустым пересечением, то квантовая вероятность необязательно аддитивна. Приведем простой пример, когда $\mathbf{P}(|\xi\rangle, \mathcal{M}_1 \oplus \mathcal{M}_2) \neq \mathbf{P}(|\xi\rangle, \mathcal{M}_1) + \mathbf{P}(|\xi\rangle, \mathcal{M}_2)$.

Пусть $|\xi\rangle = |0\rangle$, $\mathcal{M}_1 = \mathbb{C}(|0\rangle)$ (линейное подпространство, порожденное вектором $|0\rangle$), $\mathcal{M}_2 = \mathbb{C}(|\eta\rangle)$, причем $\langle\xi|\eta\rangle$ близко к 1. Тогда

$$1 = \mathbf{P}(|\xi\rangle, \mathcal{M}_1 \oplus \mathcal{M}_2) \;\neq\; \mathbf{P}(|\xi\rangle, \mathcal{M}_1) + \mathbf{P}(|\xi\rangle, \mathcal{M}_2) \approx 1 + 1.$$

Итак, мы определили в наиболее общем виде то, **что** мы измеряем. Теперь нужно обобщить то, **над чем** проводится измерение. В результате получим определение вероятности, обобщающее как классическую, так и квантовую вероятность.

Рассмотрим распределение вероятностей на конечном множестве квантовых состояний $\{|\xi_1\rangle, \ldots, |\xi_s\rangle\}$. Вероятность состояния $|\xi_j\rangle$ обозначим p_j, очевидно, что $\sum_j p_j = 1$. Подсчитаем вероятность наблюдения состояния в подпространстве $\mathcal{M}$:

$$\sum_k p_k\,\mathbf{P}(|\xi_k\rangle,\mathcal{M}) = \sum_k p_k\,\langle\xi_k|\Pi_{\mathcal{M}}|\xi_k\rangle = \sum_k p_k\ \mathrm{Tr}(|\xi_k\rangle\langle\xi_k|\Pi_{\mathcal{M}}) = \\ = \mathrm{Tr}(\rho\Pi_{\mathcal{M}}), \quad (9.2)$$

где через ρ обозначена *матрица плотности*[12)] $\rho = \sum_k p_k|\xi_k\rangle\langle\xi_k|$. Последнее выражение в (9.2) и примем за общее определение вероятности.

Задача 9.1. Докажите, что операторы вида $\rho = \sum_k p_k|\xi_k\rangle\langle\xi_k|$ — это в точности эрмитовы неотрицательно определённые операторы со следом 1, т. е. операторы, удовлетворяющие условиям:

$$1)\ \rho = \rho^\dagger; \qquad 2)\ \forall\,|\eta\rangle\ \langle\eta|\rho|\eta\rangle \geqslant 0; \qquad 3)\ \mathrm{Tr}\,\rho = 1.$$

В дальнейшем под *матрицей плотности* понимается любой оператор с этими свойствами.

Рассуждение о «распределении вероятностей на квантовых состояниях» носило вспомогательный характер. Задача состоит в том, чтобы обобщить понятие квантового состояния так, чтобы оно включало в себя классические распределения вероятностей. Полученный нами ответ (последнее выражение в (9.2)) зависит лишь от матрицы плотности, поэтому мы можем постулировать, что обобщённые квантовые состояния и матрицы плотности — это одно и то же (такая аксиома не противоречит физическим наблюдениям). Если состояние задается одним вектором ($\rho = |\xi\rangle\langle\xi|$), то оно называется *чистым*, если состояние задается общей матрицей плотности, то оно называется *смешанным*.

Определение 9.1. Для квантового состояния, задаваемого матрицей плотности ρ, и подпространства $\mathcal{M}$ вероятность «события» $\mathcal{M}$ равна $\mathbf{P}(\rho,\mathcal{M}) = \mathrm{Tr}(\rho\Pi_{\mathcal{M}})$.

Диагональные матрицы $\rho = \sum_j w_j|j\rangle\langle j|$ соответствуют классическим распределениям вероятностей на множестве базисных векторов. Это означает, что при вычислении вероятности по общей квантовой формуле для диагональной матрицы и координатного подпространства $\mathcal{M}$ (натянутого на часть базисных векторов $|a\rangle$: $a \in M$) получается то же, что и при вычислении по обычной классической формуле: $\mathbf{P}(\rho,\mathcal{M}) = \mathbf{Pr}(w, M)$.

[12)] В действительности это оператор, а не матрица, однако название «матрица плотности» уже стало традиционным. Впрочем, в дальнейшем мы часто будем иметь в виду именно матрицу, т. е. оператор, записанный в выделенном базисе.

Продолжим сравнение свойств классической и квантовой вероятности, под последней мы будем теперь понимать общее определение с использованием матриц плотности. (Свойства 1^q и 2^q остаются в силе).

Классическая вероятность	**Квантовая вероятность**
Свойства	
3. На множестве $N = N_1 \times N_2$ задано распределение вероятностей вида $w_{jk} = w_j^{(1)} w_k^{(2)}$. Имеется два множества исходов $M_1 \subseteq N_1$, $M_2 \subseteq N_2$. Тогда вероятности перемножаются: $\mathbf{Pr}(w, M_1 \times M_2) = \mathbf{Pr}(w^{(1)}, M_1)\,\mathbf{Pr}(w^{(2)}, M_2)$.	3^q. На пространстве $\mathcal{N} = \mathcal{N}_1 \otimes \mathcal{N}_2$ задана матрица плотности вида $\rho_1 \otimes \rho_2$. Имеется два подпространства $\mathcal{M}_1 \subseteq \mathcal{N}_1$, $\mathcal{M}_2 \subseteq \mathcal{N}_2$. Тогда вероятности также перемножаются: $\mathbf{P}(\rho_1 \otimes \rho_2, \mathcal{M}_1 \otimes \mathcal{M}_2) = \mathbf{P}(\rho_1, \mathcal{M}_1)\,\mathbf{P}(\rho_2, \mathcal{M}_2)$.
4. Имеется совместное распределение на множестве $N_1 \times N_2$. Интересующее нас событие не зависит от исхода во втором множестве: $M = M_1 \times N_2$. Вероятность такого события выражается через проекцию распределения на первое множество: $\mathbf{Pr}(w, M_1 \times N_2) = \mathbf{Pr}(w', M_1)$, где $w'_j = \sum_k w_{jk}$.	4^q. В квантовом случае ограничение на одну из подсистем задается операцией взятия частичного следа (см. ниже). Поэтому, даже если исходное состояние было чистым, полученное состояние подсистемы может оказаться смешанным: $\mathbf{P}(\rho, \mathcal{M}_1 \otimes \mathcal{N}_2) = \mathbf{P}(\mathrm{Tr}_{\mathcal{N}_2}\,\rho, \mathcal{M}_1)$.

Определение 9.2. Пусть $X \in \mathbf{L}(\mathcal{N}_1 \otimes \mathcal{N}_2) = \mathbf{L}(\mathcal{N}_1) \otimes \mathbf{L}(\mathcal{N}_2)$. *Частичный след* оператора X по пространству $\mathcal{N}_2$ определен следующим образом: если $X = \sum_m A_m \otimes B_m$, то $\mathrm{Tr}_{\mathcal{N}_2} X = \sum_m A_m\,(\mathrm{Tr}\,B_m)$.

Докажем, что это определение корректно, т. е. не зависит от выбора слагаемых в представлении $X = \sum_m A_m \otimes B_m$. Для этого зафиксируем некоторые ортонормированные базисы в пространствах $\mathcal{N}_1$, $\mathcal{N}_2$ и выразим частичный след через матричные элементы $X_{jj'kk'} = \langle j, j'|X|k, k'\rangle$. Пусть $A_m = \sum\limits_{j,k} a^m_{jk}|j\rangle\langle k|$ и $B_m = \sum\limits_{j',k'} b^m_{j'k'}|j'\rangle\langle k'|$. Тогда

$$X = \sum_{j,j',k,k'} X_{jj'kk'}|j,j'\rangle\langle k,k'| = \sum_m A_m \otimes B_m = \sum_{j,j'k,k',m} a^m_{jk} b^m_{j'k'}|j,j'\rangle\langle k,k'|$$

и частичный след равен

$$\mathrm{Tr}_{\mathcal{N}_2} X = \sum_m \sum_{j,k} a^m_{jk} \Bigl(\sum_l b^m_{ll}\Bigr)|j\rangle\langle k| = \sum_{j,k}\sum_l X_{jlkl}|j\rangle\langle k|.$$

Рассмотрим пример, когда взятие частичного следа от матрицы плотности, соответствующей чистому состоянию, приводит к матрице плотности, соответствующей смешанному состоянию.

Пусть $\mathcal{N}_1 = \mathcal{N}_2 = \mathcal{B}$, а $\rho = |\psi\rangle\langle\psi|$, где $|\psi\rangle = \frac{1}{\sqrt{2}}(|0,0\rangle + |1,1\rangle)$. В этом случае $\rho = \frac{1}{2}\sum_{a,b} |a,a\rangle\langle b,b|$, поэтому получаем

$$\mathrm{Tr}_{\mathcal{N}_2}\,\rho = \frac{1}{2}\sum_a |a\rangle\langle a| = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}.$$

Эта матрица соответствует смешанному состоянию (чистым состояниям соответствуют матрицы ранга 1). Более того, это смешанное состояние эквивалентно классическому распределению вероятностей: 0 и 1 имеют вероятности, равные 1/2. Таким образом, отбрасывание второго q-бита приводит к чисто классическому распределению вероятностей на первом q-бите.

Утверждение 9.1. *Любое смешанное состояние $\rho \in \mathbf{L}(\mathcal{N})$ представимо как частичный след $\mathrm{Tr}_{\mathcal{F}}(|\psi\rangle\langle\psi|)$ от чистого состояния бо́льшей системы, $|\psi\rangle \in \mathcal{N} \otimes \mathcal{F}$, причём можно считать, что $\dim \mathcal{F} \leqslant \dim \mathcal{N}$.*

Доказательство. Полагаем $\mathcal{F} = \mathcal{N}^*$. Так как ρ неотрицательно определена, то существует $\sqrt{\rho} \in \mathbf{L}(\mathcal{N}) = \mathcal{N} \otimes \mathcal{N}^*$. Докажем, что для $|\psi\rangle = \sqrt{\rho}$ выполнено искомое, т. е. $\rho = \mathrm{Tr}_{\mathcal{N}^*}(|\sqrt{\rho}\rangle\langle\sqrt{\rho}|)$.

Поскольку ρ эрмитова, она диагонализуется в некотором ортонормированном базисе $\{|\xi_j\rangle\}$. Тогда $\rho = \sum_j p_j |\xi_j\rangle\langle\xi_j|$, а $|\psi\rangle = \sqrt{\rho} = \sum_j \sqrt{p_j}|\xi_j\rangle\langle\xi_j|$, и $|\psi\rangle\langle\psi| = \sum_{jk}\sqrt{p_j p_k}(|\xi_j\rangle \otimes \langle\xi_j|)(|\xi_k\rangle \otimes \langle\xi_k|)$. Вклад в частичный след вносят лишь слагаемые с $j = k$. Поэтому $\mathrm{Tr}_{\mathcal{N}^*}(|\psi\rangle\langle\psi|) = \sum_j p_j|\xi_j\rangle\langle\xi_j| = \rho$. □

Задача 9.2. Пусть имеется чистое состояние $|\psi\rangle \in \mathcal{N} \otimes \mathcal{F}$. Докажите, что существует так называемое *разложение Шмидта*:

$$|\psi\rangle = \sum_j \lambda_j |\xi_j\rangle \otimes |\eta_j\rangle,$$

где $0 < \lambda_j \leqslant 1$, а множества векторов $\{|\xi_j\rangle\} \subset \mathcal{N}$ и $\{|\eta_j\rangle\} \subset \mathcal{F}$ являются ортонормированными.

Заметим, что числа λ_j^2 — это ненулевые собственные числа частичных следов $\rho = \mathrm{Tr}_{\mathcal{F}}(|\psi\rangle\langle\psi|)$ и $\rho' = \mathrm{Tr}_{\mathcal{N}}(|\psi\rangle\langle\psi|)$. (Таким образом, ненулевые собственные числа ρ и ρ' совпадают.) Отсюда следует, что взятие частичного следа от чистого состояния приводит к чистому состоянию тогда и только тогда, когда исходное состояние разложимо: $|\psi\rangle = |\xi\rangle \otimes |\eta\rangle$.

Задача 9.3. Пусть $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{N}\otimes\mathcal{F}$ — два чистых состояния, таких что $\mathrm{Tr}_{\mathcal{F}}(|\psi_1\rangle\langle\psi_1|) = \mathrm{Tr}_{\mathcal{F}}(|\psi_2\rangle\langle\psi_2|)$. Докажите, что $|\psi_2\rangle = (I_{\mathcal{N}} \otimes U)|\psi_1\rangle$ для некоторого унитарного оператора U на пространстве $\mathcal{F}$.

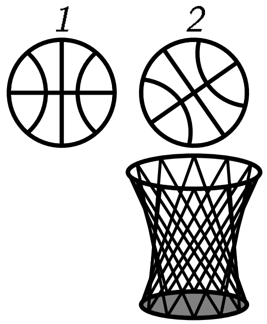
10. Физически реализуемые преобразования матриц плотности

Теперь опишем, какие преобразования матриц плотности допустимы с физической точки зрения.

1. Унитарный оператор переводит матрицу плотности чистого состояния $\rho = |\xi\rangle\langle\xi|$ в матрицу $\rho' = U|\xi\rangle\langle\xi|U^\dagger$. Естественно считать (по линейности), что такой же формулой задается и действие унитарного оператора на произвольные матрицы плотности:
$$\rho \overset{U}{\mapsto} U\rho U^\dagger.$$
2. Второй тип преобразования состоит во взятии частичного следа. Если есть $\rho \in \mathbf{L}(\mathcal{N}\otimes\mathcal{F})$, то *отбрасывание второй системы* задается преобразованием
$$\rho \mapsto \mathrm{Tr}_{\mathcal{F}}\,\rho.$$
3. Вспомним, что нам ещё бывает нужно брать напрокат q-биты в состоянии 0. Пусть есть состояние $\rho \in \mathbf{L}(\mathcal{B}^{\otimes n})$. Рассмотрим изометрическое (сохраняющее скалярные произведения) вложение $V\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes N}$ в пространство большей размерности, задаваемое формулой $|\xi\rangle \overset{V}{\mapsto} |\xi\rangle \otimes |0^{N-n}\rangle$. Матрица плотности ρ при этом преобразуются в $\rho \otimes |0^{N-n}\rangle\langle 0^{N-n}|$. Для любого изометрического вложения V по аналогии полагаем
$$\rho \overset{V}{\mapsto} V\rho V^\dagger.$$

Будем считать, что *физически реализуемые преобразования матриц плотности* есть в точности композиции любого числа преобразований типа 2 и 3 (случай 1 — частный случай преобразования типа 3).

Задача 10.1. Докажите, что любое физически реализуемое преобразование матриц плотности имеет вид $\rho \mapsto \mathrm{Tr}_{\mathcal{F}}(V\rho V^\dagger)$, где $V\colon \mathcal{N} \to \mathcal{N}\otimes\mathcal{F}$ — изометрическое вложение.



Операция взятия частичного следа означает забывание (отбрасывание) одной из подсистем. Покажем, что такая интерпретация является разумной, а именно, дальнейшая судьба отброшенной системы не влияет на величины, характеризующие оставшуюся систему. Возьмём систему, состоящую из двух подсистем и находящуюся в некотором состоянии $\rho \in \mathbf{L}(\mathcal{N}\otimes\mathcal{F})$. Если мы выбрасываем вторую систему (в мусорную корзину), то она будет подвергаться неконтролируемым воздействиям. Пусть мы применили какой-то оператор U к первой системе. Получили состояние $\gamma = (U \otimes Y)\rho(U \otimes Y)^\dagger$, где Y — произвольный унитарный оператор (действие мусорной корзины на мусор). Если мы хотим найти вероятность для подпространства $\mathcal{M} \subseteq \mathcal{N}$, относящегося к первой системе (мусор нас не интересует), то она не зависит от Y и равна

$$\mathbf{P}(\gamma, \mathcal{M} \otimes \mathcal{F}) = \mathbf{P}(\operatorname{Tr}_{\mathcal{F}} \gamma, \mathcal{M}) = \mathbf{P}\left(U(\operatorname{Tr}_{\mathcal{F}} \rho)U^\dagger, \mathcal{M}\right).$$

Здесь первое равенство — это свойство 4^{q} квантовой вероятности, а второе равенство — новое свойство:

$$\operatorname{Tr}_{\mathcal{F}}\left((U \otimes Y)\rho(U \otimes Y)^\dagger\right) = U\left(\operatorname{Tr}_{\mathcal{F}} \rho\right)U^\dagger. \tag{10.1}$$

Задача 10.2. Докажите тождество (10.1) для частичного следа.

Задача 10.3. Запишем линейный оператор $T\colon \mathbf{L}(\mathcal{N}) \to \mathbf{L}(\mathcal{M})$ в координатном виде:

$$T(|j\rangle\langle k|) = \sum_{j',k'} T_{(j'j)(k'k)} |j'\rangle\langle k'|.$$

Докажите, что физическая реализуемость T эквивалентна набору из трёх условий:

а) $\sum_{k'} T_{(k'j)(k'k)} = \delta_{jk}$ (символ Кронекера);

б) $T^*_{(j'j)(k'k)} = T_{(k'k)(j'j)}$;

в) $T_{(j'j)(k'k)}$ — неотрицательная матрица (по парам индексов).

Задача 10.4. Докажите, что линейный оператор $T\colon \mathbf{L}(\mathcal{N}) \to \mathbf{L}(\mathcal{M})$ является физически реализуемым преобразованием матриц плотности тогда и только тогда, когда выполнены три условия:

а) $\operatorname{Tr}(TX) = \operatorname{Tr} X$ для любого $X \in \mathbf{L}(\mathcal{N})$;

б) $(TX)^\dagger = TX^\dagger$ для любого $X \in \mathbf{L}(\mathcal{N})$;

в) T является вполне положительным преобразованием. А именно, для любого пространства $\mathcal{G}$ преобразование $T \otimes I_{\mathbf{L}(\mathcal{G})}\colon \mathbf{L}(\mathcal{N} \otimes \mathcal{G}) \to \mathbf{L}(\mathcal{M} \otimes \mathcal{G})$ отображает неотрицательные операторы в неотрицательные.

10.1. Подсчёт вероятностей для квантового вычисления. Теперь, имея общие определения квантовой вероятности и физически реализуемого преобразования матриц плотности, можно вычислять вероятности, входящие в определение квантового вычисления, двумя способами. Пусть мы использовали при вычислениях дополнительную подсистему. После того, как она стала нам не нужна, мы можем выбросить её в мусорную корзину, а при подсчёте вероятности взять частичный след по пространству состояний дополнительной подсистемы.

А можно тянуть весь этот мусор до самого конца и считать вероятность для событий вида $\mathcal{M}_1 \otimes \mathcal{N}_2$ (раз уж мы перестали использовать вторую подсистему, то никакие детали её состояния нам не важны — нам безразлично, что именно произойдёт с использованной подсистемой в мусорной корзине). Как уже говорилось, эти вероятности равны: $\mathbf{P}(\rho, \mathcal{M}_1 \otimes \mathcal{N}_2) = \mathbf{P}(\operatorname{Tr}_{\mathcal{N}_2} \rho, \mathcal{M}_1)$.

Замечание 10.1. Нетрудно определить более общую модель квантового вычисления, в которой элементарными действиями являются подходящие преобразования матриц плотности общего вида (не обязательно унитарные операторы). Такая модель более адекватна физической ситуации, когда квантовый компьютер взаимодействует с «окружающей средой». С вычислительной точки зрения новая модель эквивалентна стандартной (если в обоих случаях используется полный базис). Однако в модели с общими преобразованиями матриц плотности возможно более естественное определение подпрограммы для квантового вычисления, поскольку результат работы квантовой схемы — *вероятностная функция*. Здесь мы не будем давать этого определения и отсылаем заинтересованного читателя к [20].

10.2. Потеря когерентности (decoherence). Рассмотрим в качестве примера преобразование матриц плотности, которое «забывает» внедиагональные элементы:

$$\rho = \sum_{j,k} \rho_{jk} |j\rangle\langle k| \overset{D}{\mapsto} \sum_k \rho_{kk} |k\rangle\langle k|.$$

Покажем, что оно «физически реализуемо», т.е. может быть построено композицией описанных выше преобразований. Будем строить это преобразование в три шага. Вначале добавим нулевые биты:

$$\rho \mapsto \rho \otimes |0\rangle\langle 0|.$$

Затем скопируем обратимым образом исходные биты в добавленные. Обратимое копирование задается оператором $\Lambda(\sigma^x)\colon |a,b\rangle \mapsto |a, a \oplus b\rangle$. Получаем

$$\rho \otimes |0\rangle\langle 0| \overset{\Lambda(\sigma^x)}{\mapsto} \sum_{j,k} \rho_{jk} |j,j\rangle\langle k,k|.$$

А теперь возьмём частичный след по добавленным битам. Получим диагональную матрицу

$$\sum_k \rho_{kk} |k\rangle\langle k|.$$

Предостережение. Рассмотренная нами «операция копирования»

$$|j\rangle \mapsto |j,j\rangle, \qquad \sum_{j,k} \rho_{jk} |j\rangle\langle k| \mapsto \sum_{j,k} \rho_{jk} |j,j\rangle\langle k,k|$$

(композиция первых двух преобразований) на самом деле копирует только базисные состояния. Заметим, что копирование *произвольного* квантового состояния $|\xi\rangle \mapsto |\xi\rangle \otimes |\xi\rangle$ является нелинейным оператором, поэтому не может быть реализовано физически. В дальнейшем копирование всегда будет определяться относительно некоторого базиса.

Замечание 10.2. Рассмотренное преобразование переводит любое состояние в классическое (с диагональной матрицей плотности), используя копирование битов. Это можно интерпретировать так: *если постоянно наблюдать за системой* (*делать копии*), *то система будет вести себя как классическая.* В случае одного q-бита то же самое преобразование (обнуление внедиагональных элементов) можно получить, если применить оператор σ^z с вероятностью $1/2$:

$$\rho \mapsto \frac{1}{2}\rho + \frac{1}{2}\sigma^z \rho \sigma^z .$$

Подобный процесс называется случайным сбоем фазы: состояние $|1\rangle$ домножается на фазовый множитель -1 с вероятностью $1/2$. Таким образом, *сбой фазы также приводит к тому, что система ведет себя как классическая.*

Задача 10.5. Пусть имеется физически реализуемое преобразование $T\colon \mathbf{L}(\mathcal{N}) \to \mathbf{L}(\mathcal{N} \otimes \mathcal{F})$ со следующим свойством: $\mathrm{Tr}_{\mathcal{F}}(T\rho) = \rho$ для любого чистого состояния ρ. Докажите, что тогда $TX = X \otimes \gamma$ (для любого оператора X), где γ — некоторая фиксированная матрица плотности на пространстве $\mathcal{F}$.

Таким образом, нельзя получить никакой информации о неизвестном состоянии ρ, не возмущая это состояние.

10.3. Измерение. При описании квантовых алгоритмов часто бывает естественно считать, что наряду с квантовым вычислительным устройством используется и классическое. Основной механизм взаимодействия между квантовой и классической частями состоит в *измерении* квантовых регистров, дающем классический результат.

Рассмотрим систему, состоящую из двух частей, — квантовой ($\mathcal{N}$) и классической ($\mathcal{K}$). По классическим координатам матрица плотности диагональна:

$$\rho = \sum_{j,k,l} \rho_{jkll} \left(|j\rangle\langle k|\right) \otimes \left(|l\rangle\langle l|\right) = \sum_{l} w_l \gamma^{(l)} \otimes \left(|l\rangle\langle l|\right),$$

где $w_l = \sum_j \rho_{jjll}$ — вероятность иметь классическое состояние l, а оператор $\gamma^{(l)} = w_l^{-1} \sum_{j,k} \rho_{jkll}$ обладает всеми свойствами матрицы плотности. Таким образом, квантово-классическое состояние всегда разложимо на «условные» (по аналогии с условными вероятностями) матрицы плотности $\gamma^{(l)}$. Будем использовать для такого случая специальное обозначение: $\rho = \sum_l w_l \left(\gamma^{(l)}, l\right) = \sum_l \left(w_l \gamma^{(l)}, l\right)$.

Пусть имеется ряд взаимоисключающих возможностей, что выражается разложением пространства состояний в прямую сумму попарно ортогональных подпространств $\mathcal{N} = \bigoplus_{j\in\Omega} \mathcal{L}_j$, где $\Omega = \{1,\ldots,r\}$ — множество возможностей (действительно, если подпространство $\mathcal{L}_1$ ортогонально подпространству $\mathcal{L}_2$, то для любой матрицы плотности $\rho \in \mathbf{D}(\mathcal{L}_1)$ выполняется $\mathbf{P}(\rho,\mathcal{L}_2) = 0$).

Преобразование матриц плотности, которое мы будем называть *измерением*, состоит в том, что для состояний из подпространства $\mathcal{L}_j$ «измеряющий прибор» помещает в классический регистр номер состояния j:

$$\text{если } |\xi\rangle \in \mathcal{L}_j, \text{ то } |\xi\rangle\langle\xi| \mapsto \left(|\xi\rangle\langle\xi|, j\right). \tag{10.2}$$

Хотя измерение отображает пространство $\mathbf{L}(\mathcal{N})$ в $\mathbf{L}(\mathcal{N}\otimes\mathcal{K})$, результат всегда диагонален по второй компоненте. Поэтому можно считать, что измерение отображает $\mathbf{L}(\mathcal{N})$ в $\bigoplus_{j=1}^{r} \mathbf{L}(\mathcal{N})$.

Для $|\xi\rangle \in \mathcal{L}_j$ выполняется равенство $|\xi\rangle\langle\xi| = \Pi_{\mathcal{L}_j} |\xi\rangle\langle\xi| \Pi_{\mathcal{L}_j}$. Поэтому из соображений линейности можно доопределить измерение на всех остальных матрицах плотности

$$\rho \ \mapsto \ \sum_j \left(\Pi_{\mathcal{L}_j} \rho \Pi_{\mathcal{L}_j}, j\right)$$

и прийти к следующему определению.

Определение 10.1. ***(Детерминированным) измерением*** называется преобразование матриц плотности

$$\rho \ \mapsto \ \sum_j \mathbf{P}(\rho,\mathcal{L}_j) \left(\gamma^{(j)}, j\right), \tag{10.3}$$

где $\gamma^{(j)} = \mathbf{P}(\rho,\mathcal{L}_j)^{-1} \times \Pi_{\mathcal{L}_j} \rho \Pi_{\mathcal{L}_j}$.

Можно сказать, что j — это *результат измерения*, $\mathbf{P}(\rho,\mathcal{L}_j)$ — вероятность получить данный результат, а $\gamma^{(j)}$ — состояние измеряемой системы после измерения при условии, что получен результат j. Если мы измеряем чистые состояния, т.е. $\rho = |\xi\rangle\langle\xi|$, то $\gamma^{(j)} = |\eta_j\rangle\langle\eta_j|$, где $|\eta_j\rangle = \dfrac{\Pi_{\mathcal{L}_j}|\xi\rangle}{\sqrt{\mathbf{P}(|\xi\rangle, \mathcal{L}_j)}}$.

Приведём простейший пример измерения. Сделаем две копии бита. Пусть $\Pi_{\mathcal{L}_0} = |0\rangle\langle 0|$, а $\Pi_{\mathcal{L}_1} = |1\rangle\langle 1|$. Тогда

$$\rho = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix} \mapsto (\rho_{00}|0\rangle\langle 0|, 0) + (\rho_{11}|1\rangle\langle 1|, 1).$$

Задача 10.6. «Квантовая телепортация» (см. [21]). Пусть имеются три q-бита: первый из них находится в произвольном (заранее неизвестном) состоянии ρ, второй и третий — в состоянии $|\xi_{0,0}\rangle = \frac{1}{\sqrt{2}}\Big(|0,0\rangle + |1,1\rangle\Big)$. Произведём над первыми двумя q-битами измерение, соответствующее ортогональному разложению

$$\mathcal{B}^{\otimes 2} = \mathbb{C}(|\xi_{00}\rangle) \oplus \mathbb{C}(|\xi_{01}\rangle) \oplus \mathbb{C}(|\xi_{10}\rangle) \oplus \mathbb{C}(|\xi_{11}\rangle),$$

$$\text{где } |\xi_{ab}\rangle = \frac{1}{\sqrt{2}}\sum_c (-1)^{bc}\,|c, c\oplus a\rangle.$$

Покажите, что используя результат измерения и оставшийся третий q-бит, можно восстановить исходное состояние ρ. Запишите всю последовательность действий (измерение и восстановление) в виде квантовой схемы.

Замечание 10.3. Этот процесс можно представлять таким образом. Допустим, что Алиса хочет передать Бобу[13] квантовое состояние ρ по классическому каналу связи (например, по телефону). Оказывается, что это возможно, если Алиса и Боб заранее приготовили состояние $|\xi_{00}\rangle$ и взяли от него по половинке — одному q-биту. Алиса производит измерение и сообщает результат Бобу. Затем Боб переводит свой q-бит в состояние ρ.

11. Измеряющие операторы

Введём особый класс операторов — *измеряющие операторы*. Пусть есть пространство состояний $\mathcal{N}\otimes\mathcal{K}$, причем первый сомножитель разложен в прямую сумму попарно ортогональных подпространств: $\mathcal{N} = \bigoplus_j \mathcal{L}_j$. Тогда всякий оператор вида $W = \sum_j \Pi_{\mathcal{L}_j}\otimes U_j$ будем называть *измеряющим*.

[13] Эти два персонажа встречаются практически в любой статье по квантовой теории информации.

Чтобы оправдать такое название, рассмотрим следующий процесс. Пусть имеется некоторое состояние, описываемое матрицей плотности $\rho \in \mathbf{L}(\mathcal{N})$. Подсоединим *прибор*; совместное состояние системы и прибора описывается матрицей плотности $\rho\otimes|0^m\rangle\langle 0^m|$ (мы считаем, что во втором сомножителе, описывающем прибор, есть выделенный базис, например, что это $\mathcal{B}^{\otimes n}$).

Теперь применяем измеряющий оператор W. Получаем состояние

$$W\Big(\rho\otimes|0^m\rangle\langle 0^m|\Big)W^\dagger = \sum_j \Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}\otimes U_j|0\rangle\langle 0|U_j^\dagger$$

(здесь мы воспользовались характеристическими свойствами проектора $\Pi^\dagger = \Pi$, $\Pi^2 = \Pi$).

И последнее действие: прибор становится классическим. Это означает, что матрица диагонализуется по второму сомножителю. Посмотрим, во что переходят при этом вторые сомножители в написанной сумме:

$$\gamma_j = |\xi_j\rangle\langle\xi_j| \longmapsto \sum_k |\langle k|\xi_j\rangle|^2\,|k\rangle\langle k| \qquad \Big(\text{так как } (\gamma_j)\big|_{kk} = \langle k|\gamma_j|k\rangle\Big).$$

Теперь запишем получившийся результат:

$$\sum_j\sum_k\Big(\Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}\,|\langle k|U_j|0\rangle|^2,\,k\Big) = \sum_j\sum_k \mathbf{P}(k\big|j)\,\big(\Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j},k\big),$$

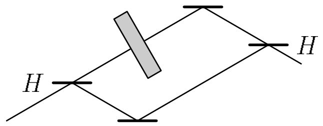
где введены *условные вероятности* $\mathbf{P}(k\big|j) = |\langle k|U_j|0\rangle|^2$. Заметим, что для измерения (как оно было определено в предыдущем разделе) $\mathbf{P}(k\big|j) = \delta_{kj}$, поэтому только что описанный процесс можно назвать «вероятностным измерением». Введённые таким образом квантовые условные вероятности ведут себя как обычные, если рассматриваются произведения измеряющих операторов, построенных на одном и том же ортогональном разложении пространства состояний (см. ниже).

Приведем примеры измеряющих операторов.

1. Оператор $\Lambda(U) = \Pi_0\otimes I + \Pi_1\otimes U$, действующий на пространстве $\mathcal{B}\otimes\mathcal{N}$, — измеряющий.

1′. Нетривиально, что он измеряющий и по второй компоненте. Поскольку U — унитарный оператор, его можно разложить в сумму проекторов на собственные подпространства: $U = \sum_j \lambda_j\Pi_{\mathcal{L}_j}$, $|\lambda_j| = 1$. Тогда $\Lambda(U) = \sum_j(\Pi_0 + \lambda_j\Pi_1)\otimes\Pi_{\mathcal{L}_j} = \sum_j \begin{pmatrix} 1 & 0 \\ 0 & \lambda_j\end{pmatrix}\otimes\Pi_{\mathcal{L}_j}$.

В этом случае условные вероятности равны $\mathbf{P}(0\big|j) = 1$ и $\mathbf{P}(1\big|j) = 0$, поэтому такой оператор, хотя и является измеряющим по определению, фактически ничего не измеряет.



Замечание для физиков. Пусть U — оператор фазового сдвига света при прохождении сквозь стеклянную пластинку. Мы можем разделить луч света на два, пропустив его через полупрозрачное зеркало, затем один из полученных лучей пропустить через стеклянную пластинку, а затем заставить полученные в результате лучи интерферировать. По картине интерференции можно узнать фазовый сдвиг.

Математический вариант предыдущего примера. Аналогом полупрозрачного зеркала будет служить оператор $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Как видно из приведенной выше картинки, его нужно применить в начале и в конце. Строго говоря, рассмотрим оператор

$$\Xi(U) = (H \otimes I)\Lambda(U)(H \otimes I).$$

Если начальный вектор имеет вид $|\psi\rangle = |\eta\rangle \otimes |\xi\rangle$ ($|\xi\rangle \in \mathcal{L}_j$), то $\Xi(U)|\psi\rangle = |\eta'\rangle \otimes |\xi\rangle$, где

$$|\eta'\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \lambda_j \end{pmatrix} \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |\eta\rangle = \frac{1}{2}\begin{pmatrix} 1+\lambda_j & 1-\lambda_j \\ 1-\lambda_j & 1+\lambda_j \end{pmatrix} |\eta\rangle.$$

Поэтому $\Xi(U) = \sum_j \overbrace{\frac{1}{2}\begin{pmatrix} 1+\lambda_j & 1-\lambda_j \\ 1-\lambda_j & 1+\lambda_j \end{pmatrix}}^{R_j} \otimes \Pi_{\mathcal{L}_j}$. Теперь подсчитаем условные вероятности. Собственные числа унитарного оператора равны по модулю 1, поэтому можно полагать $\lambda_j = e^{2\pi i \varphi_j}$. В результате имеем

$$\mathbf{P}(0|j) = |\langle 0|R_j|0\rangle|^2 = \left|\frac{1+\lambda_j}{2}\right|^2 = \frac{1+\cos(2\pi\varphi)}{2}.$$

В дальнейшем именно с помощью такого оператора мы будем оценивать собственные числа. Для этого придется брать разные биты в качестве первых сомножителей (разные «приборы»). Конечно, следует убедиться, что это корректно (т. е., что вероятности будут перемножаться).

Свойства измеряющих операторов. Мы будем рассматривать измеряющие операторы, соответствующие **одному и тому же** ортогональному разложению $\mathcal{N} = \bigoplus_j \mathcal{L}_j$.

1. Произведение измеряющих операторов — измеряющий оператор. Действительно, пусть есть два измеряющих оператора

$$W^{(1)} = \sum_j R_j^{(1)} \otimes \Pi_{\mathcal{L}_j} \text{ и } W^{(2)} = \sum_j R_j^{(2)} \otimes \Pi_{\mathcal{L}_j}.$$

Поскольку $\Pi_{\mathcal{L}_j}\Pi_{\mathcal{L}_k} \neq 0 \Leftrightarrow j = k$, имеем

$$W^{(2)}W^{(1)} = \sum_j R_j^{(2)} R_j^{(1)} \otimes \Pi_{\mathcal{L}_j}.$$

2. Условные вероятности для произведения измеряющих «разными приборами» операторов перемножаются. Более точно, пусть $R^{(1)} = \widetilde{R}^1 \otimes I$, а $R^{(2)} = I \otimes \widetilde{R}^2$. Тогда $\mathbf{P}(k_1, k_2 \,|\, j) = \mathbf{P}_1(k_1 \,|\, j)\mathbf{P}_2(k_2 \,|\, j)$. Это равенство следует непосредственно из определения условных вероятностей и из очевидного тождества

$$\Big(\langle \xi_1| \otimes \langle \xi_2|\Big)\Big(U_1 \otimes U_2\Big)\Big(|\eta_1\rangle \otimes |\eta_2\rangle\Big) = \langle \xi_1|U_1|\eta_1\rangle\,\langle \xi_2|U_2|\eta_2\rangle.$$

3. Формула полной вероятности. Пусть есть измеряющий оператор $W = \sum R_j \otimes \Pi_{\mathcal{L}_j}$. Если применить его к состоянию $|0\rangle\langle 0| \otimes \rho$, где $\rho \in \mathbf{L}(\mathcal{N})$, то вероятность наблюдения состояния k можно записать в виде:

$$\mathbf{P}\Big(W(|0\rangle\langle 0| \otimes \rho)W^\dagger,\, \mathbb{C}(|k\rangle) \otimes \mathcal{N}\Big) = \sum_j \mathbf{P}(k|j)\mathbf{P}(\rho, \mathcal{L}_j).$$

Доказательство. $W\Big(|0\rangle\langle 0| \otimes \rho\Big)W^\dagger = \gamma = \sum_j \Big(R_j|0\rangle\langle 0|R_j^\dagger\Big) \otimes \Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}$.

Ранее было доказано, что $\mathbf{P}\big(\gamma,\, \mathbb{C}(|k\rangle) \otimes \mathcal{N}\big) = \mathbf{P}\big(\mathrm{Tr}_{\mathcal{N}}\,\gamma, \mathbb{C}(|k\rangle)\big)$. Далее,

$$\mathrm{Tr}_{\mathcal{N}}\,\gamma = \sum_j \Big(R_j|0\rangle\langle 0|R_j^\dagger\Big) \times \mathrm{Tr}\left(\Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}\right).$$

Поскольку

$$\mathrm{Tr}\left(\Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}\right) = \mathrm{Tr}\left(\Pi^2_{\mathcal{L}_j}\rho\right) = \mathrm{Tr}\left(\Pi_{\mathcal{L}_j}\rho\right) \stackrel{\mathrm{def}}{=} \mathbf{P}\left(\rho, \mathcal{L}_j\right),$$

получаем искомое выражение $\mathbf{P}\big(\gamma,\, \mathbb{C}(|k\rangle) \otimes \mathcal{N}\big) = \sum_j \mathbf{P}(k|j)\mathbf{P}(\rho, \mathcal{L}_j)$. $\square$

Задача 11.1. Докажите формулу полной вероятности напрямую, не используя взятия частичного следа.

Задача 11.2. «Обратимое измерение». Пусть $W = \sum_{k=1}^{t} \Pi_{\mathcal{L}_k} \otimes V_k$ — измеряющий оператор, $V_k|0\rangle = \sum_{y,z} c_{y,z}(k)|y,z\rangle$. (Имеется в виду, что операторы V_k действуют на $m+s$ q-бит; первые m q-бит (т. е. y) — «полезный результат», остальные s q-бит (т. е. z) — «мусор».) Допустим, что V измеряет некоторую функцию $f: \{1, \ldots, t\} \to \mathbb{B}^m$ с вероятностью ошибки $\leqslant \varepsilon$, т. е.

$$\mathbf{P}\big(f(k)\,|k\big) \stackrel{\mathrm{def}}{=} \sum_z |c_{f(k),z}(k)|^2 \geqslant 1 - \varepsilon.$$

Постройте с использованием W и W^{-1} квантовую схему полиномиального размера, реализующую с точностью $O(\varepsilon^{1/2})$ новый измеряющий оператор

$$U = \sum_{k=1}^{t} \Pi_{\mathcal{L}_k} \otimes Q_{f(k)}\,, \quad \text{где } Q_v|y\rangle = |y \oplus v\rangle.$$

(Разрешается «брать напрокат» дополнительные q-биты.)

12. Быстрые квантовые алгоритмы

Единственное нетривиальное использование квантовых свойств для вычислений, которое мы уже рассмотрели, — это решение универсальной переборной задачи алгоритмом Гровера, изложенным в разделе 8.1. К сожалению, при этом достигается лишь полиномиальное ускорение. Поэтому никаких серьёзных следствий для теории сложности вычислений (типа BQP $\supset$ BPP) алгоритм Гровера не даёт. В настоящее время нет доказательства того, что квантовые вычисления превосходят по скорости классические вероятностные. Но есть косвенные свидетельства в пользу такого утверждения. Первое из них — пример *задачи с оракулом* (т. е. процедурой типа «чёрного ящика»), для которой существует полиномиальный квантовый алгоритм, в то время как любой классический вероятностный алгоритм экспоненциален.[14] Этот пример, построенный Д. Саймоном [42], называется *задачей о скрытой подгруппе в* $(\mathbb{Z}_2)^k$. В дальнейшем мы решим также задачу о скрытой подгруппе в $\mathbb{Z}^k$, обобщающую все результаты из этого раздела.

Задача о скрытой подгруппе. Пусть G — конечная группа, причём задано некоторое представление элементов G двоичными словами. Имеется устройство (*оракул*), вычисляющее функцию $f\colon G \to \mathbb{B}^n$ со следующим свойством:

$$f(x) = f(y) \iff x - y \in D, \tag{12.1}$$

где $D \subseteq G$ — некоторая заранее неизвестная подгруппа. Нужно найти эту подгруппу.

12.1. Задача о скрытой подгруппе в $(\mathbb{Z}_2)^k$.

Мы рассмотрим сформулированную выше задачу в случае $G = (\mathbb{Z}_2)^k$. Элементы этой группы

[14] Следует иметь в виду, что сложность задач с оракулом часто отличается от сложности обычных вычислительных задач. Классический пример — теорема о том, что IP = PSPACE [36, 37]. Оракульный аналог этого утверждения неверен [30]!

можно представлять строками длины k из нулей и единиц; групповая операция — побитовое сложение по модулю 2.

Легко доказать, что нельзя быстро найти «скрытую подгруппу» на классической вероятностной машине. (Классическая машина посылает на вход «чёрного ящика» строки $x_1, \ldots, x_l$ и получает ответы $y_1, \ldots, y_l$. Каждый следующий вопрос x_j зависит от предыдущих ответов $y_1, \ldots, y_{j-1}$ и некоторого случайного числа r.)

Утверждение 12.1. *Пусть $n \geqslant k$. Для любого классического вероятностного алгоритма, делающего не более $2^{k/2}$ обращений к оракулу, существует подгруппа $D \subseteq (\mathbb{Z}_2)^k$ и соответствующая функция $f\colon (\mathbb{Z}_2)^k \to \mathbb{B}^n$, для которой алгоритм ошибается с вероятностью $> 1/3$.*

Доказательство. Для одной и той же подгруппы D существует несколько различных оракулов f. Мы будем считать, что один из них выбирается случайно и равновероятно. (Если алгоритм ошибается с вероятностью $> 1/3$ при случайном оракуле, то он также будет ошибаться с вероятностью $> 1/3$ при каком-нибудь конкретном оракуле.) Случайный оракул обладает следующим свойством: если очередной ответ y_j не совпадает ни с одним из предыдущих ответов $y_1, \ldots, y_{j-1}$, то он равномерно распределён на множестве $\mathbb{B}^n \setminus \{y_1, \ldots, y_{j-1}\}$. Таким образом, случайный оракул эквивалентен устройству с памятью, которое на вопрос x_j выдает наименьшее число $s_j \leqslant j$, такое что $x_j - x_{s_j} \in D$. Классическую машину можно изменить таким образом, что она сама будет производить случайный выбор $y_j \in \mathbb{B}^n \setminus \{y_1, \ldots, y_{j-1}\}$, когда $s_j = j$.

Пусть число вопросов к оракулу равно $l \leqslant 2^{k/2}$. Без уменьшения общности все вопросы различны. В случае $D = \{0\}$ все ответы также различны, то есть $s_j = j$ для всех j. Теперь рассмотрим случай $D = \{0, z\}$, где z выбирается случайно с равномерным распределением на множестве всех ненулевых элементов группы $(\mathbb{Z}_2)^k$. Тогда, независимо от используемого алгоритма, $s_j = j$ с вероятностью $\geqslant 1 - (j-1)/(2^k - 1)$. С вероятностью $\geqslant 1 - l(l-1)/(2(2^k-1)) > 1/2$ это имеет место для всех $j = 1, \ldots, l$. Напомним, что у нас есть два случайных параметра: z и r. Мы можем зафиксировать z таким образом, чтобы вероятность получения ответов $s_j = j$ (для всех j) по-прежнему была больше $1/2$. Посмотрим, что будет делать классическая машина в этом случае. Если она выдает ответ «$D = \{0\}$» с вероятностью $\geqslant 2/3$, положим $D = \{0, z\}$ — тогда выдаваемый ответ будет неверным с вероятностью $> (2/3) \cdot (1/2) = 1/3$. Если же вероятность ответа «$D = \{0\}$» меньше $2/3$, положим $D = \{0\}$. $\square$

Теперь определим квантовый аналог описанного выше устройства. Соответствующий квантовый оракул — это унитарный оператор

$$U\colon |x,y\rangle \mapsto |x,\, y \oplus f(x)\rangle. \tag{12.2}$$

($\oplus$ обозначает побитовое сложение). Заметим, что квантовый оракул допускает линейные комбинации разных вопросов, поэтому его можно использовать более эффективно, чем классический оракул.

Пусть $E = G/D$, а E^* — группа характеров на E, т.е. гомоморфизмов $E \to \mathbf{U}(1)$. В случае $G = (\mathbb{Z}_2)^k$ группу E^* можно охарактеризовать следующим образом:

$$E^* = \{h \in (\mathbb{Z}_2)^k : \forall\, z \in (\mathbb{Z}_2)^k\ (h \cdot z = 0)\},$$

где $h \cdot z$ обозначает скалярное произведение по модулю 2. (Соответствующий h характер имеет вид $z \mapsto (-1)^{h \cdot z}$.) Покажем, как можно породить случайный элемент $h \in E^*$, используя оператор U. Породив достаточно много случайных элементов, мы найдем группу E^*, и, тем самым, исходную подгруппу D.

Начнём с того, что приготовим состояние

$$|\xi\rangle = 2^{-k/2} \sum_{x \in G} |x\rangle = H^{\otimes k} |0^k\rangle$$

в одном квантовом регистре. Во второй регистр поместим состояние $|0^n\rangle$ и применим оператор U. Затем выбросим второй регистр, т.е. не будем его больше использовать. Получится смешанное состояние

$$\rho \,=\, \mathrm{Tr}_2\Big(U(|\xi\rangle\langle\xi| \otimes |0^n\rangle\langle 0^n|)U^\dagger\Big) \,=\, 2^{-k} \sum_{x,y:x-y \in D} |x\rangle\langle y|.$$

Теперь применим оператор $H^{\otimes k}$:

$$\gamma \,=\, H^{\otimes k} \rho H^{\otimes k} \,=\, 2^{-2k} \sum_{a,b} \sum_{x,y:x-y \in D} (-1)^{a \cdot x - b \cdot y} |a\rangle\langle b|.$$

Легко видеть, что величина $\sum\limits_{x,y:x-y \in D} (-1)^{a \cdot x - b \cdot y}$ отлична от нуля только в том случае, когда $a = b \in E^*$. Таким образом,

$$\gamma = \frac{1}{|E^*|} \sum_{a \in E^*} |a\rangle\langle a|.$$

Это в точности матрица плотности для случайного равномерно распределенного элемента группы E^*. Теперь осталось воспользоваться следующей леммой, которую мы сформулируем в виде задачи.

Задача 12.1. Пусть $h_1, \ldots, h_l$ — независимые случайные равномерно распределенные элементы абелевой группы X. Докажите, что они порождают всю группу X с вероятностью $\geqslant 1 - |X|/2^l$.

Таким образом, достаточно $2k$ случайных элементов, чтобы породить всю группу E^* с вероятностью ошибки $\leqslant 2^{-k}$. (Такая маленькая вероятность ошибки получается без особых затрат по сравнению с $1/3$. Чтобы сделать её ещё меньше, эффективнее всего воспользоваться стандартной процедурой: повторить все вычисление несколько раз и выбрать наиболее часто встречающийся ответ).

Подведём итог: для нахождения «скрытой подгруппы» D требуется $O(k)$ обращений к квантовому оракулу. В целом алгоритм имеет сложность $O(k^3)$.

12.2. Разложение на множители и нахождение периода относительно возведения в степень. Второе свидетельство в пользу гипотезы $\mathrm{BQP} \supset \mathrm{BPP}$ — быстрые квантовые алгоритмы разложения числа на простые множители и вычисления дискретного логарифма. Они были найдены П. Шором [38]. Обсудим пока первую из этих двух задач.

Факторизация числа. Дано натуральное число y. Требуется найти его разложение на простые множители

$$y = p_1^{\alpha_1} p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}.$$

Эта задача считается сложной настолько, что на предположении о трудности её решения основываются практические алгоритмы криптографии. С теоретической точки зрения положение несколько хуже: неизвестно ни свед́ение к задаче факторизации задач из класса NP, ни другие «прямые» свидетельства в пользу её сложности. (Слово «прямые» взято в кавычки из-за того, что в настоящее время неизвестен ответ на вопрос $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$.) Таким образом, предположение о сложности задачи факторизации пополняет и без того обильную коллекцию недоказанных гипотез в вычислительной теории сложности. Количество таких гипотез хочется по возможности уменьшать. В этом и состоит основная ценность результата Шора — если совершить один «акт веры» и уверовать в сложность задачи факторизации, то необходимость в ещё одном акте веры (относительно бо́льших вычислительных возможностей квантового компьютера) отпадает.

Мы будем строить быстрый квантовый алгоритм не для решения задачи факторизации, а для решения другой задачи Нахождение периода, к которой задача факторизации сводится с помощью классического вероятностного алгоритма.

Нахождение периода. Имеется число q, записывающееся не более чем n двоичными цифрами ($1 \leqslant q < 2^n$) и число a такое, что $a < q$,

$(a,q)=1$ ((a,q) обозначает наибольший общий делитель). Нужно найти *период* a относительно q, т.е. такое наименьшее неотрицательное число t, что $a^t \equiv 1 \pmod{q}$.

Другими словами, период — это порядок числа a в мультипликативной группе вычетов $(\mathbb{Z}/q\mathbb{Z})^*$. Будем обозначать период числа a относительно q как $\mathrm{per}_q(a)$.

Ниже мы построим квантовый алгоритм для решения задачи о нахождении периода числа. Но начнём с того, что опишем классическое вероятностное сведéние задачи факторизации к задаче вычисления периода. Читателю также предлагается вспомнить вероятностный тест простоты числа, изложенный в первой части (см. раздел 3.3).

12.3. Сведение факторизации к вычислению периода. Итак, предположим, что мы умеем решать задачу нахождения периода. Ясно, что факторизацию числа y можно получить, используя $O(\log y)$ раз подпрограмму, которая по любому составному числу вычисляет какой-то его делитель с вероятностью, не меньшей $1/2$. (Конечно, нужна также стандартная процедура усиления вероятностей, описанная на с. 38.)

Процедура нахождения делителя.

Вход: число y.

Шаг 1. Проверяем чётность y. Если y — чётное, то выдаём ответ «2», в противном случае переходим к шагу 2.

Шаг 2. Проверяем, извлекается ли из y нацело корень k-й степени при $k=2,\ldots,\log_2 y$. Если $y=m^r$, то ответ «m», иначе переходим к шагу 3.

Шаг 3. Выбираем случайное a среди чисел от 1 до y, вычисляем $r=\mathrm{per}_y(a)$ (используя имеющийся по предположению алгоритм нахождения периода) и, если r — нечётное, то ответ «y — простое». В противном случае находим $d=(a^{r/2}-1,y)$ (скажем, алгоритмом Евклида) и переходим к шагу 4.

Шаг 4. Если $d>1$, то ответ «d», в противном случае ответ «y — простое».

Анализ процедуры нахождения делителя. Докажем, что вероятность получить делитель числа y в результате работы процедуры нахождения делителя не меньше, чем $1-1/2^{k-1}$, где k — число различных простых делителей y. (Заметим, в частности, что эта вероятность равна 0 для простого y, так что данная процедура может также использоваться и как тест простоты числа.) При доказательстве нам потребуется китайская теорема об остатках и тот факт, что мультипликативная группа вычетов по модулю p^α, где p простое, — циклическая (см. [2, Гл. 6, §3]).

Если $r=\mathrm{per}_y(a)$ — чётное, то $(a^{r/2}+1)(a^{r/2}-1)\equiv 0 \pmod{y}$. Так что в этом случае процедура выдаст ответ «y — простое» только тогда, когда $a^{r/2}\equiv -1 \pmod{y}$.

Запишем разложение y на простые множители $y=\prod_{j=1}^k p_j^{\alpha_j}$ и введём обозначения

$$a_j \equiv a \pmod{p_j^{\alpha_j}}, \qquad r_j = \mathrm{per}_{(p_j^{\alpha_j})}\, a_j = 2^{s_j} r_j', \text{ где } r_j' \text{ — нечётное.}$$

Докажем, что процедура выдаёт ответ «y — простое» тогда и только тогда, когда $s_1=s_2=\ldots=s_k$. Действительно, если $s_1=s_2=\ldots=s_k=0$, то r нечётно (поскольку r — наименьшее общее кратное всех r_j). Если $s_1=s_2=\ldots=s_k\geqslant 1$, то $a_j^{r_j/2}\equiv -1 \pmod{p_j^{\alpha_j}}$ (используем цикличность $(\mathbb{Z}/p_j^{\alpha_j}\mathbb{Z})^*$), а, значит, и $a^{r/2}\equiv -1 \pmod{y}$ (используем китайскую теорему об остатках). Наоборот, если не все s_j равны, то при некотором m получим $a_m^{r/2}\equiv 1 \pmod{p_m^{\alpha_m}}$, т.е. $a^{r/2}\not\equiv -1 \pmod{y}$.

По китайской теореме об остатках случайный равномерный выбор a есть то же самое, что независимый случайный равномерный выбор всех a_j. Оценим для некоторого s вероятность события $s_1=s$ при независимом выборе a_1. Пусть $p_1^{\alpha_1}-1=2^t q$, где q — нечётное, g — образующая (циклической) группы $(\mathbb{Z}/p_1^{\alpha_1}\mathbb{Z})^*$. Тогда

$$|\{a_1 : s_1=s\}| = |\{g^{2^{t-s}m} : m \text{ — нечётное}\}| =$$
$$= \begin{cases} q, & \text{если } s=0,\\ (2^s-2^{s-1})q, & \text{если } s>0,\end{cases}$$

поэтому вероятность $s_1=s$ не больше $1/2$. Отсюда следует искомая оценка вероятности успеха всей процедуры нахождения делителя: вероятность события $s_1=s_2=\ldots=s_k$ не выше $1/2^{k-1}$, поэтому с вероятностью не меньше $1-1/2^{k-1}$ процедура нахождения делителя найдёт делитель y.

12.4. Квантовый алгоритм нахождения периода: основная идея.

Рассмотрим оператор умножения вычета на a, действующий по правилу $U_a\colon |x\rangle \mapsto |ax \bmod q\rangle$. (Более корректное обозначение — $U_{q,a}$, однако число q не меняется на протяжении всего вычисления, поэтому мы опускаем его в индексах). Этот оператор переставляет базисные векторы при $0\leqslant x<q$ (напомним, что $(a,q)=1$). Будем считать, что на остальных базисных векторах он действует тождественно, $U_a\colon |x\rangle\mapsto|x\rangle$ при $x\geqslant q$.

Поскольку для умножения вычетов есть обычная булева схема полиномиального — $O(n^2)$ — размера, то существует и квантовая схема примерно такого же размера (использующая напрокат дополнительные q-биты, как это объяснялось раньше).

Перестановка, которую задаёт оператор U_a, разбивается на циклы. Цикл, содержащий a, содержит и 1 (после $\mathrm{per}_q(a)-1$ итераций мы попадаем из a в 1). Алгоритм, о котором пойдёт речь, начинает с состояния $|1\rangle$ и применяет к нему оператор U_a по многу раз. Но за пределы орбиты a (цикла перестановки, которому принадлежит a) мы такими преобразованиями не выйдем. Поэтому рассмотрим ограничение оператора U_a на подпространство, порождённое орбитой a.

$$\text{Собственные числа для } U_a: \quad \lambda_k = e^{2\pi i\cdot k/t}, \text{ где } t \text{ — период.}$$

$$\text{Собственные векторы для } U_a: \quad |\xi_{a,k}\rangle = \frac{1}{\sqrt{t}}\sum_{m=0}^{t-1} e^{-2\pi i\cdot km/t}|a^m\rangle.$$

Легко проверить, что написанные векторы действительно собственные. Достаточно заметить, что умножение на a приводит к сдвигу индексов в сумме. Если заменить переменную суммирования, чтобы устранить этот сдвиг, получим множитель $e^{2\pi i\cdot k/t}$.

Если бы мы могли измерять собственные числа оператора U_a, то получали бы числа k/t. Сначала разберём, как это может нам помочь в нахождении периода.

Пусть у нас есть машина M, которая при каждом запуске выдает нам число k/t, где t — искомый период, а k — равномерно распределённое на множестве $\{0,\ldots,t-1\}$ случайное число. Мы предполагаем, что k/t представлено в виде несократимой дроби k'/t' (если бы машина выдавала число в виде k/t, то вообще не было бы проблем).

Получив несколько дробей такого вида k_1'/t_1', $k_2'/t_2', \ldots, k_l'/t_l'$, можно с большой вероятностью найти число t, приводя эти дроби к общему знаменателю.

Лемма. *Если получено l дробей, то вероятность того, что наименьшее общее кратное их знаменателей отлично от t, меньше $3\cdot 2^{-l}$.*

Доказательство. Дроби $k_1'/t_1', \ldots, k_l'/t_l'$ получаются сокращением дробей $k_1/t, \ldots, k_l/t$ (т.е. $k_j'/t_j' = k_j/t$), где $k_1, \ldots, k_l$ — независимо распределённые случайные числа. Достаточно, чтобы эти числа были в совокупности взаимно просты, тогда наименьшее общее кратное $t_1', \ldots, t_l'$ будет равно t.

Вероятность того, что $k_1, \ldots, k_l$ имеют общий простой делитель p, не больше, чем $1/p^l$. Поэтому вероятность получить не t после приведения к общему знаменателю не превосходит $\displaystyle\sum_{k=2}^{\infty}\frac{1}{k^l} < 3\cdot 2^{-l}$ (эта сумма заведомо включает в себя все простые, меньшие t). □

Теперь будем строить машину M. Она должна содержать схему, измеряющую собственные числа оператора U_b *для любого* b (а не только для $b = a$ — числа, для которого ищется период). Точнее говоря, нам нужен оператор $U\colon |b,x\rangle \mapsto |b, bx \bmod q\rangle$, если $(b,q) = 1$. Как оператор U действует в остальных случаях, неважно. Его можно доопределить любым вычислительно тривиальным способом. На самом деле, все приведённые ранее рассуждения об имитации классических схем квантовыми сохраняют силу и для имитации схем, вычисляющих частично определённые функции.

Задача 12.2. Используя оператор U, реализуйте оператор $\Lambda(U_b)$ для любого b, взаимно простого с q.

Обозначим $\mathcal{L}_{a,k} = \mathbb{C}(|\xi_{a,k}\rangle)$ (подпространство, порождённое $|\xi_{a,k}\rangle$), тогда искомая схема должна реализовывать измеряющий оператор $W = \sum\limits_{k=0}^{t-1} V_{a,k}\otimes \Pi_{\mathcal{L}_{a,k}}$ с операторами $V_{a,k}$ вида $|0\rangle \mapsto \sum_{y,z} c_{y,z}|y,z\rangle$, где y — некоторая несократимая дробь, а z — мусор. При этом для условных вероятностей должно выполняться неравенство

$$\mathbf{P}\left(\boxed{\tfrac{k}{t}}\,\middle|\, k\right) \stackrel{\text{def}}{=} \sum_z \left|\left\langle \boxed{\tfrac{k}{t}}, z \,\middle|\, V_{a,k} \,\middle|\, 0\right\rangle\right|^2 \geqslant 1-\varepsilon,$$

где $\boxed{\tfrac{k}{t}}$ обозначает несократимую дробь, представляющую рациональное число k/t.

Построение такой измеряющей схемы довольно сложное, поэтому вначале объясним, как из неё строится машина M. Возьмём состояние $|1\rangle$ в качестве начального. Прямое вычисление (читателю рекомендуется его проделать) показывает, что $|1\rangle = \frac{1}{\sqrt{t}}\sum_{k=0}^{t-1}|\xi_k\rangle$. Это равенство гарантирует равномерное распределение числителей дробей. Проведём измерение в этом состоянии, тогда по формуле полной вероятности получаем

$$\mathbf{P}\Big(W(|0\rangle\otimes|1\rangle), y\Big) = \sum_k \mathbf{P}(y|k)\,\mathbf{P}(|1\rangle, \mathcal{L}_k).$$

Вероятности всех $|\xi_k\rangle$ равны: $\mathbf{P}(|1\rangle,\mathcal{L}_k) = |\langle\xi_k|1\rangle|^2 = 1/t$, а указанное выше свойство условных вероятностей гарантирует нам, что с вероятностью $1-\varepsilon$ будет получаться $\boxed{\tfrac{k}{t}}$. Как будет видно в дальнейшем, при построении W можно сделать величину ε сколь угодно малой.

Условно работу машины M можно представить в виде такого процесса:

$$|1\rangle \longrightarrow \boxed{\begin{matrix}\text{случайный выбор } k\\ \text{(бог, играющий в кости)}\end{matrix}} \xrightarrow{|\xi_{a,k}\rangle} \boxed{\text{измерение } W} \begin{matrix} \nearrow y \neq \boxed{\frac{k}{t}}\Big\} < \varepsilon, \\ \searrow y = \boxed{\frac{k}{t}}\Big\} > 1-\varepsilon. \end{matrix}$$

(Случайный выбор k происходит сам по себе, без применения какого бы то ни было оператора. Просто формула полной вероятности устроена так, как будто до начала измерения генерируется случайное k, которое затем остается постоянным. Разумеется, формула условной вероятности верна только тогда, когда оператор W является измеряющим для заданных подпространств $\mathcal{L}_{a,k}$).

12.5. Построение измеряющего оператора. Теперь будем строить оператор, измеряющий собственные числа U_a. Как уже было сказано, можно ограничиться изучением действия этого оператора на вход $|\xi_{a,k}\rangle$. Построение разделяется на три этапа.

1. Ищем информацию о $\lambda_k = e^{2\pi i\varphi_k}$, где $\varphi_k = \dfrac{k}{t} \bmod 1$ (см. 12.5.1).
2. Локализуем значение φ с небольшой точностью. Самое время подчеркнуть, что во всех приводимых рассуждениях есть два параметра: *вероятность ошибки* ε и *точность* δ. Мы получаем некоторое число z как результат измерения, при этом должно выполняться условие $\mathbf{Pr}[|z-\varphi_k| > \delta] \leqslant \varepsilon$. Пока нас устроит небольшая точность, скажем, $\delta = 1/8$ (см. 12.5.2).
3. Далее нужно увеличить точность. Необходимо уметь отличать друг от друга числа вида $\varphi = k/t$, где $0 \leqslant k < t < 2^n$. Заметим, что если $k_1/t_1 \neq k_2/t_2$, то $|k_1/t_1 - k_2/t_2| \geqslant 1/t_1t_2 > 1/2^{2n}$. Поэтому, зная значение $\varphi_k = k/t$ с точностью $1/2^{2n+1}$, мы можем определить его абсолютно точно (в виде несократимой дроби). Чтобы сделать это эффективно (за полиномиальное время), можно использовать алгоритм цепных дробей (см. 12.5.3).

12.5.1. Как получать информацию о собственном числе. В разделе 11 был введён оператор $\Xi(U_a) = (H\otimes I)\Lambda(U_a)(H\otimes I)$, измеряющий собственные числа. В нашем случае $\lambda_k = e^{2\pi i\varphi_k}$, поэтому можно записать этот оператор в виде

$$\Xi(U_a) = \sum_k V_{a,k}\otimes \Pi_{\mathcal{L}_{a,k}}, \quad V_{a,k} = \frac{1}{2}\begin{pmatrix} 1+e^{2\pi i\varphi_k} & 1-e^{2\pi i\varphi_k} \\ 1-e^{2\pi i\varphi_k} & 1+e^{2\pi i\varphi_k}\end{pmatrix},$$

а его действие в виде

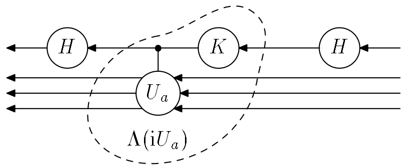
$$|0\rangle\otimes|\xi_k\rangle \overset{\Xi(U_a)}{\longmapsto} \left(\frac{1+e^{2\pi i\varphi_k}}{2}|0\rangle + \frac{1-e^{2\pi i\varphi_k}}{2}|1\rangle\right)\otimes|\xi_k\rangle,$$

так что для условных вероятностей получаем выражение

$$\mathbf{P}(0|k) = \left|\frac{1+e^{2\pi i\varphi_k}}{2}\right|^2 = \frac{1+\cos(2\pi\varphi_k)}{2}.$$

Нам потребуется ещё оператор $\Xi(iU_a)$. Его также нетрудно реализовать. Реализация, изображённая на рисунке, использует оператор $K = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$ из стандартного базиса. Обведённый фрагмент реализует оператор $\Lambda(iU_a)$. Действительно, K умножает на i только $|1\rangle$, но как раз в этом случае применяется оператор U_a (по определению оператора $\Lambda(U_a)$). Для оператора $\Xi(iU_a)$ условные вероятности равны

$$\mathbf{P}(0|k) = \frac{1-\sin(2\pi\varphi_k)}{2}.$$



Сложность реализации операторов $\Xi(U_a)$ и $\Xi(iU_a)$ зависит от сложности реализации оператора $\Lambda(U_a)$, которая ненамного выше сложности реализации оператора U_a (см. задачу 12.2).

12.5.2. Оценка условных вероятностей. Мы будем локализовывать значение φ_k, оценивая условные вероятности, приведённые выше. Для получения такой оценки будем применять операторы $\Xi(U_a)$ и $\Xi(iU_a)$ к различным «приборам» (дополнительным q-битам). Рассуждения одинаковы для обоих операторов, поэтому ограничимся случаем $\Xi(U_a)$.

У нас есть квантовый регистр A, в котором находится $|\xi_{a,k}\rangle$. (На самом деле там вначале был $|1\rangle = \dfrac{1}{\sqrt{t}}\sum_{k=0}^{t-1}|\xi_k\rangle$, но мы рассматриваем $|\xi_{a,k}\rangle$ по отдельности; это корректно в силу вида измеряющего оператора). Заведём большое количество (s штук) вспомогательных регистров длиной в 1 бит. Каждый из этих регистров будет использоваться для применения оператора $\Xi(U_a)$.

Как было доказано в разделе 11 (см. с. 89), условные вероятности в таком случае перемножаются. Для оператора $\prod\limits_{r=1}^{s}\Xi(U_a)[r,A]$ условные вероятности будут равны $\mathbf{P}(y_1, \ldots, y_s \big| k) = \prod_{r=1}^{s} \mathbf{P}(y_r \big| k)$ (здесь через y_r обозначено значение в r-ом бите).

Далее с битами, в которых записаны результаты «экспериментов», будут уже производиться классические действия. Поскольку условные вероятности перемножаются, можно считать, что мы оцениваем вероятность выпадения 1 в серии испытаний Бернулли.

Если монета брошена s раз, то доля выпавших единиц $(\sum y_r)/s$ примерно равна $\mathbf{P}(1\big|k)$. С какой точностью верна такая оценка? Из теории вероятностей известно, что $\mathbf{Pr}\left[\left|\frac{\sum_{r=1}^{s} y_r}{s} - \mathbf{P}(1\big|k)\right| > \delta\right] < 2e^{-c\delta^2 s}$, где $c > 0$ — некоторая константа. Это показывает, что при любом фиксированном δ можно добиться вероятности ошибки ε за $O(\log(1/\varepsilon))$ испытаний.

Итак, мы научились находить с некоторой точностью δ синус и косинус от φ_k. Теперь подберём δ таким, чтобы значение φ_k можно было установить по значениям синуса и косинуса с точностью $1/8$. На этом второй этап завершен.

12.5.3. Экспоненциально точное определение собственных чисел. Для увеличения точности мы будем использовать, наряду с $\Lambda(U_a)$, операторы $\Lambda((U_a)^{2^j})$ для всех $j \leqslant 2n$. Числа мы можем быстро возводить в степень, а операторы, вообще говоря, — нет. Но оператор умножения на число U_a обладает следующим замечательным свойством:

$$(U_a)^p = U_{a^p \bmod q}.$$

Следовательно, $\Lambda((U_a)^{2^j}) = \Lambda(U_b)$, где $b \equiv a^{2^j} \pmod q$. Нужные нам значения параметра b можно вычислить при помощи схемы полиномиального размера, а затем использовать результат задачи 12.2.

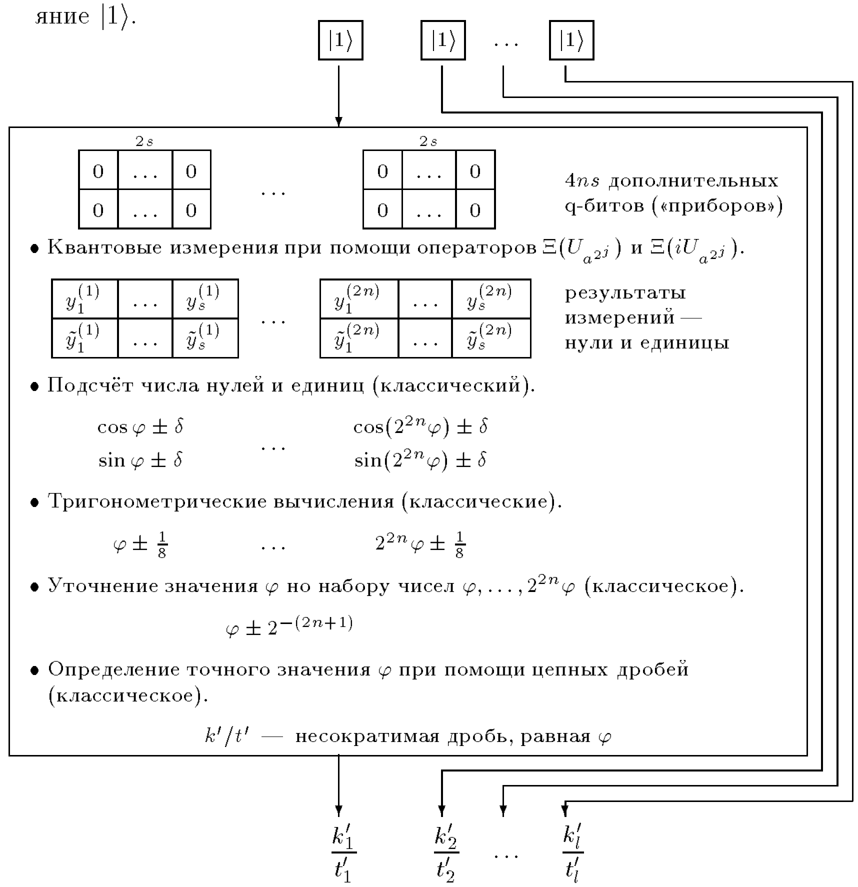
Вернёмся к схеме пункта 12.5.1. Мы находили собственное число $\lambda_k = e^{2\pi i \varphi_k}$ для некоторого собственного вектора $|\xi_{a,k}\rangle$. Этот же вектор останется собственным и для любой степени оператора U_a, поэтому можно на одном и том же квантовом регистре искать собственное число для $U_a^2 = U_{a^2}$, оно равно $\lambda_k = e^{2\pi i 2\varphi_k}$; для $U_a^4 = U_{a^4}$ оно равно $\lambda_k = e^{2\pi i 4\varphi_k}$; ...

Другими словами, мы можем с точностью $1/8$ определить значения $\varphi_k, 2\varphi_k, \ldots, 2^{2n}\varphi_k$ по модулю 1. *Но это позволяет определить φ_k с точностью $1/2^{2n+1}$ за полиномиальное время.*

Идея доказательства. Множество возможных значений φ_k удобно представлять в виде окружности единичной длины. Зная φ_k с точностью $1/8$, мы выделяем дугу в $1/4$ от всей окружности. Знание $2\varphi_k$ с

<u>Вход:</u> a и q

– Вычисление степеней a^{2^j} $(j = 1, \ldots, 2n)$ по модулю q (классическое).
– Создание l штук квантовых регистров, содержащих базисное состояние $|1\rangle$.



– Вычисление наибольшего общего знаменателя (классическое).

<u>Ответ:</u> t (с вероятностью ошибки $< 3 \cdot 2^{-l} + 4nle^{-cs}$, где $c = \text{const}$)

Таблица 1. *Общая картина вычислений по описанному алгоритму*

точностью 1/8 позволяет выделить две дуги длиной 1/8 каждая, причём только одна из них имеет непустое пересечение с предыдущей дугой.

Важные замечания.

1. Существенно, что вектор $|\xi_k\rangle$ не портится во время вычислений.
2. Всё вычисление в целом зависит от параметров l и s. Суммарная вероятность ошибки не превышает $3 \cdot 2^{-l} + 4nle^{-cs}$, где $c = \mathrm{const}$. Если требуется получить ответ с вероятностью ошибки $\leqslant 1/3$, следует положить $l = 4$, $s = c_1 \log n$, где c_1 — некоторая константа. При этом получается квантовая схема размера $O(n^3 \log n)$.

12.6. Обсуждение алгоритма. Обсудим два естественно возникающих вопроса по поводу изложенного алгоритма.

— **Можно ли находить собственные числа других операторов так же, как в алгоритме вычисления периода?** Да, например, можно находить собственные числа таких операторов U, для которых $U|0\rangle = |0\rangle$, и есть полиномиальная схема реализации оператора $\Lambda(U)$. (Из задачи 7.5 следует, что если для самого оператора U есть полиномиальная схема, то и для оператора $\Lambda(U)$ её также можно построить).

Точность определения собственных чисел произвольного оператора невелика, полиномиально зависит от размера схемы. Если можно эффективно вычислять степени оператора (как и было в рассмотренном алгоритме), то точность можно сделать экспоненциальной.

— **Какие собственные числа мы находим?** Мы находим значение случайно выбранного собственного числа. Распределением по множеству всех собственных чисел можно управлять, выбирая начальное состояние (в алгоритме вычисления периода — $|1\rangle$). Если взять в качестве начального состояние, задаваемое диагональной матрицей плотности

$$\rho = \frac{1}{t}\sum_a |a\rangle\langle a| = \frac{1}{t}\sum_k |\xi_k\rangle\langle\xi_k|,$$

где $|\xi_k\rangle$ пробегает множество собственных векторов U, то получим равномерное распределение на множестве всех собственных чисел. В алгоритме нахождения периода начальное состояние выбиралось *иначе*: мы выбирали такой вектор, чтобы получить равномерное распределение на собственных числах, соответствующих определённой орбите, остальные собственные числа не порождались.

Задача 12.3. Постройте квантовую схему размера $\mathrm{poly}(n\log(1/\delta))$, реализующую преобразование Фурье на группе $\mathbb{Z}_k$ при любом $k \leqslant 2^n$ с

точностью δ. (Определение см. в задаче 8.4. Указание: воспользуйтесь результатом задачи 11.2).

12.7. Задача о скрытой подгруппе в $\mathbb{Z}^k$. Алгоритмы, открытые Саймоном и Шором, обобщаются на довольно широкий класс задач, связанных с абелевыми группами. Самой общей из них является задача о скрытой подгруппе в $\mathbb{Z}^k$ [23]. К ней сводится задача о скрытой подгруппе в любой конечно-порожденной абелевой группе G, поскольку G можно представить как фактор-группу $\mathbb{Z}^k$ (для некоторого k).

«Скрытая подгруппа» $D \subseteq \mathbb{Z}^k$ изоморфна $\mathbb{Z}^k$, поскольку она имеет конечный индекс: порядок группы $E = \mathbb{Z}^k/D$ не превосходит 2^n. С вычислительной точки зрения D представляется базисом $(g_1, \ldots, g_k)$, двоичная запись которого имеет длину $\mathrm{poly}(k, n)$. Любой такой базис считается решением задачи. (Эквивалентность двух базисов можно проверить при помощи полиномиального алгоритма).

Задача о вычислении периода является частным случаем задачи о скрытой подгруппе в $\mathbb{Z}$. Напомним, что $\mathrm{per}_q(a) = \min\{t \geqslant 1 : a^t \equiv 1 \pmod q\}$. Функция $f\colon x \mapsto a^x \bmod q$ удовлетворяет условию (12.1), где $D = \{m\,\mathrm{per}_q(a) : m \in \mathbb{Z}\}$. Эта функция полиномиально вычислима, поэтому любой полиномиальный алгоритм нахождения скрытой подгруппы преобразуется в полиномиальный алгоритм решения задачи о вычислении периода.

Известная задача вычисления дискретного логарифма может быть сведена к задаче о скрытой подгруппе в $\mathbb{Z}^2$. *Дискретным логарифмом* числа a по основанию ζ, где ζ — некоторый первообразный корень по модулю простого числа q (образующая $(\mathbb{Z}/q\mathbb{Z})^*$), называется наименьшее положительное число s такое, что $\zeta^s = a$. Рассмотрим функцию $f\colon (x_1, x_2) \mapsto \zeta^{x_1}a^{x_2} \bmod q$. Эта функция также удовлетворяет условию (12.1), где $D = \{(x_1, x_2) \in \mathbb{Z}^2 : \zeta^{x_1}a^{x_2} \equiv 1 \pmod q\}$. Зная базис подгруппы $D \subseteq \mathbb{Z}^2$, легко найти элемент вида $(s, -1) \in D$. Тогда $\zeta^s = a$, т.е. s есть дискретный логарифм a по основанию ζ.

Опишем квантовый алгоритм решения задачи о скрытой подгруппе в $G = \mathbb{Z}^k$. Он аналогичен алгоритму для случая $G = (\mathbb{Z}_2)^k$, только вместо оператора $H^{\otimes k}$ используется процедура измерения собственных чисел. Вместо базиса самой группы D мы будем искать систему образующих для *группы характеров* $E^* = \mathrm{Hom}(E, \mathbf{U}(1))$ (переход от E^* к D осуществляется при помощи полиномиального алгоритма, см., например, [14, Т. 1]). Характер

$$(g_1, \ldots, g_k) \mapsto \exp(2\pi i \sum_j \varphi_j g_j)$$

задаётся набором чисел $\varphi_1, \ldots, \varphi_k$ по модулю 1. Это рациональные числа со знаменателями не больше $|E^*| \leqslant 2^n$.

Если породить $l = n + 3$ случайных равномерно распределённых характера $(\varphi_1^{(1)}, \ldots, \varphi_k^{(1)})$, …, $(\varphi_1^{(l)}, \ldots, \varphi_k^{(l)})$, то они порождают всю группу E^* с вероятностью $\geqslant 1 - 1/2^{l-n} = 1 - 1/8$ (см. задачу 12.1). Каждую из величин $\varphi_j^{(r)}$ достаточно знать с точностью δ и вероятностью ошибки $\leqslant \varepsilon$, где

$$\delta \leqslant \frac{1}{2^{2n+1}}, \qquad \varepsilon \leqslant \frac{1}{5kl}. \tag{12.3}$$

Последнее условие гарантирует, что суммарная вероятность ошибки будет не больше, чем $1/8 + 1/5 < 1/3$.

Выберем достаточно большое число $M = 2^m$ (конкретная оценка получается из анализа алгоритма). Мы будем работать с целыми числами в диапазоне от 0 до $M - 1$.

Приготовим в одном квантовом регистре длины km состояние

$$|\xi\rangle = M^{-k/2} \sum_{g \in \Delta} |g\rangle, \text{ где } \Delta = \{0, \ldots, M-1\}^k.$$

В другой регистр поместим $|0^n\rangle$. Применим квантовый оракул (12.2) и выбросим второй регистр. Получится смешанное состояние

$$\rho = \mathrm{Tr}_{[km+1,\ldots,km+n]}\Big(U\big(|\xi\rangle\langle\xi| \otimes |0^n\rangle\langle 0^n|\big)U^\dagger\Big) = M^{-k} \sum_{g,h \in \Delta : g-h \in D} |g\rangle\langle h|.$$

Теперь мы собираемся измерить собственные значения операторов сдвига по модулю M:

$$V_j : \Big(g_1, \ldots, g_j, \ldots, g_k\Big) \mapsto \Big(g_1, \ldots, (g_j + 1) \bmod M, \ldots, g_k\Big)$$

(меняется только j-ая компонента). Эти операторы коммутируют, поэтому у них есть общий базис из собственных векторов, и, значит, можно определять их собственные числа одновременно. Собственные числа имеют вид $e^{2\pi i s_j/M}$. Соответствующие собственные векторы равны

$$|\xi_{s_1,\ldots,s_k}\rangle = M^{-k/2} \sum_{(g_1,\ldots,g_k) \in \Delta} \exp\Big(-2\pi i \sum_{j=1}^{k} \frac{g_j s_j}{M}\Big) |g_1, \ldots, g_k\rangle.$$

Вероятность того, что реализуется данный набор $s_1, \ldots, s_k$, равна

$$\mathbf{P}(\rho, \mathcal{L}_{s_1,\ldots,s_k}) = \langle \xi_{s_1,\ldots,s_k} | \rho | \xi_{s_1,\ldots,s_k} \rangle =$$

$$= M^{-2k} \sum_{g,h \in \mathbb{Z}^k} \chi_D(g-h)\, \chi_\Delta(g) \chi_\Delta(h) \exp\Big(2\pi i \sum_{j=1}^{k} \frac{(g_j - h_j)s_j}{M}\Big),$$

где $\chi_A(\cdot)$ обозначает характеристическую функцию множества A. Фурье-образ от произведения равен свертке фурье-образов сомножителей. Таким образом, получаем:

$$\mathbf{P}(\rho, \mathcal{L}_{s_1,\ldots,s_k}) = \frac{1}{|E^*|} \sum_{(\varphi_1,\ldots,\varphi_k) \in E^*} p_{\varphi_1,\ldots,\varphi_k}(s_1, \ldots, s_k),$$

$$\text{где} \quad p_{\varphi_1,\ldots,\varphi_k}(s_1, \ldots, s_k) = \prod_{j=1}^{k} \left(\frac{\sin(M\pi(s_j/M - \varphi_j))}{M \sin(\pi(s_j/M - \varphi_j))} \right)^2.$$

При заданных значениях $\varphi_1, \ldots, \varphi_k$ функция $p_{\varphi_1,\ldots,\varphi_k}(s_1, \ldots, s_k)$ является вероятностным распределением, относительно которого

$$\mathbf{Pr}\Big[|s_j/M - \varphi_j| > \beta\Big] \leqslant \frac{1}{M\beta}$$

(β — любое). Мы измеряем величины s_j/M (как уже говорилось, измерение одной из них не меняет значения другой); при этом нас устроит точность β и вероятность ошибки $\leqslant 1/M\beta$. Тем самым мы получаем значения $\varphi_1, \ldots, \varphi_k$ с точностью $\delta = 2\beta$ и вероятностью ошибки $\leqslant \varepsilon = 2/M\beta$. Теперь осталось подобрать числа M и β, чтобы удовлетворить неравенствам 12.3.

Сложность алгоритма. Требуется $O(n)$ обращений к оракулу, каждый вопрос имеет длину $O(k(n + \log k))$. Размер квантовой схемы оценивается как $O(kn^3)\,\mathrm{poly}(\log k, \log n)$.

Замечание. Для измерения собственных чисел операторов V_j можно воспользоваться квантовым преобразованием Фурье на группе $\mathbb{Z}_M$ при $M = 2^m$ (см. задачу 8.4). Это позволяет несколько уменьшить размер схемы (на логарифмический множитель), однако приходится использовать нестандартные элементы.

13. Квантовый аналог NP: класс BQNP

Можно строить квантовые аналоги не только для класса P, но и для других классических сложностных классов. Мы разберём пример квантового аналога класса NP.

13.1. Модификация классических определений. Квантовое вычисление, как, впрочем, и вероятностное, наиболее естественно описывать, используя частично определённые функции. Ранее мы обходились без этого понятия, чтобы не усложнять изложение лишними деталями, но теперь оно нам потребуется.

Частично определённая булева функция — это функция

$$F : \mathbb{B}^n \to \{0,\, 1,\, \langle\text{не определено}\rangle\}.$$

В этом разделе всюду под булевыми функциями подразумеваются частично определённые булевы функции.

И ещё одно замечание по поводу обозначений: мы использовали обозначение P и для класса полиномиально вычислимых функций, и для класса полиномиально разрешимых предикатов; теперь поступим аналогично, используя обозначения P, NP и т. п. для классов частично определённых функций.

P, естественно, обозначает класс полиномиально вычислимых частично определённых функций. Приведём модифицированное определение класса NP.

Определение 13.1. *Функция* $F\colon \mathbb{B}^* \to \{0, 1, \langle\text{не определено}\rangle\}$ *принадлежит классу* NP, *если есть частично определённая функция* $R \in \mathrm{P}$ *от двух переменных, такая что*

$$F(x) = 1 \quad\Longrightarrow\quad \exists\, y\ \big((|y| < q(|x|)) \wedge (R(x, y) = 1)\big),$$
$$F(x) = 0 \quad\Longrightarrow\quad \forall\, y\ \big((|y| < q(|x|)) \Rightarrow (R(x, y) = 0)\big).$$

Как и раньше, $q(\cdot)$ *— полином.*

Что будет, если в определении 13.1 заменить условие $R \in \mathrm{P}$ на условие $R \in \mathrm{BPP}$? Получится другой, скорее всего, более широкий класс, который можно было бы обозначить BNP. Однако для этого класса есть другое, стандартное, обозначение — MA, указывающее на то, что он входит в иерархию классов, определяемых *играми Артура – Мерлина*. Об играх, которыми задаются сложностные классы, мы уже говорили в разделе 4.1; игры Артура – Мерлина отличаются тем, что Артур — вероятностная полиномиальная машина Тьюринга. Порядок букв в обозначении MA указывает на порядок ходов: вначале Мерлин сообщает y, затем Артур проверяет выполнение предиката $R(x, y)$.

13.2. Квантовое определение по аналогии.

Определение 13.2. *Функция* $F\colon \mathbb{B}^* \to \{0, 1, \langle\text{не определено}\rangle\}$ *принадлежит классу* BQNP, *если существует однородная последовательность квантовых схем полиномиального по* n *размера, реализующих такие операторы* $U_n\colon \mathcal{B}^{\otimes N_n} \to \mathcal{B}^{\otimes N_n}$, *что*

$$F_n(x) = 1 \quad\Longrightarrow\quad \exists\,|\xi\rangle\ \mathbf{P}\Big(U_n|\xi\rangle \otimes |x\rangle \otimes |0^{N_n - n - m_n}\rangle, \mathcal{M}\Big) \geqslant p_1,$$
$$F_n(x) = 0 \quad\Longrightarrow\quad \forall\,|\xi\rangle\ \mathbf{P}\Big(U_n|\xi\rangle \otimes |x\rangle \otimes |0^{N_n - n - m_n}\rangle, \mathcal{M}\Big) \leqslant p_0.$$

Здесь $F_n(\cdot)$ *— ограничение* F *на слова длины* n, $|\xi\rangle \in \mathcal{B}^{\otimes m_n}$, $m_n = \mathrm{poly}(n)$, $\mathcal{M} = |1\rangle \otimes \mathcal{B}^{\otimes(N_n - 1)}$, *а для* p_0 *и* p_1 *должно выполняться условие* $p_1 - p_0 = \Omega(n^{-\alpha})$, $\alpha \geqslant 0$.

Вектор $|\xi\rangle$ выполняет роль подсказки (y) из предыдущего определения. Нам удобнее считать его первым аргументом оператора U_n (чтобы можно было в любой момент положить x константой и исключить из обозначений).

Замечание 13.1. В определении 13.2 кванторы по $|\xi\rangle$ включают в себя только векторы единичной длины. Аналогичное соглашение будем использовать и далее в этом разделе, вынося нормировочные множители за знак $|\,\rangle$.

Если вместо чистых состояний $|\xi\rangle$ рассматривать смешанные, то получается эквивалентное определение: максимум вероятности всё равно достигается на чистом состоянии.

По сути происходит всё та же игра Мерлина с Артуром, только теперь подчиняющаяся законам квантовой механики. Сообщение Мерлина (состояние $|\xi\rangle$) даёт Артуру возможность убедиться в том, что $F(x) = 1$ с вероятностью p_1, если это так. А если $F(x) = 0$, то вероятность, что Мерлину удастся убедить Артура в обратном, не выше p_0 для любого сообщения Мерлина.

Обсудим теперь соотношения между пороговыми вероятностями. Из приведённого в определении 13.2 соотношения следует гораздо более сильное условие на p_0 и p_1.

Лемма 13.1 (усиление вероятностей). *Если* $F \in \mathrm{BQNP}$, *то она удовлетворяет также и такому варианту определения* 13.2, *где условие* $p_1 - p_0 = \Omega(n^{-\alpha})$ *заменено на* $p_1 = 1 - \varepsilon$, $p_0 = \varepsilon$, $\varepsilon = \exp(-O(n^{\beta}))$, $\beta > 0$.

Доказательство. Общая идея усиления вероятностей остаётся прежней: рассмотрим большое, но ограниченное полиномом, количество копий схемы, реализующей оператор $U = U_n$ (индекс n мы будем опускать). К результатам их работы применим функцию голосования с пороговым значением, разделяющим вероятности p_0 и p_1:

$$G(z_1, \ldots, z_k) = \begin{cases} 1, & \text{если } \sum\limits_{j=1}^{k} z_j \geqslant l \\[2ex] 0, & \text{если } \sum\limits_{j=1}^{k} z_j < l, \end{cases} \tag{13.1}$$

где $l = pk$, $p = (p_0 + p_1)/2$. Но теперь появляется дополнительная трудность — Мерлин может пытаться обмануть Артура, сообщая ему неразложимую в тензорное произведение подсказку.

Пусть мы используем k копий схемы U. Предоставим Мерлину бóльшую свободу, разрешив в качестве подсказки любую матрицу плотности $\rho \in \mathbf{L}(\mathcal{B}^{\otimes km})$. Вероятность получения ответов $z_1, \ldots, z_k$ при подсказке ρ равна

$$\mathbf{P}(z_1, \ldots, z_k \,|\, \rho) = \mathrm{Tr}(X^{(z_1)} \otimes \ldots \otimes X^{(z_k)} \rho), \tag{13.2}$$

где

$$X^{(a)} = \mathrm{Tr}_{[m+1,\ldots,N]}\Big(U^\dagger \Pi_1^{(a)} U \big(I_{\mathcal{B}^{\otimes m}} \otimes |x, 0^{N-n-m}\rangle\langle x, 0^{N-n-m}|\big)\Big). \tag{13.3}$$

Здесь $\Pi_1^{(a)}$ — проектор на подпространство состояний, имеющих a в первом q-бите (т. е. $\mathbb{C}(|a\rangle) \otimes \mathcal{B}^{\otimes(N-1)}$).

Чтобы убедить Артура в правильности $F(x) = 1$, Мерлин может дать подсказку $\rho = \rho_x^{\otimes k}$, где $\rho_x = |\xi_x\rangle\langle\xi_x|$ — сообщение, которое убеждает Артура, действующего по схеме U, с вероятностью p_1. По общим свойствам квантовой вероятности, формула (13.2) преобразуется в

$$\mathbf{P}(z_1, \ldots, z_k \,|\, \rho) = \prod_{j=1}^{k} \mathrm{Tr}(X^{(z_j)} \rho_x) = \prod_{j=1}^{k} \mathbf{P}(z_j | \rho_x).$$

Рассмотрим теперь случай, когда $F(x) = 0$. Нам нужно оценить вероятность $\mathbf{P}(z_1, \ldots, z_k)$ для произвольного сообщения Мерлина ρ. Выберем в пространстве $\mathcal{B}^{\otimes m}$ ортонормированный базис, в котором диагонализуется оператор $X^{(1)}$ (этот оператор, очевидно, эрмитов). Оператор $X^{(0)} = I - X^{(1)}$ диагонален в том же базисе. Определим набор «условных вероятностей» $p(z|d) = \langle d|X^{(z)}|d\rangle$, где $|d\rangle$ — один из базисных векторов. (Очевидно, что $p(z|d) \geqslant 0$ и $p(0|d) + p(1|d) = 1$.) Тогда величина $\mathbf{P}(z_1, \ldots, z_k \,|\, \rho)$ приобретает вид

$$\mathbf{P}(z_1, \ldots, z_k \,|\, \rho) = \sum_{d_1,\ldots,d_k} p_{d_1\ldots d_k}\, p(z_1|d_1) \ldots p(z_k|d_k), \qquad \sum_{d_1,\ldots,d_k} p_{d_1\ldots d_k} = 1. \tag{13.4}$$

Здесь $p_{d_1\ldots d_k} = \langle d_1 \ldots d_k|\rho|d_1 \ldots d_k\rangle$.

Формула (13.4) имеет следующую интерпретацию. Рассмотрим набор вероятностей $\mathbf{P}(z_1, \ldots, z_k \,|\, \rho)$ для всех последовательностей $(z_1, \ldots, z_k)$ как вектор в k-мерном вещественном пространстве. Мы показали, что этот вектор на произвольной подсказке ρ принадлежит выпуклой оболочке таких же векторов на разложимых подсказках $|d_1, \ldots, d_n\rangle$. Поэтому наибольшая вероятность события $G(z_1, \ldots, z_k) = 1$ (для любой функции G) достигается на подсказках такого вида.

В случае, когда G — пороговая функция (13.1),

$$p_{\max} = \max_{\rho} \mathbf{Pr}\left[G(z_1, \ldots, z_k) = 1 \big| \rho\right] = \sum_{j \geqslant l} \binom{k}{j} p_*^j (1 - p_*)^{k-j}, \tag{13.5}$$

где $p_* = \max_{|\xi\rangle} \langle\xi|X^{(1)}|\xi\rangle$. Согласно условию, $p_* \geqslant p_1$, если $F(x) = 1$, и $p_* \leqslant p_0$, если $F(x) = 0$. Оценим величину $p_{\max}$ в этих случаях, соответственно, снизу и сверху.

Будем использовать неравенство Чернова (см. [18]). Пусть

$$H(p, q) = -(1-p) \ln \frac{1-q}{1-p} - p \ln \frac{q}{p}.$$

Тогда при $p_* \geqslant p_1$ получаем

$$1 - p_{\max} \leqslant \exp(-H(p, p_1)k),$$

а при $p_* \leqslant p_0$ —

$$p_{\max} \leqslant \exp(-H(p, p_0)k).$$

Из неравенства $\ln(1 + x) \leqslant x$ следует, что $H(p, q) \geqslant 0$. Используя более точное разложение $\ln(1+x) \leqslant x - x^2/2 + x^3/3$, можно получить оценку $H(p, q) = \Omega((p-q)^2)$. Так что при $k = n^{2\alpha+\beta}$ указанные в условии оценки на ε выполнены. □

Замечание 13.2. Важным моментом в изложенном доказательстве является тот факт, что $X^{(0)}$ и $X^{(1)}$ диагонализуются в одном и том же базисе. Вообще, усиление вероятностей для нетривиальных сложностных классов (как квантовых, так и классических) — вещь довольно тонкая.

13.3. Полные задачи. В классе BQNP, как и в NP, есть полные задачи относительно той же самой полиномиальной сводимости, которую мы рассматривали раньше. Вот простейший пример.

Задача 0. Зададим функцию F следующим образом. Пусть Z — множество троек вида

$$(\langle\text{описание квантовой схемы } W\rangle, p_0, p_1),$$

где под описанием схемы понимается её приближённая реализация в стандартном базисе, а $p_1 - p_0 = \Omega(n^{-\alpha})$ ($\alpha > 0$, n — размер описания схемы). Тогда для $z \in Z$

$F(z) = 1 \iff$ если существует вектор $|\xi\rangle$, при действии на который мы получим в первом бите 1 с вероятностью, бóльшей p_1;

$F(z) = 0 \iff$ если для всех $|\xi\rangle$ вероятность получить в первом бите 1 меньше p_0.

Полнота задачи 0 очевидна. Всё, что требуется для построения сведéния, содержится в определении 13.2. Вход x войдёт в описание схемы W вместе со схемой U_n.

Рассмотрим более интересные примеры. Для начала дадим определение квантового аналога 3-КНФ — локального гамильтониана (локальность является аналогом ограниченности числа переменных, входящих в одну дизъюнкцию).

Определение 13.3. *Оператор* $H\colon \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes n}$ *называется* k-локальным гамильтонианом, *если он выражается в виде*

$$H = \sum_j H_j[S_j],$$

где каждое слагаемое — эрмитов оператор, действующий на множестве q-битов S_j, $|S_j| \leqslant k$, *на остальных q-битах он действует тождественно.*

При этом выполнено условие нормировки $0 \leqslant H_j \leqslant 1$ *(другими словами, и* H_j, *и* $I - H_j$ *— положительно полуопределённые).*

Задача 1: локальный гамильтониан. Пусть Z — множество троек вида

$$(\langle \text{описание } k\text{-локального гамильтониана } H\rangle, a, b),$$

где $k = O(1)$, $0 \leqslant a < b$, $b - a = \Omega(n^{-\alpha})$, $(\alpha > 0)$. Тогда для $z \in Z$

$F(z) = 1 \iff$ если у H есть собственное число, не большее a,

$F(z) = 0 \iff$ если все собственные числа H больше b.

Утверждение 13.2. *Задача* локальный гамильтониан *принадлежит* BQNP.

Доказательство. Опишем вначале основную идею. Мы построим такую схему W, которая использует подсказку из пространства, в котором действует H, и выдаёт ответ «да» (значение 1) на подсказке $|\eta\rangle$ с вероятностью $p = 1 - r^{-1}\langle\eta|\, H\, |\eta\rangle$, где r — число слагаемых в гамильтониане H. Если $|\eta\rangle$ — собственный вектор, соответствующий собственному числу a, то вероятность ответа «да» будет

$$p = 1 - r^{-1}\langle\eta|\, H\, |\eta\rangle \geqslant 1 - r^{-1}a,$$

а если все собственные числа H больше b, то

$$p = 1 - r^{-1}\langle\eta|\, H\, |\eta\rangle \leqslant 1 - r^{-1}b.$$

Сперва построим такую схему для одного слагаемого. Пусть это будет $H_j = \sum_s \lambda_s |\psi_s\rangle\langle\psi_s|$, действующий на множестве q-битов S_j. Поскольку размерность пространства, на котором действует H_j, ограничена константой, мы можем реализовать оператор

$$W_j\colon |\psi_s, 0\rangle \mapsto |\psi_s\rangle \otimes \left(\sqrt{\lambda_s}|0\rangle + \sqrt{1-\lambda_s}|1\rangle\right),$$

который действует на множестве q-битов $S_j \cup \{\langle\text{ответ}\rangle\}$, где $\langle\text{ответ}\rangle$ обозначает q-бит, из которого берётся результат работы схемы. На остальных q-битах подсказки W_j действует тождественно.

Вычислим вероятность 1 в бите результата после применения W_j к состоянию $|\eta, 0\rangle$ (бит результата установлен в 0 перед началом работы схемы). Пусть $|\eta\rangle = \sum_s y_s |\psi_s\rangle$ — разложение $|\eta\rangle$ по ортогональной системе собственных векторов H_j. Имеем, по определению вероятности,

$$\mathbf{P}_j(1) = \langle\eta, 0|\, W_j^\dagger \big(I \otimes \underbrace{|1\rangle\langle 1|}_{\langle\text{ответ}\rangle}\big) W_j\, |\eta, 0\rangle =$$

$$= \left(\sum_s y_s^* \langle\psi_s, 0|\right) W_j^\dagger \big(I \otimes \underbrace{|1\rangle\langle 1|}_{\langle\text{ответ}\rangle}\big) W_j \left(\sum_t y_t |\psi_t, 0\rangle\right) =$$

$$= \sum_{s,t} \sqrt{1-\lambda_s}\, y_s^* \sqrt{1-\lambda_t}\, y_t \langle\psi_s|\psi_t\rangle = \sum_s (1-\lambda_s) y_s^* y_s = 1 - \sum_s \lambda_s y_s^* y_s =$$

$$= 1 - \langle\eta|\, H\, |\eta\rangle. \qquad (13.6)$$

Общая схема W выбирает случайно и равновероятно номер j, после чего применяет оператор W_j. Такое действие можно реализовать измеряющим оператором вида $\sum_j |j\rangle\langle j| \otimes W_j$, применённым к вектору $\left(\frac{1}{\sqrt{r}} \sum_j |j\rangle\right) \otimes |\eta, 0\rangle$. (Здесь $|j\rangle$ обозначает базисный вектор во вспомогательном r-мерном пространстве.) Проводя вычисления аналогично (13.6), получаем

$$\mathbf{P}(1) = \sum_j \frac{1}{r} \langle j, \eta, 0|\, W_j^\dagger \big(I \otimes \underbrace{|1\rangle\langle 1|}_{\langle\text{ответ}\rangle}\big) W_j\, |j, \eta, 0\rangle = \sum_j \frac{1}{r} \mathbf{P}_j(1) = \qquad (13.7)$$

$$= 1 - r^{-1}\langle\eta|\, H\, |\eta\rangle.$$

□

Утверждение 13.3. *Задача* локальный гамильтониан *полна в классе* BQNP *относительно полиномиальной сводимости.*

Идея доказательства восходит к Фейнману [29]: замена унитарной эволюции не зависящим от времени гамильтонианом (т. е. переход от схемы к локальному гамильтониану).

Доказательство. Итак, пусть есть схема размера L: $U = U_L \cdot \ldots \cdot U_1$. Будем считать, что U действует на пространстве из N q-битов, первые m из которых — q-биты подсказки, а остальные — вспомогательные (взятые напрокат на время вычислений); считаем также, что схема состоит из операторов, действующих на парах q-битов.

Гамильтониан, сопоставляемый схеме. Он действует на пространстве

$$\mathcal{L} = \mathcal{B}^{\otimes N} \otimes \mathbb{C}^{L+1},$$

где первый сомножитель — пространство, на котором действует схема, а второй сомножитель — пространство счётчика шагов (часы). Состоит этот гамильтониан из трёх слагаемых

$$H = H_{\text{in}} + H_{\text{prop}} + H_{\text{out}}.$$

Слагаемое H_{in} отвечает начальному состоянию и равно

$$H_{\text{in}} = \left(\sum_{s=m+1}^{N} \Pi_s^{(1)}\right) \otimes |0\rangle\langle 0|, \tag{13.8}$$

где $\Pi_s^{(\alpha)}$ — проектор на подпространство векторов, у которых s-й бит равен α. Второй сомножитель в этой формуле действует в пространстве счётчика.

Слагаемое H_{out} отвечает конечному состоянию и равно

$$H_{\text{out}} = \Pi_1^{(0)} \otimes |L\rangle\langle L|, \tag{13.9}$$

здесь мы считаем, что бит результата — первый.

И, наконец, слагаемое H_{prop} описывает эволюцию системы и состоит, как и следовало ожидать, из L слагаемых, каждое из которых отвечает за переход от $j-1$ к j:

$$H_{\text{prop}} = \sum_{j=1}^{L} H_j,$$

$$H_j = -\frac{1}{2} U_j \otimes |j\rangle\langle j-1| - \frac{1}{2} U_j^\dagger \otimes |j-1\rangle\langle j| + \frac{1}{2} I \otimes \big(|j\rangle\langle j| + |j-1\rangle\langle j-1|\big). \tag{13.10}$$

Каждое слагаемое H_j действует на два q-бита из пространства состояний и на q-биты пространства счётчика.

Замена базиса. Произведём замену базиса, задаваемую оператором

$$W = \sum_{j=0}^{L} U_j \cdot \ldots \cdot U_1 \otimes |j\rangle\langle j|.$$

Полезно обратить внимание на то, что W — измеряющий оператор: измеряется значение счётчика j и к q-битам пространства состояний схемы применяется оператор эволюции за время j.

Гамильтониан при такой замене изменится на сопряжённый: $\widetilde{H} = W^\dagger H W$. Посмотрим, как действует сопряжение оператором W на слагаемые H.

На слагаемое H_{in} сопряжение не влияет:

$$\widetilde{H}_{\text{in}} = W^\dagger H_{\text{in}} W = H_{\text{in}}. \tag{13.11}$$

Действие на слагаемое H_{out}:

$$\widetilde{H}_{\text{out}} = W^\dagger H_{\text{out}} W = \left(U^\dagger \Pi_1^{(0)} U\right) \otimes |L\rangle\langle L|. \tag{13.12}$$

Слагаемое H_j состоит из трёх. Вначале запишем действие сопряжения на первое из слагаемых в (13.10):

$$\begin{aligned}
&W^\dagger \left(-\frac{1}{2} U_j \otimes |j\rangle\langle j-1|\right) W = \\
&= -\frac{1}{2} \sum_{u,t} \big(U_u \cdot \ldots \cdot U_1 \otimes |u\rangle\langle u|\big)^\dagger \big(U_j \otimes |j\rangle\langle j-1|\big)\big(U_t \cdot \ldots \cdot U_1 \otimes |t\rangle\langle t|\big) = \\
&= -\frac{1}{2}\, \big((U_j \cdot \ldots \cdot U_1)^\dagger U_j \cdot \ldots \cdot U_1\big) \otimes \Big(\big(|j\rangle\langle j|\big)^\dagger\, |j\rangle\langle j-1|\, \big(|j-1\rangle\langle j-1|\big)\Big) = \\
&= -\frac{1}{2} I \otimes |j\rangle\langle j-1|.
\end{aligned}$$

Сопряжение двух других слагаемых происходит аналогично, в итоге получаем

$$\begin{aligned}
&\widetilde{H}_j = W^\dagger H_j W = \\
&\quad = I \otimes \frac{1}{2}\Big(|j-1\rangle\langle j-1| - |j-1\rangle\langle j| - |j\rangle\langle j-1| + |j\rangle\langle j|\Big) = I \otimes E_j
\end{aligned}$$

и

$$\widetilde{H}_{\text{prop}} = W^\dagger H_{\text{prop}} W = I \otimes E, \tag{13.13}$$

где

$$E = \sum_{j=1}^{L} E_j = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & & & \\ -\frac{1}{2} & 1 & -\frac{1}{2} & & \mathbf{0} \\ & -\frac{1}{2} & 1 & -\frac{1}{2} & \\ & & -\frac{1}{2} & \ddots & \ddots \\ \mathbf{0} & & & \ddots & \ddots \end{pmatrix}.$$

Оценка собственного числа при ответе «да». Предположим, что схема, на вход которой подан вектор $|\xi\rangle$, даёт ответ 1 с вероятностью не меньше, чем $1-\varepsilon$. Это, по определению, означает, что

$$\mathbf{P}(0) = \langle \xi, 0|\, U^\dagger \Pi_1^{(0)} U\, |\xi, 0\rangle \leqslant \varepsilon.$$

Докажем, что в этом случае у $\widetilde{H}$ (а, значит, и у H) есть малое собственное число. Для этого предъявим такой вектор $|\widetilde{\eta}\rangle$, что $\langle\widetilde{\eta}|\, \widetilde{H}\, |\widetilde{\eta}\rangle$ достаточно мало (минимум квадратичной формы $\langle\cdot|\, \widetilde{H}\, |\cdot\rangle$ достигается на собственном векторе).

В пространстве счётчика выберем вектор

$$|\psi\rangle = \frac{1}{\sqrt{L+1}} \sum_{j=0}^{L} |j\rangle. \qquad (13.14)$$

Искомый вектор $|\widetilde{\eta}\rangle$ равен $|\xi, 0\rangle \otimes |\psi\rangle$. Оценим $\langle\widetilde{\eta}|\, H\, |\widetilde{\eta}\rangle$.

Очевидно, что $E|\psi\rangle = 0$. Поэтому

$$\langle\widetilde{\eta}|\, \widetilde{H}_{\mathrm{prop}}\, |\widetilde{\eta}\rangle = 0 = \langle\widetilde{\eta}|\, \widetilde{H}_j\, |\widetilde{\eta}\rangle.$$

Поскольку все вспомогательные q-биты вначале установлены в 0, то непосредственно из определяющей формулы (13.8) получаем

$$\langle\widetilde{\eta}|\, \widetilde{H}_{\mathrm{in}}\, |\widetilde{\eta}\rangle = 0.$$

Осталось оценить последнее слагаемое

$$\langle\widetilde{\eta}|\, \widetilde{H}_{\mathrm{out}}\, |\widetilde{\eta}\rangle = \langle\widetilde{\eta}|\, \left(U^\dagger \Pi_1^{(0)} U\right) \otimes |L\rangle\langle L|\, |\widetilde{\eta}\rangle = \mathbf{P}(0) \cdot \frac{1}{L+1} \leqslant \frac{\varepsilon}{L+1}.$$

Итак, мы доказали, что

$$\langle\widetilde{\eta}|\, \widetilde{H}\, |\widetilde{\eta}\rangle \leqslant \frac{\varepsilon}{L+1},$$

поэтому у H есть собственное число с такой же верхней оценкой.

Оценка собственного числа при ответе «нет». В этом случае нам нужно доказать, что все собственные числа велики. Пусть для любого вектора $|\xi\rangle$ вероятность ответа 1 не превосходит ε, т. е.

$$\langle\xi, 0|\, U^\dagger \Pi_1^{(0)} U\, |\xi, 0\rangle \geqslant 1 - \varepsilon.$$

Докажем, что в этом случае все собственные числа H больше либо равны $c(1-\sqrt{\varepsilon})L^{-3}$, где c — некоторая константа.

Доказательство довольно длинное, поэтому вначале приведём его краткий план. Представив гамильтониан в виде суммы $\widetilde{H} = A_1 + A_2$ операторов $A_1 = \widetilde{H}_{\mathrm{in}} + \widetilde{H}_{\mathrm{out}}$ и $A_2 = \widetilde{H}_{\mathrm{prop}}$, мы оценим снизу наименьшие ненулевые собственные числа A_1 и A_2 по отдельности. Получим оценки 1 и $c'L^{-2}$ соответственно. Чтобы оценить наименьшее собственное число $A_1 + A_2$, нам потребуется лемма, которая даёт такую оценку для суммы через оценки для слагаемых и угол между их нулевыми подпространствами. Углом между подпространствами $\mathcal{L}_1$ и $\mathcal{L}_2$ с нулевым пересечением будем называть величину $\vartheta(\mathcal{L}_1, \mathcal{L}_2)$, задаваемую условиями

$$\cos\vartheta(\mathcal{L}_1, \mathcal{L}_2) = \max_{\substack{|\eta_1\rangle \in \mathcal{L}_1 \\ |\eta_2\rangle \in \mathcal{L}_2}} \bigl|\langle\eta_1|\eta_2\rangle\bigr|, \qquad 0 < \vartheta(\mathcal{L}_1, \mathcal{L}_2) < \frac{\pi}{2}. \qquad (13.15)$$

Лемма 13.4. *Пусть A_1, A_2 — неотрицательные операторы, $\mathcal{L}_1$, $\mathcal{L}_2$ — их нулевые подпространства, причём $\mathcal{L}_1 \cap \mathcal{L}_2 = 0$. Пусть также ненулевые собственные числа A_1 и A_2 не меньше v. Тогда*

$$A_1 + A_2 \geqslant v \cdot 2\sin^2\frac{\vartheta}{2}, \qquad (13.16)$$

где $\vartheta = \vartheta(\mathcal{L}_1, \mathcal{L}_2)$ — угол между $\mathcal{L}_1$ и $\mathcal{L}_2$.

Обозначение $A \geqslant a$ (A — оператор, a — число) нужно понимать как сокращение от $A - aI \geqslant 0$. Другими словами, если $A \geqslant a$, то все собственные числа A не меньше a.

В нашем случае мы получим оценки 1 и $c'L^{-2}$ для ненулевых собственных чисел A_1 и A_2 (об этом уже говорилось выше) и $\sin^2\vartheta \geqslant (1-\sqrt{\varepsilon})/(L+1)$ для угла. Отсюда вытекает искомое неравенство

$$H \geqslant c(1-\sqrt{\varepsilon})L^{-3}.$$

Доказательство (леммы 13.4). Очевидно, что $A_1 \geqslant v(I - \Pi_{\mathcal{L}_1})$ и $A_2 \geqslant v(I - \Pi_{\mathcal{L}_2})$, поэтому достаточно доказать неравенство $(I - \Pi_{\mathcal{L}_1}) + (I - \Pi_{\mathcal{L}_2}) \geqslant 2\sin^2(\vartheta/2)$. Оно, в свою очередь, эквивалентно такому неравенству:

$$\Pi_{\mathcal{L}_1} + \Pi_{\mathcal{L}_2} \leqslant 1 + \cos\vartheta. \qquad (13.17)$$

Пусть $|\xi\rangle$ — собственный вектор оператора $\Pi_{\mathcal{L}_1} + \Pi_{\mathcal{L}_2}$, отвечающий собственному числу $\lambda > 0$. Тогда

$$\Pi_{\mathcal{L}_1}|\xi\rangle = u_1|\eta_1\rangle, \qquad \Pi_{\mathcal{L}_2}|\xi\rangle = u_2|\eta_2\rangle, \qquad u_1|\eta_1\rangle + u_2|\eta_2\rangle = \lambda|\xi\rangle,$$

где $|\eta_1\rangle \in \mathcal{L}_1$ и $|\eta_2\rangle \in \mathcal{L}_2$ — единичные векторы, а u_1 и u_2 — неотрицательные вещественные числа. Отсюда находим

$$\lambda = \langle\xi|\bigl(\Pi_{\mathcal{L}_1} + \Pi_{\mathcal{L}_2}\bigr)|\xi\rangle = u_1^2 + u_2^2,$$

$$\lambda^2 = \bigl(u_1\langle\eta_1| + u_2\langle\eta_2|\bigr)\bigl(u_1|\eta_1\rangle + u_2|\eta_2\rangle\bigr) = u_1^2 + u_2^2 + 2u_1u_2\,\mathrm{Re}\langle\eta_1|\eta_2\rangle.$$

Следовательно,

$$(1+x)\lambda - \lambda^2 = x(u_1 \pm u_2)^2 \geqslant 0, \qquad \text{где } x = \bigl|\mathrm{Re}\langle\eta_1|\eta_2\rangle\bigr|.$$

Таким образом, $\lambda \leqslant 1 + x \leqslant 1 + \cos\vartheta$. □

Теперь получим упомянутые выше оценки. Нулевые подпространства A_1 и A_2 представляются в виде

$$\begin{aligned} \mathcal{L}_1 = \mathcal{B}^{\otimes m} \otimes |0^{N-m}\rangle \otimes |0\rangle \oplus \mathcal{B}^{\otimes N} \otimes \mathbb{C}(|1\rangle, \ldots, |L-1\rangle) \oplus \\ \oplus U^\dagger\bigl(|1\rangle \otimes \mathcal{B}^{\otimes(N-1)}\bigr) \otimes |L\rangle \end{aligned} \qquad (13.18)$$

(последний сомножитель во всех слагаемых относится к пространству счётчика),

$$\mathcal{L}_2 = \mathcal{B}^{\otimes N} \otimes |\psi\rangle, \qquad (13.19)$$

где вектор $|\psi\rangle$ определён формулой (13.14).

Для оценки
$$A_1\big|_{\mathcal{L}_1^\perp} \geqslant 1 \tag{13.20}$$
достаточно заметить, что A_1 является суммой коммутирующих друг с другом проекторов, поэтому все собственные числа этого оператора целые.

Для оценки $A_2\big|_{\mathcal{L}_2^\perp}$ нужно найти первое положительное собственное число матрицы E. Собственные векторы и собственные числа E даются формулами
$$|\psi_k\rangle = \alpha_k \sum_{j=0}^{L} \cos\Big(q_k\big(j+\tfrac{1}{2}\big)\Big)\,|j\rangle, \qquad \lambda_k = 1-\cos q_k,$$
где $q_k = \pi k/(L+1)$ $(k=0,\ldots,L)$. Отсюда следует, что
$$A_2\big|_{\mathcal{L}_2^\perp} \geqslant 1-\cos\left(\frac{\pi}{L+1}\right) \geqslant c' L^{-2}. \tag{13.21}$$

Наконец, нужно оценить угол между подпространствами $\mathcal{L}_1$ и $\mathcal{L}_2$. Будем оценивать квадрат косинуса угла
$$\cos^2\vartheta = \max_{\substack{|\eta_1\rangle\in\mathcal{L}_1\\ |\eta_2\rangle\in\mathcal{L}_2}} \big|\langle\eta_1|\eta_2\rangle\big|^2 = \max_{|\eta_2\rangle\in\mathcal{L}_2} \langle\eta_2|\Pi_{\mathcal{L}_1}|\eta_2\rangle. \tag{13.22}$$
Представим $|\eta_2\rangle$ в виде $|\xi\rangle\otimes|\psi\rangle$. Проектор на $\mathcal{L}_1$ распадается на сумму трёх проекторов, в соответствии с (13.18). Совсем легко подсчитать вклад второго слагаемого, он равен $(L-1)/(L+1)$. Первое и третье слагаемые в сумме дают
$$\frac{1}{L+1}\langle\xi|\big(\Pi_{\mathcal{K}_1}+\Pi_{\mathcal{K}_2}\big)|\xi\rangle \leqslant \frac{1+\cos\varphi}{L+1},$$
где $\mathcal{K}_1=\mathcal{B}^{\otimes N}$, $\mathcal{K}_2 = U^\dagger\left(|1\rangle\otimes\mathcal{B}^{\otimes(N-1)}\right)$, а φ — угол между этими двумя подпространствами. (Здесь используется неравенство (13.17), полученное в ходе доказательства леммы 13.4).

Величина $\cos^2\varphi$ равна максимальной вероятности получения ответа 1 исходной схемой; по условию она не больше, чем ε. Получаем такую, продолжающую (13.22), оценку:
$$\langle\eta_2|\Pi_{\mathcal{L}_1}|\eta_2\rangle \leqslant \frac{L-1}{L+1}+\frac{1+\sqrt{\varepsilon}}{L+1} = 1-\frac{1-\sqrt{\varepsilon}}{L+1}.$$
Следовательно, $\sin^2\vartheta = 1-\cos^2\vartheta \geqslant (1-\sqrt{\varepsilon})/(L+1)$, как и утверждалось выше.

Реализация счётчика. Мы написали замечательный гамильтониан, почти удовлетворяющий требуемым свойствам. У него есть только один недостаток — он лишь $(\log L)$-локальный (в пространстве счётчика мы действуем на все q-биты).

Этот недостаток можно преодолеть, если вложить пространство счётчика в бóльшее пространство. Возьмём L q-битов, занумерованных от 1 до L. Искомое вложение $\mathbb{C}^{L+1}\to\mathcal{B}^{\otimes L}$ выглядит так:
$$|j\rangle \mapsto |\underbrace{1,\ldots,1}_{j},\underbrace{0,\ldots,0}_{L-j}\rangle.$$
Используемые в конструкции гамильтониана H операторы на пространстве счётчика заменяются согласно схеме
$$\begin{aligned}
&|0\rangle\langle 0| \text{ на } \Pi_1^{(0)}, && |0\rangle\langle 1| \text{ на } \big(|0\rangle\langle 1|\big)_1\Pi_2^{(0)},\\
&|j\rangle\langle j| \text{ на } \Pi_j^{(1)}\Pi_{j+1}^{(0)}, && |j-1\rangle\langle j| \text{ на } \Pi_{j-1}^{(1)}\big(|0\rangle\langle 1|\big)_j\Pi_{j+1}^{(0)},\\
&|L\rangle\langle L| \text{ на } \Pi_L^{(1)}, && |L-1\rangle\langle L| \text{ на } \Pi_{L-1}^{(1)}\big(|0\rangle\langle 1|\big)_L.
\end{aligned} \tag{13.23}$$
Теперь они 3-локальные (а сам гамильтониан, с учетом действия на q-биты исходной схемы, — 5-локальный).

Если говорить точнее, мы заменили гамильтониан H, действовавший на пространстве $\mathcal{L}=\mathcal{B}^{\otimes N}\otimes\mathbb{C}^{L+1}$, на новый гамильтониан H_{ext}, определенный на бóльшем пространстве $\mathcal{L}_{\text{ext}}=\mathcal{B}^{\otimes N}\otimes\mathcal{B}^{\otimes L}$. Оператор H_{ext} отображает подпространство $\mathcal{L}\subseteq\mathcal{L}_{\text{ext}}$ в себя и действует на нем так же, как H.

Теперь возникает новая проблема: что делать с лишними состояниями в расширенном пространстве счётчика? Мы справимся с этой проблемой, добавив ещё одно слагаемое к гамильтониану H_{ext}:
$$H_{\text{stab}} = I_{\mathcal{B}^{\otimes N}}\otimes\sum_{j=1}^{L-1}\Pi_j^{(0)}\Pi_{j+1}^{(1)}.$$

Нулевое подпространство оператора H_{stab} совпадает со старым рабочим пространством $\mathcal{L}$, поэтому дополнительное слагаемое не меняет верхней оценки минимального собственного числа при ответе «да».

При ответе «нет» требуемую нижнюю оценку для собственных чисел оператора $H_{\text{ext}}+H_{\text{stab}}$ можно получить следующим образом. Оба слагаемых оставляют инвариантным подпространство $\mathcal{L}$, поэтому можно оценивать независимо на $\mathcal{L}$ и его ортогональном дополнении $\mathcal{L}^\perp$. На $\mathcal{L}$ имеем $H_{\text{ext}}\geqslant c(1-\sqrt{\varepsilon})L^{-3}$ и $H_{\text{stab}}=0$, а на $\mathcal{L}^\perp$ — $H_{\text{ext}}\geqslant 0$ и $H_{\text{stab}}\geqslant 1$. (Здесь мы пользуемся тем, что каждое из слагаемых гамильтониана, (13.8), (13.9) и (13.10), остается неотрицательным при замене (13.23)). В любом случае
$$H_{\text{ext}}+H_{\text{stab}} \geqslant c(1-\sqrt{\varepsilon})L^{-3}.$$
Это завершает доказательство полноты задачи о локальном гамильтониане в классе BQNP. $\square$

13.4. Место BQNP среди других сложностных классов. Прямо из определения следует, что класс BQNP содержит класс MA (а, значит, и BPP, и NP). Ничего более определённого о силе «недетерминированных» квантовых алгоритмов сказать пока нельзя.[15]

Не слишком много можно сказать и об их «слабости».

Утверждение 13.5. BQNP $\subseteq$ PSPACE.

Доказательство. Максимальная вероятность того, что подсказка Мерлина будет принята Артуром, равна масимальному собственному числу оператора $X = X^{(1)}$ (см. формулу (13.3)). Нам нужно вычислить эту величину с точностью $O(n^{-\alpha})$ $(\alpha > 0)$.

Заметим, что $0 \leqslant X \leqslant 1$. Для оценки максимального собственного числа будем использовать следующее предельное равенство:

$$\ln \lambda_{\max} = \lim_{d\to\infty} \frac{\ln \operatorname{Tr} X^d}{d}.$$

Пусть $\lambda_{\max} = \lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_{2^m}$ — собственные числа оператора X (здесь $m = \operatorname{poly}(n)$ — длина подсказки). Имеем оценку

$$\ln \lambda_{\max} \leqslant \frac{\ln \operatorname{Tr} X^d}{d} = \frac{\ln \sum\limits_{j=1}^{2^m} \lambda_j^d}{d} \leqslant \ln \lambda_{\max} + \frac{m}{d} \ln 2,$$

из которой следует, что для оценки с полиномиальной точностью неотрицательного оператора, действующего на m q-битах, достаточно вычислить след от его степени, ограниченной полиномом от m.

Вычисление величины $\operatorname{Tr} X^d$ делается на полиномиальной памяти тем же способом, что и моделирование работы квантовой схемы. $\square$

Замечание 13.3. Полученный результат можно усилить: BQNP $\subseteq$ PP. Доказательство полностью аналогично решению задачи 8.3.

Замечание 13.4. Мы ограничились случаем игр Мерлина и Артура, которые продолжаются один раунд. Недавно было показано [45], что уже двух раундов такой квантовой игры достаточно, чтобы получить весь класс PSPACE. В классическом случае для достижения класса PSPACE требуется полиномиальное количество раундов [36, 37], причём в широких кругах узких специалистов господствует мнение, что никакого фиксированного количества раундов недостаточно.

[15] Предупреждение: в литературе встречается другое определение квантового недетерминированного вычисления, для которого получена полная характеризация в терминах классических сложностных классов (см. [46]).

14. Классические и квантовые коды

Как уже обсуждалось ранее, квантовое вычисление «не слишком» чувствительно к погрешностям реализации унитарных операторов: ошибки накапливаются линейно. Если есть последовательность унитарных операторов $U_1, \ldots, U_L$ и последовательность приближений $\tilde U_1, \ldots$ $\ldots, \tilde U_L$, $\|\tilde U_j - U_j\| < \delta$, то выполняется неравенство

$$\|\tilde U_L \cdot \ldots \cdot \tilde U_1 - U_L \cdot \ldots \cdot U_1\| < L\delta.$$

Отсюда легко заключить, насколько изменится вероятность получения правильного ответа $F(x)$ квантовой схемой $U = U_L \cdot \ldots \cdot U_1$ (см. определение 8.1). Эта вероятность (левая часть неравенства в определении) может быть записана как $\langle \xi | U^\dagger \Pi_{\mathcal{M}} U | \xi \rangle$, где $|\xi\rangle = |x, 0^{N-n}\rangle$, а $\mathcal{M} = |F(x)\rangle \otimes \mathcal{B}^{\otimes (N-m)}$. Имеет место следующая оценка:

$$\bigl| \langle \xi | \tilde U^\dagger \Pi_{\mathcal{M}} \tilde U | \xi \rangle - \langle \xi | U^\dagger \Pi_{\mathcal{M}} U | \xi \rangle \bigr| \leqslant 2 \|\tilde U - U\| \leqslant 2L\delta.$$

Таким образом, при неточной реализации унитарных операторов правильный ответ получается с вероятностью $\geqslant 1 - \varepsilon - 2L\delta$; в общем случае эта оценка неулучшаема.

С точки зрения физической реализации квантового компьютера, полученный результат не является удовлетворительным. Получается, что размер квантовой схемы L не должен превосходить $1/(4\delta)$, иначе вероятность правильного ответа может стать меньше $1/2$. Поэтому возникает важный вопрос: можно ли избежать накопления ошибок, используя схемы специального вида?

Ответ на этот вопрос положительный. Идея состоит в том, чтобы *закодировать* (заменить) каждый q-бит, использующийся в вычислениях, несколькими при помощи определённого изометрического вложения $V\colon \mathcal{B} \to \mathcal{B}^{\otimes n}$. Дело в том, что ошибки, как правило, действуют одновременно на небольшое число q-битов, поэтому кодирование повышает устойчивость квантового состояния.

Конструкции, необходимые для организации вычислений без потери точности, довольно сложны. Подробно они изложены в [41, 19, 34, 4, 32], а здесь мы в основном ограничимся более простым вопросом: как сохранять неограниченно долго заданное квантовое состояние? (Легко понять, что это — частный случай предыдущего вопроса, когда реализуется последовательность тождественных операторов.) Для решения такой упрощённой задачи конкретный вид кодирующего отображения V неважен; нужно задать лишь подпространство $\mathcal{M} = \operatorname{Im} V \subseteq \mathcal{B}^{\otimes n}$.

Определение 14.1. Квантовый код типа (n, m) — это подпространство $\mathcal{M} \subseteq \mathcal{B}^{\otimes n}$ размерности 2^m. (Число m — количество закодированных q-битов — не обязательно должно быть целым).

Ошибки, возникающие при хранении информации, будут приводить к тому, что состояние системы будет выходить за пределы $\mathcal{M}$. Поэтому необходимо научиться восстанавливать состояние системы после воздействия ошибок определённого типа.

14.1. Классические коды. Вначале рассмотрим случай классических кодов. Мы лишь слегка затронем эту обширную тему. Подробное изложение теории кодов, корректирующих ошибки, (так обычно называется эта наука) можно найти в [9].

Классический код типа (n, m) — это подмножество $M \subseteq \mathbb{B}^n$ мощности 2^m. Для описания ошибок необходимо также определить *канал связи* — нечто вроде неоднозначного отображения $\mathbb{B}^n \to \mathbb{B}^{n'}$. Существует две модели ошибок: более реалистичная — вероятностная, и упрощённая — теоретико-множественная. Согласно вероятностной модели, канал связи задается условными вероятностями $p(y\,|\,x)$ приема слова y при передаче слова x. Мы будем рассматривать случай независимо распределённых ошибок, полагая что $n' = n$, а условные вероятности определяются через вероятность ошибки при передаче одного бита p_1:

$$p(y\,|\,x) = p_1^{d(x,y)}(1-p_1)^{n-d(x,y)}. \qquad (14.1)$$

Здесь $d(x, y)$ — *расстояние Хэмминга* (число различных битов).

Есть стандартный способ упростить модель независимо распределённых ошибок. Оценим вероятность того, что случится более k ошибок (как ясно из формулы (14.1), эта величина от x не зависит). Считаем, что n, k — фиксированы, $p_1 \to 0$. Тогда

$$\mathbf{Pr}[\text{число ошибок } > k] = \sum_{j>k} \binom{n}{j} p_1^j (1-p_1)^{n-j} = o(p_1^k). \qquad (14.2)$$

Итак, вероятность того, что число ошибок больше k, мала. Поэтому можно сильно упростить модель. Будем считать, что при передаче слова x может получиться любое слово y, такое что $d(x, y) \leqslant k$ (параметр k задаёт интересующий нас порог точности), а другие ошибки не встречаются.

Введём обозначения:

$N = \mathbb{B}^n$	—	множество входов,
$N' = \mathbb{B}^{n'}$	—	множество выходов,
$E \subseteq N \times N'$	—	множество переходов (оно же — множество ошибок),
$E(n, k)$	—	множество $\{(x, y) : d(x, y) \leqslant k\}$.

Определение 14.2. Код M *исправляет ошибки* из множества E, если для любых $x_1, x_2 \in M$ из $(x_1, y) \in E$ и $(x_2, y) \in E$ следует $x_1 = x_2$.

Другими словами это условие можно сформулировать так: для любых пар (x_1, y_1), (x_2, y_2), принадлежащих E, из $x_1, x_2 \in M$ и $x_1 \neq x_2$ следует $y_1 \neq y_2$.

В том случае, когда $E = E(n, k)$, говорят, что *код исправляет k ошибок*.

Замечание. Термин «код, исправляющий ошибки» является неточным. Правильнее было бы сказать, что код оставляет возможность для исправления ошибок. *Исправляющие преобразование* — это отображение $P\colon N' \to N$, такое что, если $(x, y) \in E$ и $x \in M$, то $P(y) = x$, вычисление значения исправляющего преобразования называется *декодированием*.

Пример 14.1. Код с повторением:

$$M_3 = \{(0,0,0), (1,1,1)\} \subseteq \mathbb{B}^3.$$

Такой код исправляет одну ошибку.

Очевидное обобщение примера 14.1 приводит к классическим кодам, исправляющим любое количество ошибок. Построим более интересные примеры классических кодов. Для начала дадим ещё одно стандартное определение.

Определение 14.3. Кодовое расстояние — это

$$d(M) = \min\{d(x_1, x_2) : x_1, x_2 \in M;\ \ x_1 \neq x_2\}.$$

Для кода из примера 14.1 кодовое расстояние равно 3. Имеется очевидное утверждение.

Утверждение 14.1. *Код исправляет k ошибок тогда и только тогда, когда $d(M) > 2k$.*

14.2. Примеры классических кодов.

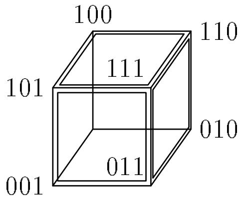
1. M_n типа $(n, 1)$; для него $d(M_n) = n$.

$$M_n = \{(\underbrace{0, \ldots, 0}_{n}),\ (\underbrace{1, \ldots, 1}_{n})\}.$$

   Это самая простая схема кодирования. Повторяем каждый бит много раз, а после каждой операции восстанавливаем кодовое слово, заменяя значения битов на то, которое встречается чаще.
   Эта серия кодов, как будет показано ниже, не обобщается на квантовый случай.

2. Проверка на чётность. Код $M_n^{(2)}$ типа $(n,\, n-1)$, для него $d(M_n^{(2)}) = 2$. Состоит из всех *чётных* слов, т.е. слов, содержащих чётное число единиц.

3. Код Хэмминга H_r. Это код типа $(n,\ n-r)$, где число $n = 2^r - 1$.



Слова из $\mathbb{B}^n$ — это последовательности битов $x = (x_\alpha : \alpha = 1, \ldots, n)$. Номер каждого бита можно записать в двоичной системе как $\alpha = (\alpha_1, \ldots, \alpha_n)$. Введём множество контрольных сумм

$$\mu_j(x) = \sum_{\alpha : \alpha_j = 1} x_\alpha$$

(суммирование здесь понимается по модулю 2). На рисунке выделены множества битов, входящих в контрольные суммы при $r = 3$ (полезно видеть в этой картинке трёхмерный куб).

Множество слов кода Хэмминга задаётся условием равенства всех контрольных сумм 0 (т.е. оно является подпространством по модулю 2). Можно показать, что для кода Хэмминга $d = 3$.

14.3. Линейные коды. Пусть есть множество $N = \mathbb{B}^n = \mathbb{F}_2^n$. *Линейный код* $M \subseteq N$ — это линейное подпространство. Линейные коды удобно задавать двойственным базисом (как множество решений системы линейных уравнений).

Пример 14.2. Код Хэмминга, рассмотренный выше, задаётся как множество решений системы уравнений

$$\begin{cases} x_{100} + x_{101} + x_{110} + x_{111} = \mu_1(x) = 0, \\ x_{010} + x_{011} + x_{110} + x_{111} = \mu_2(x) = 0, \\ x_{001} + x_{011} + x_{101} + x_{111} = \mu_3(x) = 0. \end{cases}$$

Поэтому код Хэмминга образует подпространство коразмерности 3.

14.4. Квантовые коды. Будем давать определения аналогично классическому случаю. Набору условных вероятностей $\big(p(y|x) : x \in N,\ y \in N'\big)$ соответствует физически реализуемое преобразование матриц плотности $T\colon \mathbf{L}(\mathcal{N}) \to \mathbf{L}(\mathcal{N}')$. Имеет смысл и упрощённая модель: по аналогии с множеством переходов $E \subseteq N \times N'$ определим *пространство ошибок* — произвольное линейное пространство $\mathcal{E} \subseteq \mathbf{L}(\mathcal{N}, \mathcal{N}')$. (Таким образом, квантовая ошибка — это любой линейный оператор $\mathcal{N} \to \mathcal{N}'$). Есть и прямой аналог множества $E(n,k)$. Рассмотрим $\mathcal{N} = \mathcal{N}' = \mathcal{B}^{\otimes n}$. Через $\mathcal{E}[A]$ обозначим те ошибки, которые действуют на q-битах из множества A и не действуют на остальных q-битах, т.е. $\mathcal{E}[A] = \mathbf{L}(\mathcal{B}^{\otimes A}) \otimes I_{\mathcal{B}^{\otimes[n]\setminus A}}$ (здесь и далее $[n]$ обозначает множество всех q-битов $\{1, \ldots, n\}$). Тогда полагаем

$$\mathcal{E}(n,k) = \sum_{|A| \leqslant k} \mathcal{E}[A].$$

(Здесь стоит просто сумма подпространств, не прямая.) В дальнейшем нас будет интересовать именно устойчивость к ошибкам из $\mathcal{E}(n,k)$.

Но перед тем, как заняться изучением кодов, устойчивых к ошибкам из $\mathcal{E}(n,k)$, рассмотрим аналог модели независимо распределённых ошибок в квантовом случае и его связь с ошибками из $\mathcal{E}(n,k)$.

14.5. Модель независимых ошибок в квантовом случае. Предположим, что на каждый q-бит действует одно и то же малое возмущение. Это означает, что на матрицу плотности рассматриваемой системы из n q-битов действует преобразование $T = (I+R)^{\otimes n}$, где R — «мало». В классическом случае малое возмущение означает малую вероятность ошибки. В квантовом случае малое возмущение меняет матрицы плотности «не слишком сильно». Чтобы придать этому выражению точный смысл, нужно ввести норму на матрицах плотности, характеризующую их близость, а затем — такую норму на преобразованиях матриц плотности, чтобы выполнялось условие: малое по норме преобразование переводит матрицу плотности в близкую к ней.

Начнём с того, что выясним, какие нормы пригодны для характеризации близости матриц плотности. Матрица плотности, как мы помним из раздела 9, задаёт вероятностное распределение на чистых состояниях. Вероятностные распределения естественно сравнивать в ℓ^1-норме: если $\boldsymbol{p} = (p_1, \ldots, p_n)$, $\boldsymbol{q} = (q_1, \ldots, q_n)$ — два распределения, то мерой их различия считаем $\sum_{j=1}^n |p_j - q_j| = \|\boldsymbol{p} - \boldsymbol{q}\|_1$. Дадим определение аналогичной нормы для матриц плотности.

Определение 14.4. Следовая норма оператора $A \in \mathbf{L}(\mathcal{N})$ равна

$$\|A\|_{\mathrm{tr}} = \mathrm{Tr}\left(\sqrt{A^\dagger A}\right). \tag{14.3}$$

Для эрмитова оператора следовая норма — это сумма модулей собственных чисел.

Задача 14.1. Проверьте, что (14.3) действительно определяет норму. Докажите, что

$$\|A\|_{\mathrm{tr}} = \sup_{X \neq 0} \frac{|\mathrm{Tr}\, AX|}{\|X\|} \tag{14.4}$$

($\|X\|$ — операторная норма, см. опр. 7.2 на с. 64).

Задача 14.2. Проверьте выполнение следующих свойств следовой нормы:

а) $\|AB\|_{\mathrm{tr}} \leqslant \|B\|\,\|A\|_{\mathrm{tr}}$, г) $\|\operatorname{Tr}_{\mathcal{M}} A\|_{\mathrm{tr}} \leqslant \|A\|_{\mathrm{tr}}$,

б) $\|BA\|_{\mathrm{tr}} \leqslant \|B\|\,\|A\|_{\mathrm{tr}}$, д) $\|A\otimes B\|_{\mathrm{tr}} = \|A\|_{\mathrm{tr}}\|B\|_{\mathrm{tr}}$.

в) $|\operatorname{Tr} A| \leqslant \|A\|_{\mathrm{tr}}$,

Следующая лемма показывает, почему можно рассматривать следовую норму для матриц плотности как аналог ℓ^1-нормы для вероятностных распределений.

Лемма 14.2. *Пусть $\mathcal{N} = \bigoplus_j \mathcal{N}_j$ — разложение пространства $\mathcal{N}$ в прямую сумму взаимно ортогональных подпространств. Тогда для любой пары матриц плотности ρ, γ*

$$\sum_j |\mathbf{P}(\rho,\mathcal{N}_j) - \mathbf{P}(\gamma,\mathcal{N}_j)| \;\leqslant\; \|\rho-\gamma\|_{\mathrm{tr}}\,.$$

Доказательство. Левую часть этого неравенства можно представить в виде $\operatorname{Tr}((\rho-\gamma)B)$, где $B = \sum_j(\pm\Pi_{\mathcal{N}_j})$. Ясно, что $\|B\|\leqslant 1$. Теперь применим представление следовой нормы в виде (14.4). □

Теперь кажется естественным определить норму преобразования матриц плотности аналогично операторной норме

$$\|T\|_1 = \sup_{X\neq 0}\frac{\|TX\|_{\mathrm{tr}}}{\|X\|_{\mathrm{tr}}} \qquad (14.5)$$

и мерить этой нормой малость возмущения. Однако использование нормы (14.5) оказывается неудобным, так как она не согласована с тензорным произведением. Поясним это подробнее. Чтобы иметь оценку, аналогичную оценке (14.2) в классическом случае, мы должны написать разложение

$$T = (I+R)^{\otimes n} = I + \sum_{|A|\leqslant k} R^{\otimes A}\otimes I_{[n]\setminus A} + \underbrace{\sum_{|A|>k} R^{\otimes A}\otimes I_{[n]\setminus A}}_{P} \qquad (14.6)$$

и оценить норму P при условии $\|P\|_1 < \delta$. Если бы выполнялось неравенство $\|T\otimes R\|_1 \leqslant \|T\|_1\|R\|_1$, можно было бы буквально повторить оценку (14.2). Однако это неравенство не всегда выполняется.

Пример 14.3. Рассмотрим преобразование

$$T\colon |j\rangle\langle k| \mapsto |k\rangle\langle j| \quad (j,k=0,1).$$

Очевидно, что $\|T\|_1 = 1$, однако $\|T\otimes I_{\mathbf{L}(\mathcal{B})}\|_1 = 2$. (Подействуйте преобразованием $T\otimes I_{\mathbf{L}(\mathcal{B})}$ на оператор $X=\sum_{j,k}|j,j\rangle\langle k,k|$.)

Оказывается (ниже это будет доказано), что патология примера 14.3 имеет ограничение по размерности. А именно, если $\dim\mathcal{G}\geqslant\dim\mathcal{N}$, то $\|T\otimes I_{\mathbf{L}(\mathcal{G})}\|_1 = \|T\|_\diamond$, где величина $\|T\|_\diamond$ от $\mathcal{G}$ не зависит. Прежде чем доказывать это утверждение, посмотрим на его следствия.

Во-первых, ясно, что определённая таким образом величина $\|T\|_\diamond$ является нормой.

Во-вторых, поскольку следовая норма мультипликативна относительно тензорного умножения, то $\|T\otimes R\|_1 \geqslant \|T\|_1\|R\|_1$. Поэтому $\|T\|_\diamond \geqslant \|T\|_1$.

В-третьих, из определения следует, что $\|TR\|_1\leqslant\|T\|_1\|R\|_1$, поэтому имеем такие неравенства

$$\|T\|_1\|R\|_1 \leqslant \|T\otimes R\|_1 = \|(T\otimes I)(I\otimes R)\|_1 \leqslant \|T\otimes I\|_1\|I\otimes R\|_1. \quad (14.7)$$

Из этих неравенств следует мультипликативность нормы $\|\cdot\|_\diamond$ относительно тензорного умножения.

Чтобы доказать приведённое выше свойство норм $\|T\otimes I\|_1$, дадим другое определение величины $\|T\|_\diamond$.

Определение 14.5. Рассмотрим представления $T\colon \mathbf{L}(\mathcal{N})\to\mathbf{L}(\mathcal{N}')$ в виде $T=\operatorname{Tr}_{\mathcal{F}} A\cdot B^\dagger$. Здесь $A\cdot B^\dagger$ обозначает преобразование $X\mapsto AXB^\dagger$, а $A,B\in\mathbf{L}(\mathcal{N},\mathcal{N}'\otimes\mathcal{F})$, где $\mathcal{F}$ — произвольное унитарное пространство размерности не меньшей, чем $(\dim\mathcal{N})(\dim\mathcal{N}')$. Тогда $\|T\|_\diamond$ — точная нижняя грань чисел вида $\|A\|\,\|B\|$ (это операторные нормы) по всем представлениям указанного вида.

Замечание. Условие $\dim\mathcal{F}\geqslant(\dim\mathcal{N})(\dim\mathcal{N}')$ гарантирует, что существует хотя бы одно представление $T=\operatorname{Tr}_{\mathcal{F}} A_0\cdot B_0^\dagger$. Для минимизации произведения $\|A\|\,\|B\|$ достаточно рассматривать операторы с нормами $\|A\|\leqslant\|A_0\|$ и $\|B\|\leqslant\|B_0\|$, поэтому инфимум достигается в силу компактности. То, что он не зависит от размерности $\mathcal{F}$, вытекает из следующей теоремы.

Теорема 14.1. *Если $\dim\mathcal{G}\geqslant\dim\mathcal{N}$, то $\|T\|_\diamond = \|T\otimes I_{\mathbf{L}(\mathcal{G})}\|_1$.*

Доказательство. Пусть $TX = \operatorname{Tr}_{\mathcal{F}} AXB^\dagger$. Тогда, используя свойства следовой нормы из задачи 14.2, получаем

$$\|(T\otimes I_{\mathbf{L}(\mathcal{G})})X\|_{\mathrm{tr}} = \|\operatorname{Tr}_{\mathcal{F}}(A\otimes I_{\mathcal{G}})X(B^\dagger\otimes I_{\mathcal{G}})\|_{\mathrm{tr}} \leqslant$$
$$\leqslant \|(A\otimes I_{\mathcal{G}})X(B^\dagger\otimes I_{\mathcal{G}})\|_{\mathrm{tr}} \leqslant \|A\|\,\|B\|\,\|X\|_{\mathrm{tr}}.$$

Поэтому $\|T\|_\diamond \geqslant \|T\otimes I_{\mathbf{L}(\mathcal{G})}\|_1$.

Доказать неравенство в обратную сторону несколько сложнее. Без уменьшения общности, $\|T\|_\diamond = 1$. Инфимум в определении 14.5 достигается при $\|A\|=\|B\|=1$.

Покажем сначала, что существуют три матрицы плотности $\rho, \gamma \in \mathbf{L}(\mathcal{N})$ и $\tau \in \mathbf{L}(\mathcal{F})$, такие что $\mathrm{Tr}_{\mathcal{N}'}(A\rho A^\dagger) = \mathrm{Tr}_{\mathcal{N}'}(B\gamma B^\dagger) = \tau$. Пусть

$$\mathcal{K} = \mathrm{Ker}(A^\dagger A - I_{\mathcal{N}}), \qquad \mathcal{L} = \mathrm{Ker}(B^\dagger B - I_{\mathcal{N}}),$$

$$E = \Big\{\mathrm{Tr}_{\mathcal{N}'}(A\rho A^\dagger) : \rho \in \mathbf{D}(\mathcal{K})\Big\}, \qquad F = \Big\{\mathrm{Tr}_{\mathcal{N}'}(B\gamma B^\dagger) : \gamma \in \mathbf{D}(\mathcal{L})\Big\},$$

где $\mathbf{D}(\mathcal{L})$ обозначает множество матриц плотности на пространстве $\mathcal{L}$. Тогда $E, F \subseteq \mathbf{D}(\mathcal{F})$, поэтому в качестве τ годится любой элемент из $E \cap F$.

Докажем, что $E \cap F \neq \varnothing$. Так как E и F — компактные выпуклые множества, достаточно доказать, что не существует разделяющей их гиперплоскости. Другими словами, нет такого эрмитова оператора $Z \in \mathbf{L}(\mathcal{F})$, что $\mathrm{Tr}\, XZ > \mathrm{Tr}\, YZ$ для любых $X \in E$, $Y \in F$. А это, в свою очередь, следует из минимальности величины $\|A\|\,\|B\|$ по отношению к преобразованию

$$A \mapsto (I_{\mathcal{N}'} \otimes e^{-tZ})\, A, \qquad B \mapsto (I_{\mathcal{N}'} \otimes e^{tZ})\, B,$$

где t положительно, но мало.

Итак, пусть $\mathrm{Tr}_{\mathcal{N}'}(A\rho A^\dagger) = \mathrm{Tr}_{\mathcal{N}'}(B\gamma B^\dagger) = \tau \in \mathbf{D}(\mathcal{F})$, где $\rho, \gamma \in \mathbf{D}(\mathcal{N})$. Представим ρ и γ в виде $\rho = \mathrm{Tr}_{\mathcal{G}}\Big(|\xi\rangle\langle\xi|\Big)$, $\gamma = \mathrm{Tr}_{\mathcal{G}}\Big(|\eta\rangle\langle\eta|\Big)$, где $|\xi\rangle, |\eta\rangle \in \mathcal{N} \otimes \mathcal{G}$ — единичные векторы. Здесь мы используем условие $\dim \mathcal{G} \geqslant \dim \mathcal{N}$ и утверждение 9.1. Положим $X = |\xi\rangle\langle\eta|$. Очевидно, что $\|X\|_{\mathrm{tr}} = 1$.

Докажем, что $\|(T \otimes I_{\mathbf{L}(\mathcal{G})})X\|_{\mathrm{tr}} \geqslant 1$. Обозначим

$$X' = (T \otimes I_{\mathbf{L}(\mathcal{G})})X, \qquad |\xi'\rangle = (A \otimes I_{\mathcal{G}})|\xi\rangle, \qquad |\eta'\rangle = (B \otimes I_{\mathcal{G}})|\eta\rangle,$$

тогда

$$X' = \mathrm{Tr}_{\mathcal{F}}(|\xi'\rangle\langle\eta'|), \qquad \mathrm{Tr}_{\mathcal{N}'\otimes\mathcal{G}}(|\xi'\rangle\langle\xi'|) = \mathrm{Tr}_{\mathcal{N}'\otimes\mathcal{G}}(|\eta'\rangle\langle\eta'|) = \tau.$$

Отсюда, во-первых, следует, что векторы $|\xi'\rangle$ и $|\eta'\rangle$ имеют единичную длину. Во-вторых, найдётся унитарный оператор U на пространстве $\mathcal{N}' \otimes \mathcal{G}$, такой что $(U \otimes I_{\mathcal{F}})|\xi'\rangle = |\eta'\rangle$ (это утверждение из задачи 9.3). Следовательно,

$$\|X'\|_{\mathrm{tr}} \geqslant \frac{|\mathrm{Tr}\, UX'|}{\|U\|} = \big|\mathrm{Tr}\,\big((U \otimes I_{\mathcal{F}})|\xi'\rangle\langle\eta'|\big)\big| = |\mathrm{Tr}(|\eta'\rangle\langle\eta'|)| = 1. \qquad \square$$

Теперь можно оценить остаточный член P в формуле (14.6), почти дословно повторяя рассуждения в классическом случае. Если $\|R\|_1 < \delta$, то $\|R\|_\diamond < C\delta$, где C — константа (все нормы эквивалентны, причём множитель ограничен размерностью пространства; R действует на пространстве матриц плотности одного q-бита). Используя мультипликативность нормы $\|\cdot\|_\diamond$, заключаем, что

$$\|P\|_1 \leqslant \|P\|_\diamond = o(\delta^k).$$

Поэтому будем пренебрегать всеми слагаемыми из P, а действие всех остальных слагаемых будем учитывать. Преобразования $R^{\otimes A}$ имеют вид $R^{\otimes A}\rho = \sum\limits_i X_i \rho Y_i^\dagger$, где X_i, $Y_i \in \mathcal{E}(A)$; символически это можно записать так: $R^{\otimes A} \in \mathcal{E}(A)\cdot\mathcal{E}^\dagger(A)$. Сумма преобразований $\sum_{|A|\leqslant k} R_1^{\otimes A} \otimes I_{[n]\setminus A}$ лежит в $\mathcal{E}(n,k)\cdot\mathcal{E}^\dagger(n,k)$. Таким образом, мы приходим к выводу, что нужно рассматривать ошибки из $\mathcal{E}(n,k)$.

14.6. Основные определения и простейшие следствия. Следующее определение даёт в квантовом случае формальное выражение требования «различные состояния переходят в различные состояния» (это необходимое условие возможности восстановления исходных состояний физически реализуемым преобразованием).

Определение 14.6. Квантовый код (подпространство $\mathcal{M} \subseteq \mathcal{N}$) исправляет ошибки из $\mathcal{E} \subseteq \mathbf{L}(\mathcal{N}, \mathcal{N}')$, если

$$\forall\, |\xi_1\rangle, |\xi_2\rangle \in \mathcal{M}\,\forall\, X, Y \in \mathcal{E}\ \ \big(\langle\xi_2|\xi_1\rangle = 0\big) \Rightarrow \big(\langle\xi_2|Y^\dagger X|\xi_1\rangle = 0\big). \qquad (14.8)$$

Определение 14.7. Физически реализуемое преобразование

$$P \colon \mathbf{L}(\mathcal{N}') \to \mathbf{L}(\mathcal{M})$$

называется *исправляющим* (для кода $\mathcal{M}$ и пространства ошибок $\mathcal{E}$), если

$$\forall\, T \in \mathcal{E}\cdot\mathcal{E}^\dagger\ \exists\, c = c(T)\ \forall\, \rho \in \mathbf{L}(\mathcal{M})\ \big(PT\rho = c(T)\rho\big).$$

Если при этом T сохраняет след, то $c(T) = 1$.

Теорема 14.2. *Если код $\mathcal{M}$ исправляет ошибки из $\mathcal{E}$, то исправляющее преобразование существует.*

Доказательство будет дано ниже. Обратное утверждение доказано в [4].

Пример 14.4. Тривиальный код типа (n,m): пусть $\mathcal{M} = \mathcal{B}^{\otimes m} \otimes |0^{n-m}\rangle$, а $\mathcal{E} = \mathcal{E}[m+1,\ldots,n]$, т.е. для кодирования используются первые m q-битов, а ошибки действуют на остальные q-биты. Условие (14.8), очевидно, выполнено. В качестве исправляющего преобразования можно взять $P = I_{\mathbf{L}(\mathcal{B}^{\otimes m})} \otimes R$, где $R\colon X \mapsto (\mathrm{Tr}\, X)|0^{n-m}\rangle\langle 0^{n-m}|$. Преобразование P реализуется очень просто: выбрасываем последние $n-m$ q-битов и заменяем их на новые q-биты в состоянии $|0\rangle$. Практической пользы от такого кода, конечно, мало. Интересно, однако, что любой квантовый код, исправляющий ошибки, в определённом смысле похож на тривиальный (см. лемму 14.3 ниже).

Пример 14.5. Рассмотрим квантовый аналог кода с повторением. Пусть пространство $\mathcal{M} = \mathbb{C}(|0,\ldots,0\rangle, |1,\ldots,1\rangle)$. Рассмотрим два состояния $|\xi_1\rangle = |0,\ldots,0\rangle + |1,\ldots,1\rangle$ и $|\xi_2\rangle = |0,\ldots,0\rangle - |1,\ldots,1\rangle$. Ошибку выберем так: $X = I$, $Y = \sigma^z[1]$. Очевидно, что $X, Y \in \mathcal{E}(n,1)$. При этом $Y|\xi_2\rangle = X|\xi_1\rangle$, что противоречит определению кода, исправляющего ошибки. Мы видим, что код с произвольно большим повторением не защищает даже от одной ошибки.

Ошибки вида $\sigma^x[1]$ называются *классическими*, а ошибки вида $\sigma^z[1]$ называются *фазовыми*.

В определении 14.6 речь шла только о парах ортогональных состояний. Давайте посмотрим, что получается на произвольных парах. Зафиксируем $X, Y \in \mathcal{E}$ и обозначим $Z = Y^\dagger X$. Оказывается, что

$$\forall\, |\xi_1\rangle, |\xi_2\rangle \in \mathcal{M}\ \langle\xi_2|Z|\xi_1\rangle = c(Z)\langle\xi_2|\xi_1\rangle, \tag{14.9}$$

где $c(Z)$ — некоторое комплексное число, не зависящее от $|\xi_1\rangle, |\xi_2\rangle$. Действительно, пусть $|\eta_1\rangle, \ldots, |\eta_m\rangle$ — ортонормированный базис пространства $\mathcal{M}$. По определению 14.6 $\langle\eta_j|Z|\eta_k\rangle = 0$ при $j \neq k$, а $\langle\eta_j|Z|\eta_j\rangle$ не зависит от j, так как

$$\langle\eta_j|Z|\eta_j\rangle - \langle\eta_k|Z|\eta_k\rangle = \langle\eta_j - \eta_k|Z|\eta_j + \eta_k\rangle + \langle\eta_k|Z|\eta_j\rangle - \langle\eta_j|Z|\eta_k\rangle = 0.$$

(Все три слагаемых в правой части равенства равны нулю, так как входящие в них пары векторов ортогональны.)

Заметим, что если $X, Y \in \mathcal{E}(n,k)$, то $Z = Y^\dagger X \in \mathcal{E}(n, 2k)$.

Определение 14.8. Код $\mathcal{M}$ *обнаруживает* ошибки из $\mathcal{E}' \subseteq \mathbf{L}(\mathcal{N})$, если

$$\forall\, |\xi_1\rangle, |\xi_2\rangle \in \mathcal{M}\ \forall\, Z \in \mathcal{E}'\ \langle\xi_2|Z|\xi_1\rangle = c(Z)\langle\xi_2|\xi_1\rangle.$$

Кодовым расстоянием называется наименьшее число $d = d(\mathcal{M})$, при котором код не обнаруживает ошибки из $\mathcal{E}(n,d)$.

Таким образом, код исправляет k ошибок, если $2k < d(\mathcal{M})$.

Теперь мы перейдем к доказательству теоремы 14.2.

Лемма 14.3. *Пусть квантовый код $\mathcal{M} \subseteq \mathcal{N}$ исправляет ошибки из $\mathcal{E} \subseteq \mathbf{L}(\mathcal{N}, \mathcal{N}')$. Тогда существует унитарное пространство $\mathcal{F}$, изометрическое вложение $V\colon \mathcal{M}\otimes\mathcal{F} \to \mathcal{N}'$ и линейное отображение $f\colon \mathcal{E} \to \mathcal{F}$, такие что*

$$\forall\, |\xi\rangle \in \mathcal{M}\ \forall\, X \in \mathcal{E}\ X|\xi\rangle = V\big(|\xi\rangle \otimes |f(X)\rangle\big). \tag{14.10}$$

Доказательство. Пусть $\mathcal{E}_0 = \{X \in \mathcal{E} : \forall\, |\xi\rangle \in \mathcal{M}\ X|\xi\rangle = 0\}$. Рассмотрим фактор-пространство $\mathcal{F} = \mathcal{E}/\mathcal{E}_0$ и естественное отображение $f\colon \mathcal{E} \to \mathcal{F}$. Линейное отображение $V\colon \mathcal{M}\otimes\mathcal{F} \to \mathcal{N}'$, удовлетворяющее условию (14.10), строится каноническим образом; нужно лишь проверить его изометричность.

Скалярное произведение на пространстве $\mathcal{F}$ можно задать при помощи функции c из свойства кода (14.9): если $|\eta_1\rangle = |f(X)\rangle$ и $|\eta_2\rangle = |f(Y)\rangle$, то $\langle\eta_2|\eta_1\rangle = c(Y^\dagger X)$. Очевидно, что эта величина не зависит от выбора X и Y. Ясно также, что $\langle\eta|\eta\rangle > 0$, если $|\eta\rangle \neq 0$. Формула (14.9) как раз и означает, что отображение V является изометрическим. □

Доказательство (теоремы 14.2). Представим пространство $\mathcal{N}'$ как сумму взаимно ортогональных подпространств: $\mathcal{N}' = (\operatorname{Im} V) \oplus \mathcal{K}$, где V — отображение из предыдущей леммы. Пусть $W\colon \mathcal{K} \to \mathcal{N}'$ — каноническое вложение, а $R\colon \mathbf{L}(\mathcal{K}) \to \mathbf{L}(\mathcal{M})$ — произвольное физически реализуемое преобразование. Тогда мы можем определить

$$P\colon \rho \mapsto \operatorname{Tr}_{\mathcal{F}}(V^\dagger \rho V) + R(W^\dagger \rho W), \qquad c\colon X \cdot Y^\dagger \mapsto \langle f(Y)|f(X)\rangle.$$

(Функция c линейно продолжается на все пространство $\mathcal{E}\cdot\mathcal{E}^\dagger$). □

Лемму 14.3 и доказательство теоремы 14.2 можно неформально изложить таким образом. Код, исправляющий ошибки, характеризуется тем, что ошибка не смешивается с закодированной информацией, т.е. остаётся в виде отдельного тензорного сомножителя. Исправляющее преобразование извлекает эту «встроенную ошибку» и выбрасывает ее в мусорную корзину.

Задача 14.3. Пусть код $\mathcal{M} \subseteq \mathcal{B}^{\otimes n}$ обнаруживает ошибки из $\mathcal{E}(A)$. Докажите, что состояние $\rho \in \mathbf{L}(\mathcal{M})$ можно восстановить, не используя q-битов из множества A.

14.7. Код Шора [40]. Опишем серию кодов со сколь угодно большим кодовым расстоянием. Они используют $n = r^2$ кодовых q-битов и кодируют один q-бит (т.е. $\dim\mathcal{M} = 2$), а кодовое расстояние d равно r.

Поскольку количество кодовых q-битов — точный квадрат, удобно задавать базисные состояния в таком кодовом пространстве в виде матрицы. В этих обозначениях код Шора порождается векторами

$$|\xi_\alpha\rangle = \sum_{y_1,\ldots,y_r \in \mathbb{B}^r} (-1)^{\alpha \sum_{j=1}^r y_j} \left| \begin{matrix} y_1 & \cdots & \cdots & y_1 \\ y_2 & \cdots & \cdots & y_2 \\ \cdots & \cdots & \cdots & \cdots \\ y_r & \cdots & \cdots & y_r \end{matrix} \right\rangle. \tag{14.11}$$

Для анализа кода Шора нам потребуется классификация операторов с помощью матриц Паули. Их, вообще говоря, три, но четвёртой матрицей Паули будем считать единичную. Введём нестандартную индексацию матриц Паули:

$$\sigma_{00} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; \qquad \sigma_{01} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \sigma^z;$$

$$\sigma_{10} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma^x; \qquad \sigma_{11} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \sigma^y.$$

Матрицы Паули замечательны тем, что они эрмитовы и унитарные одновременно. Введённая индексация позволяет удобно записывать коммутационные соотношения между матрицами Паули

$$\sigma_{\alpha\beta}\sigma_{\alpha'\beta'} = (-i)^{\alpha\beta'-\alpha'\beta}\sigma_{\alpha\oplus\alpha',\beta\oplus\beta'}, \quad \sigma_{\alpha\beta}\sigma_{\alpha'\beta'} = (-1)^{\alpha\beta'-\alpha'\beta}\sigma_{\alpha'\beta'}\sigma_{\alpha\beta}. \tag{14.12}$$

Множество индексов образует группу $G = \mathbb{Z}_2 \oplus \mathbb{Z}_2$ или 2-мерное пространство над полем $\mathbb{F}_2$.

Матрицы Паули образуют базис пространства $\mathbf{L}(\mathcal{B})$:

$$\mathbf{L}(\mathcal{B}) = \mathbb{C}(\sigma_{00}) \oplus \mathbb{C}(\sigma_{01}) \oplus \mathbb{C}(\sigma_{10}) \oplus \mathbb{C}(\sigma_{11}).$$

Для пространства $\mathcal{B}^{\otimes n}$ будет уже 4^n базисных операторов. Введём обозначение

$$\sigma(f) = \sigma(\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_n, \beta_n) \stackrel{\text{def}}{=} \sigma_{\alpha_1,\beta_1} \otimes \sigma_{\alpha_2,\beta_2} \otimes \cdots \otimes \sigma_{\alpha_n,\beta_n}.$$

Здесь $f \in G^n = \mathbb{F}_2^{2n}$.

Используя коммутационные соотношения, можно написать, с точностью до общего фазового множителя, $\sigma(f) = c\sigma(f^{(x)}) \cdot \sigma(f^{(z)})$, где $f^{(x)} = (\alpha_1, 0, \alpha_2, 0, \ldots)$ называется *классической ошибкой*, а $f^{(z)} = (0, \beta_1, 0, \beta_2, \ldots)$ — *фазовой ошибкой*.

Теперь проанализируем код Шора. В силу линейности определения достаточно ограничиться изучением базисных ошибок. Пусть $Z \in \mathcal{E}(r^2, k)$, $(k < r)$ и $Z = \sigma(f) = c\sigma(f^{(x)})\sigma(f^{(z)})$. Поскольку $|f| \leqslant k$ ($|f|$ — число ненулевых переменных в f), то $|f^{(x)}|$, $|f^{(z)}| \leqslant k < r$.

Достаточно показать, что в этом случае

$$\langle\xi_1|Z|\xi_0\rangle = 0, \quad \langle\xi_1|Z|\xi_1\rangle = \langle\xi_0|Z|\xi_0\rangle. \tag{14.13}$$

Рассмотрим два случая.

1. Классическая ошибка отлична от 0. В этом случае каждое базисное состояние

$$\left|\begin{matrix} y_1 & \ldots & \ldots & y_1 \\ y_2 & \ldots & \ldots & y_2 \\ \ldots & \ldots & \ldots & \ldots \\ y_r & \ldots & \ldots & y_r \end{matrix}\right\rangle$$

изменяется под действием Z в некоторых e битах, $0 < e < r$. Поэтому в скалярных произведениях (14.13) все слагаемые будут равны 0.

2. $f^{(x)} = 0$. Ошибка чисто фазовая: $Z = (\sigma^z)^{\beta_{11}} \otimes \cdots \otimes (\sigma^z)^{\beta_{rr}}$, где $\sum_{j,l}\beta_{jl} < r$. Обозначим $\lambda_j = \sum_l \beta_{jl}$. Тогда (см. (14.11))

$$Z|\xi_\alpha\rangle = \sum_{y_1,\ldots,y_r} (-1)^{\sum_j(\lambda_j+\alpha)y_j} \left|\begin{matrix} y_1 & \ldots & \ldots & y_1 \\ y_2 & \ldots & \ldots & y_2 \\ \ldots & \ldots & \ldots & \ldots \\ y_r & \ldots & \ldots & y_r \end{matrix}\right\rangle.$$

Нас интересуют значения λ_j по модулю 2. Возможны 3 случая:

а) $(\lambda_1, \ldots, \lambda_r) = (0, \ldots, 0) \pmod 2$.

б) $(\lambda_1, \ldots, \lambda_r) = (1, \ldots, 1) \pmod 2$.

в) $(\lambda_1, \ldots, \lambda_r) \neq (0, \ldots, 0), (1, \ldots, 1) \pmod 2$.

Случай б) в действительности реализоваться не может, так как $\sum_j \lambda_j \leqslant k < r$. В случае в) все скалярные произведения обращаются в нуль, $\langle\xi_\alpha|Z|\xi_{\alpha'}\rangle = 0$. В случае а) $Z|\xi_\alpha\rangle = |\xi_\alpha\rangle$, т.е. Z действует на кодовом подпространстве тождественным образом. (Такая ошибка, по существу, не является ошибкой, поскольку ничего не портит). Следовательно, $\langle\xi_\alpha|Z|\xi_{\alpha'}\rangle = \langle\xi_\alpha|\xi_{\alpha'}\rangle$.

Итак, код Шора обнаруживает $r-1$ ошибку; кодовое расстояние равно r.

Замечание. Код Шора основан на дуальности между классическими и фазовыми ошибками, которая выражается равенством $\sigma^z = H\sigma^x H^\dagger$. Внутри каждой строки $y_1, \ldots, y_r$ реализован обычный повторительный код, исправляющий классические ошибки. Строки организованы в аналогичный код, отличающийся заменой базиса в каждом q-бите: $\sum_{y_1,\ldots,y_r}(-1)^{\alpha(\sum_j y_j)}|y_1, \ldots, y_r\rangle = (H \otimes \cdots \otimes H)|\alpha, \ldots, \alpha\rangle$. Этот код исправляет фазовые ошибки.

14.8. Симплектические (стабилизирующие) коды [26]. Это аналог классических линейных кодов. Квантовые симплектические коды устроены так же, только вместо контрольных сумм будут использоваться σ-операторы ($\sigma(\alpha_1, \beta_1, \ldots, \alpha_n, \beta_n)$ в предыдущих обозначениях).

Зададим, например, таким способом код Шора. Для него $\mathcal{N} = \mathcal{B}^{\otimes r^2}$, $\dim\mathcal{M} = 2$. Кодовое подпространство порождено двумя векторами, см. (14.11).

Каким условиям удовлетворяют базисные векторы $\mathcal{M}$?

1. Для любых j, l $(l < r)$ выполнено $\sigma^z_{jl}\sigma^z_{j(l+1)}|\xi\rangle = |\xi\rangle$, т.е. каждая строка состоит из повторений одного бита.

2. Для любого $j < r$ выполнено $\prod_l\left(\sigma^x_{jl}\sigma^x_{(j+1)l}\right)|\xi\rangle = |\xi\rangle$. Что означает это условие? Оператор $\prod_l\left(\sigma^x_{jl}\sigma^x_{(j+1)l}\right)$ переводит

$$\text{базисный вектор} \left|\begin{matrix} \ldots\ldots\ldots \\ y_j \quad \ldots y_j \\ y_{j+1} \ldots y_{j+1} \\ \ldots\ldots\ldots \end{matrix}\right\rangle \text{ в вектор } \left|\begin{matrix} \ldots\ldots\ldots\ldots\ldots \\ y_j\oplus 1 \quad \ldots y_j\oplus 1 \\ y_{j+1}\oplus 1 \ldots y_{j+1}\oplus 1 \\ \ldots\ldots\ldots\ldots\ldots \end{matrix}\right\rangle$$

(остальные строки не меняются). Эти два вектора должны входить в $|\xi\rangle$ с одинаковыми коэффициентами.

Утверждение 14.4. *Если $|\xi\rangle$ удовлетворяет условиям 1 и 2, то* $|\xi\rangle = c_0|\xi\rangle_0 + c_1|\xi\rangle_0$.

Прежде чем перейти к изучению более общих симплектических кодов, рассмотрим подробнее свойства σ-операторов.

Как уже говорилось, σ-операторы удобно индексировать элементами группы $G = (\mathbb{Z}_2)^2$. Будем обозначать мультииндекс σ-оператора через $\gamma = (\alpha_1, \beta_1, \ldots, \alpha_n, \beta_n) \in G^n$.

σ-операторы образуют базис в $\mathbf{L}(\mathcal{B}^{\otimes n})$, более того, есть естественная G^n-градуировка:

$$\mathbf{L}(\mathcal{B}^{\otimes n}) = \bigoplus_{\gamma \in G^n} \mathbb{C}(\sigma(\gamma)).$$

Для σ-операторов выполнены следующие соотношения

$$\sigma(\gamma_1)\sigma(\gamma_2) = (-i)^{\tilde{\omega}(\gamma_1,\gamma_2)}\sigma(\gamma_1+\gamma_2) = (-1)^{\omega(\gamma_1,\gamma_2)}\sigma(\gamma_2)\sigma(\gamma_1),$$

где

$$\tilde{\omega}(\alpha_1,\beta_1,\ldots,\alpha_n,\beta_n;\alpha'_1,\beta'_1,\ldots,\alpha'_n,\beta'_n) = \sum_{j=1}^{n}(\alpha_j\beta'_j - \alpha'_j\beta_j) \bmod 4,$$

$$\omega(\gamma_1,\gamma_2) = \tilde{\omega}(\gamma_1,\gamma_2) \bmod 2.$$

Рассмотрим унитарные преобразования — действия унитарных операторов $X \mapsto UXU^\dagger$. Такое действие не меняется при домножении U на число, равное по модулю единице, поэтому группа унитарных преобразований имеет вид $\mathbf{UT}(\mathcal{B}^{\otimes n}) = \mathbf{U}(\mathcal{B}^{\otimes n})/\mathbf{U}(1)$. Отметим, что унитарные преобразования — это в точности автоморфизмы $*$-алгебры $\mathbf{L}(\mathcal{B}^{\otimes n})$.

Нас интересуют такие преобразования, для которых $U\sigma(\gamma)U^\dagger = \sigma(u(\gamma))\cdot c(\gamma)$ ($u\colon G^n \to G^n$ — некоторая функция). Оператор $U\sigma(\gamma)U^\dagger$ эрмитов, поэтому $c(\gamma) = \pm 1$, то есть мы можем написать

$$U\sigma(\gamma)U^\dagger = (-1)^{v(\gamma)}\,\sigma(u(\gamma)), \qquad u\colon G^n \to G^n, \quad v\colon G^n \to \mathbb{Z}_2. \quad (14.14)$$

Группа таких преобразований называется *расширенной симплектической группой* и обозначается $\mathrm{ESp}_2(n)$. Операторы из этой группы будем называть симплектическими. Приведём примеры.

1. σ-операторы. $\sigma(f)\sigma(\gamma)\sigma(f)^\dagger = \sigma(f)\sigma(\gamma)\sigma(f) = (-1)^{\omega(f,\gamma)}\sigma(\gamma)$. В данном случае $u(\gamma) = \gamma$.
2. Оператор $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Непосредственно проверяется, что

$$H\sigma^x H = \sigma^z, \quad H\sigma^z H = \sigma^x, \quad H\sigma^y H = -\sigma^y.$$

Таким образом, преобразование $H \cdot H^\dagger \in \mathrm{ESp}_2(1)$.

Можно показать, что $|\,\mathrm{ESp}_2(1)| = 24$. Если учитывать фазовые множители, то получилась бы *группа Клиффорда* из $24 \cdot 8 = 192$ элементов.

Основные свойства введённого отображения $u\colon G^n \to G^n$ таковы:

1. u линейно на G^n.
2. u сохраняет форму ω, т.е. $\omega(u(f), u(g)) = \omega(f, g)$.

Отображения с такими свойствами, как известно, называются симплектическими; они образуют *симплектическую группу* $\mathrm{Sp}_2(n)$. Таким образом, определён гомоморфизм $\theta\colon \mathrm{ESp}_2(n) \to \mathrm{Sp}_2(n)$.

Теорема 14.3. $\mathrm{Im}\,\theta = \mathrm{Sp}_2(n)$, $\mathrm{Ker}\,\theta = G^n$ (*ядро состоит из σ-операторов*). *Таким образом*, $\mathrm{ESp}_2(n)/G^n \cong \mathrm{Sp}_2(n)$.

Для понимания доказательства желательно знать что-нибудь про расширения и когомологии групп [15]. Читателю, незнакомому с этими понятиями, будет предложен «обходной путь» (см. ниже).

Доказательство. Преобразование (14.14) должно быть автоморфизмом $*$-алгебры $\mathbf{L}(\mathcal{B}^{\otimes n})$. Это имеет место тогда и только тогда, когда сохраняются правила умножения операторов $\sigma(\gamma)$. Это означает, что функция u обладает указанными свойствами, а v удовлетворяет уравнению

$$v(x+y) - v(x) - v(y) = w(x,y), \quad (14.15)$$

где $w(x,y) = \dfrac{\tilde{\omega}(u(x),u(y)) - \tilde{\omega}(x,y)}{2} \in \mathbb{Z}_2$.

В случае, когда u — тождественное отображение, правая часть уравнения (14.15) равна нулю. Решениями являются все линейные функции. Это доказывает, что $\mathrm{Ker}\,\theta = G^n$.

Утверждение $\mathrm{Im}\,\theta = \mathrm{Sp}_2(n)$ равносильно тому, что уравнение (14.15) имеет решение при любом u из $\mathrm{Sp}_2(n)$. Чтобы доказать это, заметим, что функция w обладает следующими свойствами:

$$w(y,z) - w(x+y,\, z) + w(x,\, y+z) - w(x,y) = 0, \quad (14.16)$$

$$w(x,y) = w(y,x), \quad (14.17)$$

$$w(x,x) = 0. \quad (14.18)$$

Формула (14.16) — это уравнение коцикла. Оно означает, что функция w задает структуру группы на декартовом произведении множеств $G^n \times \mathbb{Z}^2$ согласно правилу $(x,p)\cdot(y,q) = (x+y,\ p+q+w(x,y))$. Полученная группа (обозначим ее E) является расширением G^n посредством $\mathbb{Z}_2$, т.е. определён гомоморфизм $\lambda\colon E \to G^n$, $\lambda\colon (x,p) \mapsto x$ с ядром $\mathbb{Z}_2$.

Уравнение (14.17) означает, что группа E абелева. Наконец, уравнение (14.18) означает, что все элементы группы E имеют порядок 2 (или 1). Следовательно, $E \cong (\mathbb{Z}_2)^{2n+1}$. Отсюда вытекает, что расширение $E \to G^n$ тривиально: существует гомоморфизм $\mu\colon G^n \to E$, такой что $\lambda\mu = \mathrm{id}_{G^n}$. Записывая этот гомоморфизм в виде $\mu\colon x \mapsto (x, v(x))$, получаем решение уравнения (14.15). $\square$

Существует другой, несколько кустарный способ доказать, что $\operatorname{Im}\theta = \mathrm{Sp}_2(n)$. Рассмотрим следующие симплектические преобразования: $(H\cdot H^\dagger)[j]$, $(K\cdot K^\dagger)[j]$ и $\left(\Lambda(\sigma^x)\cdot\Lambda^\dagger(\sigma^x)\right)[j,k]$. (Напомним, что $K=\begin{pmatrix}1&0\\0&i\end{pmatrix}$). Их образы при гомоморфизме θ порождают всю группу $\mathrm{Sp}_2(n)$. (Намёк на доказательство: любую пару векторов $\gamma_1,\gamma_2\in G^n$, такую что $\omega(\gamma_1,\gamma_2)=1$, можно перевести этими преобразованиями в $(1,0,0,0,\ldots)$ и $(0,1,0,0,\ldots)$.) На самом деле, таким способом можно получить и другой интересный результат. Указанные элементы группы $\mathrm{ESp}_2(n)$ порождают все матрицы Паули, т.е. ядро гомоморфизма θ. Следовательно, верно такое утверждение.

Утверждение 14.5. *Группа* $\mathrm{ESp}_2(n)$ *порождается элементами*

$$(H\cdot H^\dagger)[j],\ (K\cdot K^\dagger)[j],\ \left(\Lambda(\sigma^x)\cdot\Lambda^\dagger(\sigma^x)\right)[j,k].$$

Для примера посмотрим на действие оператора $U=\Lambda(\sigma^x)[1,2]$. По определению имеем $U|a,b\rangle=|a,a\oplus b\rangle$. Действие U на образующие алгебры $\mathbf{L}(\mathcal{B}^{\otimes 2})$:

$$\begin{aligned}U\sigma_1^zU^\dagger&=\sigma_1^z,\\ U\sigma_2^zU^\dagger&=\sigma_1^z\sigma_2^z,\\ U\sigma_2^xU^\dagger&=\sigma_2^x,\\ U\sigma_1^xU^\dagger&=\sigma_1^x\sigma_2^x.\end{aligned}$$

Приведённые равенства можно без труда проверить прямым вычислением. Однако полезно привести объяснение «на пальцах». Оператор σ^z можно понимать как измерение значения соответствующего q-бита. Первые два равенства просто показывают, как меняются эти значения при замене базиса, задаваемой U. Третье равенство показывает, что изменение значения второго q-бита коммутирует с заменой базиса U. Четвёртое — что изменение значения первого q-бита при неизменном втором бите в повернутом базисе означает одновременное изменение значений обоих q-битов в исходном базисе.

Дадим теперь определение симплектического кода. В пространстве $\mathcal{N}=\mathcal{B}^{\otimes n}$ мы выделим подпространство $\mathcal{M}$ условиями $X_j|\xi\rangle=|\xi\rangle$ (X_j будем называть *проверочными операторами*). Проверочные операторы будут иметь вид

$$X_j=(-1)^{\varphi_j}\sigma(f_j),\quad f_j\in G^n,\quad \varphi_j\in\mathbb{Z}_2. \tag{14.19}$$

Без ограничения общности можно считать, что $\{f_j\}$ линейно независимы. Потребуем ещё, чтобы все операторы X_j коммутировали. Поскольку $X_jX_k=(-1)^{\omega(f_j,f_k)}X_kX_j$, условие коммутирования означает, что $\omega(f_j,f_k)=0$.

Определение 14.9. *Симплектический квантовый код* задаётся условиями (14.19), где все X_j коммутируют.

Итак, симплектическому квантовому коду соответствует изотропное подпространство $F\subseteq G^n$; изотропность означает, что для любых $f,g\in F$ выполнено $\omega(f,g)=0$. Поэтому размерность симплектического кода легко вычисляется.

Теорема 14.4. $\dim\mathcal{M}=2^{n-\dim F}$.

Лемма 14.6. *Любой симплектический код приводится преобразованиями из* $\mathrm{ESp}_2(n)$ *к стандартному виду, когда проверочными операторами являются* $\sigma^z[1],\ldots,\sigma^z[s]$, *где* $s=\dim F$.

Доказательство. Подпространство $F\subseteq G^n$ можно перевести отображением из $\mathrm{Sp}_2(n)$ в подпространство F', состоящее из векторов вида $(0,\beta_1,0,\beta_2,\ldots,0,\beta_s,0,\ldots,0)$, где β_j — произвольные. Согласно теореме 14.3, этому отображению соответствует некоторое унитарное преобразование $U\cdot U^\dagger$. Оно переводит кодовое подпространство в подпространство, заданное проверочными операторами $\pm\sigma^z[j]$ $(j=1,\ldots,s)$. Применяя дополнительное преобразование вида $\sigma(f)\cdot\sigma(f)^\dagger$, все знаки можно сделать плюсами. □

Теперь посмотрим, какие ошибки способен обнаруживать симплектический код. Напомним, что код обнаруживает ошибки из $\mathcal{E}(n,k)$, если

$$\forall\,|\xi\rangle,|\eta\rangle\in\mathcal{M}\ \forall\,Z\in\mathcal{E}(n,k)\ \langle\xi|Z|\eta\rangle=c(Z)\langle\xi|\eta\rangle \tag{14.20}$$

По соображениям линейности достаточно рассматривать лишь $Z=\sigma(g)$, $|g|\leqslant k$.

Пусть $|\eta\rangle\in\mathcal{M}$, т.е. $X_j|\eta\rangle=|\eta\rangle$ для всех j, где $X_j=(-1)^{\varphi_j}\sigma(f_j)$. Обозначим $Z|\eta\rangle=|\psi\rangle$, где $Z=\sigma(g)$. Как мы сейчас увидим, вектор $|\psi\rangle$ является собственным для всех проверочных операторов X_j, поэтому он либо принадлежит кодовому подпространству $\mathcal{M}$, либо ему ортогонален. Подействуем проверочным оператором на $|\psi\rangle$:

$$\begin{aligned}X_j|\psi\rangle=X_jZ|\eta\rangle=(-1)^{\omega(f_j,g)}ZX_j|\eta\rangle=(-1)^{\omega(f_j,g)}Z|\eta\rangle=\\ =(-1)^{\omega(f_j,g)}|\psi\rangle.\end{aligned}$$

Условие $|\psi\rangle\in\mathcal{M}$ равносильно условию $\omega(f_j,g)=0$ для всех j. Такие g образуют линейное подпространство, которое мы обозначим F_+, т.е. $F_+=\{g\in G^n:\forall\,f\in F\ \omega(f,g)=0\}$.

Возможны следующие три случая:

1. $g\notin F_+$. При этом $|\psi\rangle\perp\mathcal{M}$, поэтому $\langle\xi|\sigma(g)|\eta\rangle=0$. Такую ошибку код обнаружит.

2. $g \in F$. Такая ошибка фактически неотличима от тождественного оператора, так как она не меняет кодового вектора $|\eta\rangle$ (с точностью до фазового множителя). Пусть $g = a_1 f_1 + \ldots + a_s f_s$, тогда $\sigma(g) = c(g) X_1^{a_1} \cdot \ldots \cdot X_s^{a_s}$, где $c(g) = \pm 1, \pm i$. В этом случае $\langle \xi | \sigma(g) | \eta \rangle = c(g) \langle \xi | \eta \rangle$ — условие (14.20) выполнено.
3. $g \in F_+ \setminus F$. В этом случае (проверьте!) $\langle \xi | \sigma(g) | \eta \rangle$ не имеет вида $c(g) \langle \xi | \eta \rangle$. Такую ошибку код не обнаруживает.

Этими рассуждениями доказана следующая теорема.

Теорема 14.5. *Кодовое расстояние для симплектического кода* $\mathcal{M}$

$$d(\mathcal{M}) = \min\{|f| : f \in F_+ \setminus F\}.$$

Заметим отличие от классических линейных кодов. Там кодовое расстояние определяется как наименьшая норма вектора из подпространства с выкинутым нулём. А у симплектических кодов нуль раздувается до подпространства.

Задача 14.4. Постройте симплектический квантовый код типа $(5, 1)$, исправляющий одну ошибку.

Задача 14.5. Докажите, что не существует квантового кода типа $(4, 1)$, исправляющего одну ошибку.

14.9. Торические коды. Приведём важный пример симплектического кода. Он строится так. Пусть есть квадратная решетка размера $r \times r$ на торе. Сопоставим каждому её ребру по q-биту. Таким образом, всего имеется $n = r^2$ q-битов. Проверочные операторы будут двух типов.

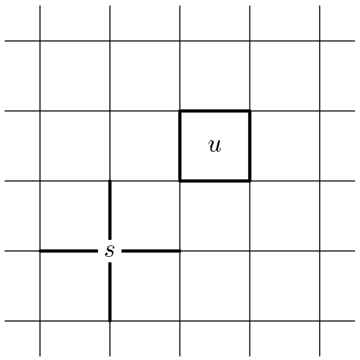


Рис. 6.

Тип I задаётся вершинами. Выберем некоторую вершину s и сопоставим ей проверочный оператор

$$A_s^{(x)} = \sigma(f_s^{(x)}) = \prod_{j \in \text{звезда}(s)} \sigma_j^x.$$

Тип II задаётся гранями. Выберем некоторую грань u и сопоставим ей проверочный оператор

$$A_u^{(z)} = \sigma(f_u^{(z)}) = \prod_{j \in \text{граница}(u)} \sigma_j^z.$$

Операторы $A_s^{(x)}$ и $A_u^{(z)}$ коммутируют, поскольку граница и звезда всегда пересекаются по чётному числу рёбер. (Перестановочность операторов одного типа очевидна.)

Хотя мы указали $r^2 + r^2 = 2r^2$ проверочных операторов (по одному на грань и на вершину), между ними есть соотношения. Произведение всех $A^{(x)}$-операторов, как и произведение всех $A^{(z)}$-операторов, равны тождественному. Можно показать, что других соотношений нет. Поэтому $\dim F = 2r^2 - 2$, а $\dim \mathcal{M} = 2^2$. Поэтому торический код позволяет закодировать два q-бита. Посмотрим, чему равно кодовое расстояние для торического кода.

Для торического кода имеются естественные разложения

$$F = F^{(x)} \oplus F^{(z)}, \qquad F_+ = F_+^{(x)} \oplus F_+^{(z)}$$

на подпространства, соответствующие проверочным операторам, состоящим только из σ_j^x, либо только из σ_j^z. Такие коды называются *CSS кодами* (по фамилиям авторов, впервые рассмотревших этот класс кодов [25, 44]). В случае торического кода элементы подпространств $F^{(z)}$, $F_+^{(z)}$ имеют вид $(0, \beta_1, \ldots, 0, \beta_n)$; им можно сопоставить 1-цепи, т. е. формальные линейные комбинации рёбер с коэффициентами $\beta_1, \ldots, \beta_n \in \mathbb{F}_2$. Элементам подпространств $F^{(x)}$, $F_+^{(x)}$ сопоставляются 1-коцепи. Рассмотрим вектор $f_u^{(z)} \in F^{(z)}$, отвечающий грани с номером u. Ему будет сопоставлена 1-цепь, являющаяся границей этой грани. Легко видеть, что пространство $F^{(z)}$ состоит из всех 1-границ. Аналогично, векторам $f_s^{(x)}$ будут сопоставляться 1-кограницы, порождающие всё пространство 1-кограниц.

Возьмём произвольный элемент $g \in F_+$, $g = g^{(x)} + g^{(z)}$. Условия коммутирования запишутся следующим образом:

$$\omega(f_s^{(x)}, g) = 0 \iff \omega(f_s^{(x)}, g^{(z)}) = 0,$$
$$\omega(f_u^{(z)}, g) = 0 \iff \omega(f_u^{(z)}, g^{(x)}) = 0.$$

Чтобы выполнялось $\omega(f_s^{(x)}, g^{(z)}) = 0$, нужно, чтобы в любой звезде было чётное число рёбер с ненулевыми весами из $g^{(z)}$. Другими словами, $g^{(z)}$ — это 1-цикл (с коэффициентами в $\mathbb{Z}_2$). Аналогично, $g^{(x)}$ должен быть 1-коциклом.

Итак, пространства $F_+^{(z)}$, $F_+^{(x)}$ состоят из 1-циклов и 1-коциклов, а пространства $F^{(z)}$, $F^{(x)}$ состоят из 1-границ и 1-кограниц. Следовательно, кодовое расстояние есть минимальная мощность (количество ненулевых коэффициентов) по циклам, не являющимся границами, и коциклам, не являющимся кограницами. Легко видеть, что этот минимум равен r (нужны либо цикл, либо разрез, не гомологичные 0). Это означает, что торический код исправляет $\lfloor (r-1)/2 \rfloor$ ошибок.

Будем обозначать коды описанного вида через $\mathrm{TOR}(r)$.

Замечание. Торические коды являются очень важным примером, обладающим рядом замечательных свойств. В частности, это *коды с локальными проверками*. Последовательность кодов называется кодами с локальными проверками, если выполнены следующие условия:

- каждый проверочный оператор действует на ограниченное константой число q-битов;
- каждый q-бит входит в ограниченное константой число проверочных операторов;
- кодовое расстояние неограниченно возрастает.

Такие коды представляют интерес для задачи построения вычислительных схем, устойчивых к ошибкам. При исправлении ошибок могут происходить новые ошибки. Но для кодов с локальными проверками схемы исправления ошибок имеют фиксированную глубину, поэтому одна ошибка при работе такой схемы портит ограниченное число q-битов.

14.10. Процедура исправления ошибок. Определение 14.6 и теорема 14.2 указывают только на принципиальную возможность восстановить исходное состояние системы после действия ошибки. На примере симплектических кодов покажем, как реализовать процедуру исправления ошибки.

Рассмотрим частный случай, к которому все сводится. Пусть имеются две ошибки, заданные операторами $X = \sigma(g_1)$, $Y = \sigma(g_2)$. Тогда $Z = Y^\dagger X = c\sigma(g_1 - g_2)$ ($|c| = 1$). Назовём *синдромом* ошибки g_1 вектор $(\omega(g_1, f_1), \ldots, \omega(g_1, f_s))$ (для g_2 — аналогично).

Возьмём вектор $|\eta\rangle \in \mathcal{M}$. Обозначим $|\psi\rangle = X|\eta\rangle$. Проверочные операторы действуют на $|\psi\rangle$ так: $X_j|\psi\rangle = (-1)^{\omega(g_1, f_j)}|\psi\rangle$. Поэтому, измеряя собственные числа X_j на состоянии $|\psi\rangle$, можно измерить синдром.

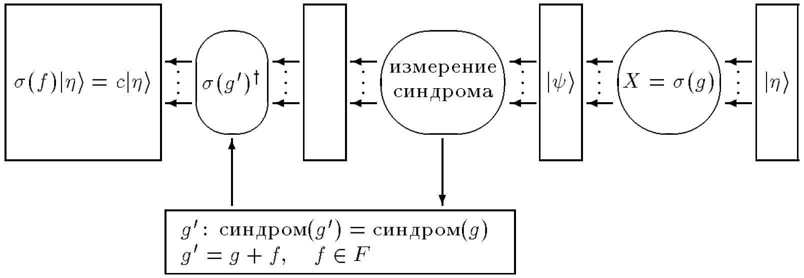


Рис. 7. *Исправление ошибки*

Если кодовое расстояние равно k, то выполнено

$$\forall\, g_1, g_2 \Big((|g_1| \leqslant k, |g_2| \leqslant k) \Rightarrow \big((g_1 - g_2 \in F) \vee (g_1 - g_2 \notin F_+) \big) \Big).$$

Условие $g_1 - g_2 \in F$ означает эквивалентность ошибок g_1 и g_2, т. е. $\sigma(g_1)|\eta\rangle = c\sigma(g_2)|\eta\rangle$ для любого вектора $|\eta\rangle$ из кодового подпространства. Условие $g_1 - g_2 \notin F_+$ равносильно тому, что синдромы ошибок g_1 и g_2 не совпадают. Итак, либо ошибки эквивалентны, либо их можно различить по синдрому. Следовательно, по синдрому можно определить ошибку с точностью до эквивалентности, т. е. по модулю подпространства F.

Теперь ясно, как нужно исправлять ошибку. После того как определён синдром, применим оператор, обратный к оператору восстановленной по синдрому ошибки. Получим состояние, отличающееся от исходного лишь на фазовый множитель. Вся процедура изображена на рис. 7.

Выше рассмотрен случай ошибки типа $\sigma(g)$. На самом деле ошибка состоит в действии преобразования матриц плотности вида

$$T = \sum_{|h| \leqslant k, |h'| \leqslant k} b_{h,h'} \sigma(h) \cdot \sigma(h')^\dagger.$$

В качестве упражнения читателю предлагается проверить, как работает приведённая выше схема в случае такого общего преобразования матриц плотности.

Задача 14.6. Постройте полиномиальный алгоритм определения ошибки по синдрому для торического кода.

14.11. Анионы (иллюстративный пример на основе торического кода). На примере торического кода можно дать более точное представление об анионных системах, о которых говорилось во введении.

Итак, вновь рассмотрим квадратную сетку на торе (а можно и на плоскости — сейчас нас будет интересовать только её центральная часть). Как и раньше, для каждой вершины s и каждой грани u рассмотрим проверочные операторы

$$A_s^{(x)} = \prod_{j\in\,\text{звезда}(s)} \sigma_j^x, \quad A_u^{(z)} = \prod_{j\in\,\text{граница}(u)} \sigma_j^z.$$

Состояние кодового подпространства задается условиями $A_s^{(x)}|\xi\rangle = |\xi\rangle$, $A_u^{(z)}|\xi\rangle = |\xi\rangle$. Их можно переписать другим способом. Рассмотрим следующий *гамильтониан* — эрмитов оператор

$$H = \sum_s (I - A_s^{(x)}) + \sum_u (I - A_u^{(z)}).$$

Этот оператор неотрицательный, причем его нулевое подпространство в точности совпадает с кодовым подпространством торического кода. Таким образом, векторы из кодового подпространства являются собственными и обладают наименьшей энергией (т. е. собственным числом гамильтониана). Такие векторы называются *основными состояниями*, а векторы из ортогонального дополнения — *возбуждёнными состояниями*.

Рассмотрим возбуждённые состояния с наименьшей ненулевой энергией, когда нарушено ровно два условия (например, вершинных). (Число нарушенных условий каждого типа чётное, поскольку $\prod_s A_s^{(x)} = \prod_u A_u^{(z)} = I$.) Тогда для двух вершин, в которых кодовые условия нарушаются, выполнено

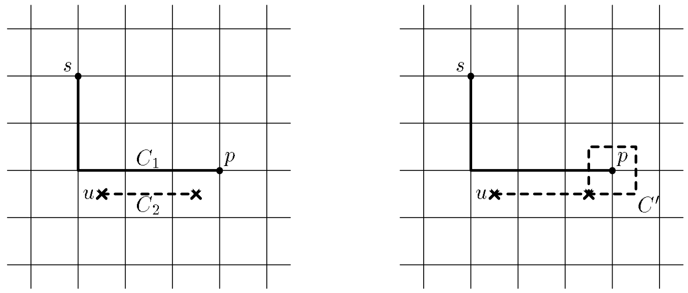
$$A_s^{(x)}|\eta\rangle = -|\eta\rangle, \quad A_p^{(x)}|\eta\rangle = -|\eta\rangle.$$

Как можно получить состояние $|\eta\rangle$ из кодового состояния $|\xi\rangle$? Соединим p и s решёточным путем C_1 и подействуем на $|\xi\rangle$ оператором $W = \prod_{j\in C_1} \sigma_j^z$. Этот оператор коммутирует с проверочными вершинными операторами для всех промежуточных вершин пути C_1, а в концах — антикоммутирует: $WA_s^{(x)} = -A_s^{(x)}W$. Положим $|\eta\rangle = W|\xi\rangle$ и покажем, что $|\eta\rangle$ удовлетворяет требуемым свойствам. Для вершины s (аналогично и для p) имеем

$$|\eta\rangle = W|\xi\rangle = WA_s^{(x)}|\xi\rangle = -A_s^{(x)}W|\xi\rangle = -A_s^{(x)}|\eta\rangle$$

($A_s^{(x)}|\xi\rangle = |\xi\rangle$, так как состояние $|\xi\rangle$ — кодовое).

Любое состояние системы можно построить из элементарных возбуждений двух типов, одни из которых «живут» на вершинах, другие — на гранях. Элементарное возбуждение — это просто нарушенное кодовое условие, но теперь мы думаем о нём как о частице. Частицы-возбуждения можно двигать, создавать и уничтожать. Пара возбуждений первого типа получается из основного (кодового) состояния дей-



а) б)

Рис. 8.

ствием оператора W, приведённого выше; пара возбуждений второго типа — действием оператора $V = \prod_{j\in C_2} \sigma_j^x$, где C_2 — путь, соединяющий две грани, как показано на рис. 8а). Как и раньше проверяется, что $A_u^{(z)}(V|\xi\rangle) = -V|\xi\rangle$.

Что случится, если двигать возбуждение одного типа (крестик) вокруг возбуждения второго типа (кружочка)? (См. рис. 8б).) Движение возбуждения описывается оператором $\prod_{j\in C'} \sigma_j^x = \prod_r A_r^{(x)}$, зависящим от контура обхода C' (здесь r пробегает все грани внутри C'). Очевидно, что $A_r^{(x)}|\psi\rangle = |\psi\rangle$ для всех $r \neq p$. В результате мы получим

$$|\psi\rangle \mapsto \prod_{j\in C'} \sigma_j^x|\psi\rangle = A_p^{(x)}|\psi\rangle = -|\psi\rangle.$$

То есть вектор состояния домножился на -1. Это и означает, что рассматриваемые возбуждения являются (абелевыми) анионами.

На торе можно двигать частицы по двум различным циклам, образующим базис в группе гомологий. Например, можно создать из основного состояния пару возбуждений одного типа, обнести одно из возбуждений по циклу и проаннигилировать со вторым возбуждением. Этот процесс описывается некоторым оператором, действующим на кодовом подпространстве, — произведением σ_j^z вдоль пути на решетке, либо σ_j^x вдоль пути на двойственной решётке. Поскольку существует два типа возбуждений, мы имеем 4 таких оператора: $Y_1^{(z)}$ и $Y_2^{(x)}$ соответствуют одному базисному циклу, а $Y_2^{(z)}$ и $Y_1^{(x)}$ — другому. Эти операторы действуют на два закодированных q-бита как σ_i^z, σ_i^x $(i = 1, 2)$, потому что они обладают такими же коммутационными соотношениями: $Y_i^{(x)}Y_i^{(z)} = -Y_i^{(z)}Y_i^{(x)}$ (остальные пары коммутируют).

Часть III

Решения задач

В этой части приводятся решения задач или неформальные указания, пользуясь которыми заинтересованный читатель может восстановить строгое решение самостоятельно.

Из раздела 1

1.1. Неформально описать такую машину легко. Она переносит символы по одному слева направо и справа налево, пока не обнаруживает, что достигнута середина слова, после чего останавливается.

Дадим теперь формальное описание этой машины.

Внешний алфавит: $\mathcal{A} = \{0, 1\}$. Алфавит $\mathcal{S} = \{␣, 0, 1, *, 0', 1'\}$, помимо символов внешнего алфавита и пустого символа ␣, включает три вспомогательных метки, указывающих на положение символов, которые переносятся.

Множество состояний управляющего устройства

$$\mathcal{Q} = \{q_0, q_f, r_0, r_1, l_0, l_1, l_{0'}, l_{1'}\}.$$

Буквы r и l указывают на направление переноса символа, индексы при этих буквах — на переносимый символ.

Теперь зададим функцию переходов. Для всех пар, на которых функция переходов не определена приводимыми ниже правилами, её значения можно выбирать произвольно.

Начало работы:

$$(q_0, 0) \mapsto (r_0, *, +1), \qquad (q_0, 1) \mapsto (r_1, *, +1),$$
$$(q_0, ␣) \mapsto (q_0, ␣, -1).$$

Первая строка означает, что машина поставила метку в первой позиции и понесла вправо символ, который в ней стоял. Вторая строка означает, что на пустом слове машина сразу останавливается.

Перенос вправо:

$$(r_0, 0) \mapsto (r_0, 0, +1), \qquad (r_1, 0) \mapsto (r_1, 0, +1),$$
$$(r_0, 1) \mapsto (r_0, 1, +1), \qquad (r_1, 1) \mapsto (r_1, 1, +1).$$

Машина двигается вправо, пока не достигнет конца слова или метки.

Перемена направления движения справа налево состоит из двух действий: снять метку (если это не пустой символ)

$$(r_0, 0') \mapsto (l_{0'}, 0, -1), \qquad (r_1, 0') \mapsto (l_{1'}, 0, -1),$$
$$(r_0, 1') \mapsto (l_{0'}, 1, -1), \qquad (r_1, 1') \mapsto (l_{1'}, 1, -1),$$
$$(r_0, ␣) \mapsto (l_{0'}, ␣, -1), \qquad (r_1, ␣) \mapsto (l_{1'}, ␣, -1)$$

и поставить её на левого соседа

$$(l_{0'}, 0) \mapsto (l_0, 0', -1), \qquad (l_{1'}, 0) \mapsto (l_0, 1', -1),$$
$$(l_{0'}, 1) \mapsto (l_1, 0', -1), \qquad (l_{1'}, 1) \mapsto (l_1, 1', -1).$$

Перенос влево:

$$(l_0, 0) \mapsto (l_0, 0, -1), \qquad (l_1, 0) \mapsto (l_0, 0, -1),$$
$$(l_0, 1) \mapsto (l_1, 1, -1), \qquad (l_1, 1) \mapsto (l_1, 1, -1).$$

Перемена движения слева направо:

$$(l_0, *) \mapsto (q_0, 0, +1), \qquad (l_1, *) \mapsto (q_0, 1, +1).$$

Завершение работы зависит от чётности длины слова: при чётной длине остановка происходит при начале движения вправо

$$(q_0, 0') \mapsto (q_f, 0, -1), \qquad (q_0, 1') \mapsto (q_f, 1, -1),$$

а при нечётной длине — при начале движения влево

$$(l_{0'}, *) \mapsto (q_f, 0, -1), \qquad (l_{1'}, *) \mapsto (q_f, 1, -1),$$
$$(q_f, 0) \mapsto (q_f, 0, -1), \qquad (q_f, 1) \mapsto (q_f, 1, -1).$$

1.2. Неформально делается следующее: ко второму слагаемому поочерёдно добавляются разряды первого, добавленный разряд стирается. Добавление одного разряда происходит за время, не превышающее удвоенной длины второго слагаемого, так что общее время работы машины квадратично зависит от длины входа.

Дадим теперь формальное описание такой машины.

Внешний алфавит: $\mathcal{A} = \{0, 1, +\}$. Алфавит $\mathcal{S} = \{␣, 0, 1, +, 0', 1', +'\}$, помимо символов внешнего алфавита и пустого символа ␣, включает три вспомогательных метки.

Множество состояний управляющего устройства

$$\mathcal{Q} = \{q_0, q_f, q_p, r_0, r_1, l_0, l_1, d\}.$$

Буквы r и l указывают на направление переноса символа, индексы при этих буквах — на переносимый символ (влево машина носит бит переноса в следующий разряд, а вправо — добавляемый бит), в состоянии d машина осуществляет стирание обработанного разряда.

Теперь зададим функцию переходов. Как и в задаче 1.1, для не упомянутых ниже пар функция переходов определена произвольно.

Начало и конец работы:

$$\begin{array}{lll}
(q_0,0)\mapsto(q_0,0',+1), & (q_0,1)\mapsto(q_p,1',+1), & (q_0,+)\mapsto(d,0,-1),\\
(q_p,0)\mapsto(q_p,0,+1), & (q_p,1)\mapsto(q_p,1,+1), & (q_p,+)\mapsto(d,0,-1),\\
(q_p,0')\mapsto(q_f,0,+1), & (q_p,1')\mapsto(q_f,1,+1), & \\
(q_f,0)\mapsto(q_f,0,-1), & (q_f,1)\mapsto(q_f,1,-1). &
\end{array}$$

Самый левый символ первого слагаемого помечается, чтобы не пропустить конец работы. Далее машина движется вправо в состоянии q_p. Если найден знак $+$, то происходит переход к началу добавления очередного слагаемого. Если найдена метка, то она стирается, а машина останавливается. При этом на ленте остаётся результат сложения (считаем, что сумма может начинаться нулями).

Определение очередного бита, который нужно добавлять ко второму слагаемому, перенос его вправо и переход в режим сложения:

$$\begin{array}{ll}
(d,0)\mapsto(r_0,+,+1), & (d,1)\mapsto(r_1,+,+1),\\
(d,0')\mapsto(r_0,+',+1), & (d,1')\mapsto(r_1,+',+1),\\
(r_0,0)\mapsto(r_0,0,+1), & (r_1,0)\mapsto(r_1,0,+1),\\
(r_0,1)\mapsto(r_0,1,+1), & (r_1,1)\mapsto(r_1,1,+1),\\
(r_0,\sqcup)\mapsto(l_{0'},\sqcup,-1), & (r_1,\sqcup)\mapsto(l_{1'},\sqcup,-1),\\
(r_0,0')\mapsto(l_{0'},0,-1), & (r_1,0')\mapsto(l_{1'},0,-1),\\
(r_0,1')\mapsto(l_{0'},1,-1), & (r_1,1')\mapsto(l_{1'},1,-1),\\
(l_{0'},0)\mapsto(l_0,0',-1), & (l_{1'},0)\mapsto(l_0,1',-1),\\
(l_{0'},1)\mapsto(l_0,1',-1), & (l_{1'},1)\mapsto(l_1,0',-1).
\end{array}$$

Сложение, пока не достигнут знак $+$

$$\begin{array}{ll}
(l_0,0)\mapsto(l_0,0,-1), & (l_1,0)\mapsto(l_0,1,-1),\\
(l_0,1)\mapsto(l_0,1,-1), & (l_1,1)\mapsto(l_1,0,-1),\\
(l_0,+)\mapsto(d,0,-1), & (l_1,+)\mapsto(d,1,-1),\\
(l_0,+')\mapsto(q_p,0,-1), & (l_1,+')\mapsto(q_p,1,-1).
\end{array}$$

Последняя строчка применяется в конце, когда левее знака $+'$ уже ничего нет. Поэтому машина начинает двигаться вправо, чтобы стереть оставшуюся метку.

1.3. Доказательство от противного. Предположим, что такой алгоритм есть, т. е. существует машина A, которая на входе $([M],x)$ даёт ответ «да», если машина M останавливается на входе x, в противном случае даёт ответ «нет» (через $[M]$ обозначено описание машины M). Тогда есть и такая машина A', которая на входе X моделирует работу A на входе (X,X). Затем, если ответ машины A — «да», то A' начинает двигать головку вправо и не останавливается, а если ответ A — «нет», то A' останавливается.

Остановится ли A' на входе $[A']$? Если остановится, то A даёт ответ «да» на входе $([A'],[A'])$. Тогда, по определению машины A', на входе $[A']$ она не остановится. Итак, A' на входе $[A']$ не останавливается. Но тогда A даёт ответ «нет» на входе $([A'],[A'])$. Но это означает, что A' на входе $[A']$ останавливается. Пришли к противоречию.

1.4. Во-первых, заметим, что есть алгоритм, который выписывает одну за другой те МТ, которые *останавливаются*, будучи запущенными на пустой ленте. Этот алгоритм перебирает все пары $([M],n)$ ($[M]$ — описание машины M, n — натуральное число) и для каждой пары моделирует работу M на пустом входе в течение n тактов. Если за это время происходит остановка, то M включается в список, если не была включена в него ранее.

Если бы существовал ещё и такой алгоритм, который выписывает одну за другой машины, не останавливающиеся на пустом входе, то можно было бы построить и алгоритм, проверяющий, останавливается ли МТ A на пустом входе: запускаем оба алгоритма перечисления и ждём, когда описание A появится в одном или в другом списке.

Но тогда существовал бы и алгоритм, решающий проблему остановки: по машине M и входу x легко строится машина, которая сначала записывает x на ленту, а затем моделирует работу M. Так что из предыдущей задачи заключаем, что нет алгоритма, перечисляющего машины, не останавливающиеся на пустом слове.

1.5. Ограничимся указанием. Для любой вычислимой функции $b(n)$ при достаточно больших n среди машин с n символами алфавита и n состояниями, останавливающихся на пустом входе, можно найти и такую, которая вычисляет $nb(n)$, а затем вычитает из этого числа по единице, пока не дойдёт до нуля.

1.6. Пусть имеется двухленточная МТ M_2, работающая за время $T(n)\geqslant n$ на входах длины n. Опишем неформально машину M_1 с единственной лентой, моделирующую работу M_2. Алфавит этой машины достаточно велик, чтобы кодировать в одной ячейке содержимое ячеек с тем же номером на лентах M_2 и информацию о положении головок (на каких лентах головки находятся над ячейкой с тем же номером). Управляющее устройство M_1 позволяет запоминать содержимое ячеек, над которыми находятся головки M_2, и состояние управляющего устройства M_2.

Машина M_1 работает циклами, каждый из которых имитирует один такт работы M_2. В начале каждого цикла головка M_1 находится над самой левой ячейкой.

Цикл состоит из двух последовательных проходов по записанному слову. Вначале M_1 движется вправо и собирает информацию о состояниях в ячейках M_2, над которыми находятся головки. При обратном проходе справа налево M_1 выполняет действия, имитирующие такт работы M_2. На каждое такое действие требуется $O(1)$ тактов.

Цикл выполняется за $O(S)$ тактов работы M_1, где S — длина используемой части ленты. Так как $T(n) \geqslant n$, то $S \leqslant \max\{n, T(n)\} = T(n)$. Поэтому M_1 работает за время $O(ST(n)) = O(T^2(n))$.

1.7. Приведём ещё более неформальное, чем в предыдущей задаче, описание алгоритма. Алфавит моделирующей машины выберем достаточно большим, чтобы кодировать в одной ячейке содержимое ячеек на всех лентах, информацию о положении головок (на каких лентах головки находятся над данной ячейкой) и дополнительные метки, которые потребуются в процессе работы.

Опишем алгоритм A, который решает такую задачу: на одной из лент записано слово (t, w), на второй головка находится в конце используемой части ленты, нужно промоделировать работу трёхленточной машины за период времени t (записанный двоичным словом), если вначале состояние лент определено словом w.

Запишем на свободное место на второй ленте число $t/2$, после чего скопируем туда же $(t/2)$-окрестности каждой из головок на всех трёх лентах исходной машины и переместим головку в конец слова w (в конец используемой части первой ленты). К полученному на второй ленте слову применим рекурсивно алгоритм A, по завершении его работы скопируем на первую ленту результат моделирования, скопируем на вторую ленту $(t/2)$-окрестности головок во вновь полученном состоянии, отвечающем состоянию трёхленточной машины после $t/2$ тактов работы, опять переместим головку в конец используемой части первой ленты. Ещё раз рекурсивно применим A к слову на второй ленте, по завершении его работы скопируем результат моделирования на первую ленту.

Для корректного описания алгоритма нужно ещё задать его работу на слове $(1, w)$. В этом случае просто применяем алгоритм, аналогичный описанному в предыдущей задаче.

Операцию копирования с ленты на ленту можно осуществить за линейное от длины копируемого слова время. Поэтому для времени $\widetilde{T}(t, s)$ работы в наихудшем случае алгоритма A, моделирующего работу трёхленточной машины за время t на словах длины s, получаем оценку:

$$\widetilde{T}(t, s) \leqslant 2\widetilde{T}(\frac{t}{2}, t) + O(t) + O(s). \qquad (*)$$

Из $(*)$ сразу следует, что при некоторой константе C_1 и $t > 1$

$$\widetilde{T}(t, 2t) \leqslant C_1 t(\log t + 1).$$

Поэтому

$$\widetilde{T}(t, s) = O(t \log t) + O(t) + O(s).$$

Заметим, что получить слово (t, w) из слова w можно за время $|w| \log t$, если t известно.

Алгоритм моделирования трёхленточной машины на двухленточной использует алгоритм A следующим образом. Промоделируем работу машины из начального состояния за 1 такт, затем работу за 2 такта из достигнутого состояния и т. д. Оценим время работы этого алгоритма. Пусть исходная трёхленточная машина работает на словах длины n за время $T(n) \geqslant n$. Тогда время $T'(n)$ работы моделирующей машины будет оцениваться как

$$T'(n) \leqslant \sum_{k=0}^{\lceil \log T(n) \rceil} \widetilde{T}(2^k, n + 2^k) \leqslant \sum_{k=0}^{\lceil \log T(n) \rceil} O(n + 2^k) + O(k2^k) + O(2^k) \leqslant$$
$$\leqslant \sum_{k=0}^{\lceil \log T(n) \rceil} O(T(n)) = O(T(n) \log T(n)).$$

1.8. Информация в машине Тьюринга переносится головкой управляющего устройства. Более точно эта мысль формулируется так. Рассмотрим последовательность $\{Q_j(x, k)\}$ состояний управляющего устройства МТ в моменты переходов головки между k-й и $(k+1)$-й ячейками при работе МТ на входе x (рассматриваем переходы в обе стороны). Работа машины Тьюринга справа от k-й ячейки полностью определяется последовательностью $\{Q_j(x, k)\}$.

Теперь запишем нижнюю оценку на время работы МТ, копирующей входное слово. Она основана на том, что каждый переход головки требует отдельного такта работы МТ. Введём параметры ε и τ_k, значения которых определим позже. Для слова w через $\tau_k(w)$ обозначим число переходов головки между k-й и $(k+1)$-й ячейками при работе МТ на входе w. Будем искать такое слово w, что $\tau_k(w) > \tau_k$ при $k \geqslant n/2$.

Поскольку МТ копирует начальный кусок длины k справа от k-й ячейки, последовательности $\{Q_j(vu,k)\}$ при фиксированном u должны быть различны для различных k-буквенных слов v. Коротких (длины не больше τ) последовательностей состояний управляющего устройства не больше, чем $|\mathcal{Q}|^{\tau+1}/(|\mathcal{Q}|-1)$, где $\mathcal{Q}$ — множество состояний управляющего устройства. Для любого $(n-k)$-буквенного слова u при

$$\frac{|\mathcal{Q}|^{\tau+1}}{|\mathcal{Q}|-1} \leqslant 2|\mathcal{Q}|^{\tau} \leqslant \varepsilon|\mathcal{A}|^k, \qquad (*)$$

где $\mathcal{A}$ — внешний алфавит, неравенство $\tau_k(vu) > \tau$ выполнено по крайней мере для доли $1-\varepsilon$ от всех k-буквенных слов v. Это означает, что для доли $1-|\mathcal{A}|^{n-k}\varepsilon$ от всех k-буквенных слов v неравенство $\tau_k(vu) > \tau$ выполняется для всех u. Поэтому при

$$|\mathcal{A}|^{n-k}\varepsilon < \frac{1}{n} \qquad (**)$$

найдётся слово w такое, что $\tau_k(w) > \tau_k$ при $k \geqslant n/2$.

Если $\varepsilon = (n|\mathcal{A}|^{n/2})^{-1}$, то $(**)$ выполняется при $k \geqslant n/2$. Если при этом ещё и $\tau_k = \lfloor (\log(\varepsilon/2) + \frac{2}{3}n\log|\mathcal{A}|)/\log|\mathcal{Q}| \rfloor$, то $(*)$ выполняется при $k \geqslant 2n/3$. При таком выборе параметров $\tau_k = \Omega(n)$ при $k \geqslant 2n/3$. Оценим время работы МТ на слове w таком, что $\tau_k(w) > \tau_k$ при $k \geqslant 2n/3$ (мы уже доказали, что такое слово есть)

$$T(n) \geqslant \sum_{k=\lceil 2n/3 \rceil}^{n} \tau_k = \frac{n}{3}\cdot\Omega(n) = \Omega(n^2).$$

Для оценки минимального времени работы $T'(n)$ можно считать, что МТ вначале дописывает за $T_1(n)$ тактов последовательность из одних 0 длины n, затем за $O(n)$ шагов проверяет, состоит ли исходное слово из одних 0, после чего прекращает работу, если это так, а в противном случае работает любым правильным способом. Ясно, что $T'(n) = T_1(n) + O(n)$. В свою очередь, $T_1(n) = O(n\log n)$. Действительно, если бы машина во внутренней памяти могла хранить числа, то копирование слова из нулей не создало бы проблемы (надо было бы подсчитать длину слова и потом написать столько же нулей). Но этого сделать нельзя. Зато машина Тьюринга может хранить число в двоичной записи в окрестности своей головки (мы можем расширить алфавит и считать, что на ленте есть место для дополнительных пометок рядом с буквами слова из нулей и единиц). При этом прибавление единицы к такому счётчику, вычитание единицы и сдвиг счётчика по ленте (его ведь надо возить с собой) требуют времени порядка длины счётчика, т. е. $O(\log n)$, так что всего мы укладываемся в $O(n\log n)$ тактов.

Замечание. Можно показать, что $T'(n) = \Omega(n\log n)$. Читателю предлагается самостоятельно понять, как нужно модифицировать изложенную выше нижнюю оценку времени работы в худшем случае.

1.9. Приведём идею написания такой программы.

Будем писать программу, моделирующую работу универсальной машины Тьюринга. Поскольку значение переменной может быть сколь угодно велико, то в одной переменной можно хранить всю ленту машины (цифры $|\mathcal{S}|$-ичного представления числа, где $\mathcal{S}$ — алфавит машины). Указатель положения головки — это ещё одна переменная, состояние управляющего устройства — третья.

Преобразования этих переменных за такт работы описываются простыми арифметическими действиями (сложение, умножение, возведение в степень, деление с остатком и нахождение этого остатка, сравнение чисел). Все эти действия легко реализовать без рекурсии, используя их стандартные определения и привлекая небольшое количество дополнительных переменных.

Поскольку состояний управляющего устройства и символов алфавита фиксированное число, набором вложенных проверок `if-then-else` конечной глубины можно задать выбор текущих значений этих переменных, и, тем самым, функцию переходов.

Замечание. Когда значения переменных не ограничены, в одной переменной можно хранить целый массив таких переменных:

$$(x_1,\ldots,x_n) \mapsto 2^{x_1}3^{x_2}\ldots p_n^{x_n},$$

p_j — простые числа. Поэтому ограничение на число переменных несущественно.

1.10. Будем искать все функции от двух переменных, которые выражаются формулами в базисе $\mathcal{F}$. Вначале есть две такие функции (проекции): $p_1(x,y) = x$ и $p_2(x,y) = y$. Далее к множеству уже построенных функций $\mathcal{F}'$ применяется следующая процедура. Добавляем к множеству $\mathcal{F}'$ все функции вида $f(g_1(x_1,x_2), g_2(x_3,x_4),\ldots,g_k(x_{2k-1},x_{2k}))$, где $x_j \in \{x,y\}$, $g_j \in \mathcal{F}'$, $f \in \mathcal{F}$. Если множество $\mathcal{F}'$ увеличилось, повторяем процедуру ещё раз. В противном случае возможны два варианта: либо уже получены все функции от двух переменных (тогда базис полон), либо — нет (тогда базис не является полным).

Оценим время работы этого алгоритма. Расширять множество $\mathcal{F}'$ можно лишь 14 раз (всего есть 16 булевых функций от двух переменных). Каждый раз нужно проверить не более $|\mathcal{F}'|^m\cdot 2^m = (2^m)^{1+\log|\mathcal{F}'|}$ вариантов, где m — максимальное количество аргументов у базисных функций. Длина входа (длина записи базиса) не превосходит $|\mathcal{F}|2^m$, а $|\mathcal{F}'| = O(1)$. Так что алгоритм работает за время, ограниченное полиномом от длины входа.

1.11. Верхняя оценка $n2^n < 2{,}01^n$ (при $n \geqslant 2000$) для c_n сразу следует из представления функции в дизъюнктивной нормальной форме (см. (1.1) на с. 24).

Для получения нижней оценки подсчитаем число различных схем размера s и сравним его с количеством функций от n переменных. Для определённости считаем, что используется стандартный полный базис. Тогда для k-го присваивания схемы есть не более

$$2 \cdot (n+k)(n+k-1)/2 + (n+k-1) < (n+k)^2$$

возможностей (конъюнкция и дизъюнкция — симметрические функции, а выбирать мы можем из $n+k-1$ переменной), поэтому количество N_s различных схем размера s не больше, чем

$$\prod_{k=1}^{s}(n+k)^2 \leqslant \prod_{k=1}^{s} 2n^2k^2 \leqslant (2n^2)^s(s!)^2 \leqslant 2^{s(1+2\log n)+2s\log s}.$$

А число булевых функций от n переменных равно 2^{2^n}. При

$$2^n > s(1+2\log n) + 2s\log s \qquad (*)$$

функций больше, чем схем, поэтому $c_n > s$. Если $s = 1{,}99^n$, то неравенство $(*)$ выполняется при $n \geqslant 3000$.

1.12. Снова используем дизъюнктивную нормальную форму. Если строить по формуле (1.1) схему из элементов AND и OR, имеющих произвольное число входов, то понадобится не более 3 слоёв элементов: один слой на отрицания, один — на конъюнкции, и один — на итоговую дизъюнкцию.

1.13. Как говорилось на с. 23, схему можно представлять в виде графа. Из каждой невыходной вершины графа схемы есть хотя бы один ориентированный путь в одну из выходных вершин. Поэтому размер схемы ограничен сверху суммой числа выходных вершин и числа таких путей.

Длина ориентированного пути в графе схемы не превосходит глубины схемы d. Для схемы в стандартном базисе входная степень каждой вершины не больше 2. Поэтому ориентированных путей, ведущих в выходные вершины, не больше чем $2^d = O(n^{O(1)})$ (двигаемся от выходной вершины против направления стрелок и применяем индукцию).

1.14. Результатов сравнения двух чисел x и y три — $x > y$, или $x = y$, или $x < y$. Будем строить схему, которая выдаёт два бита результата, кодирующие эти три возможности.

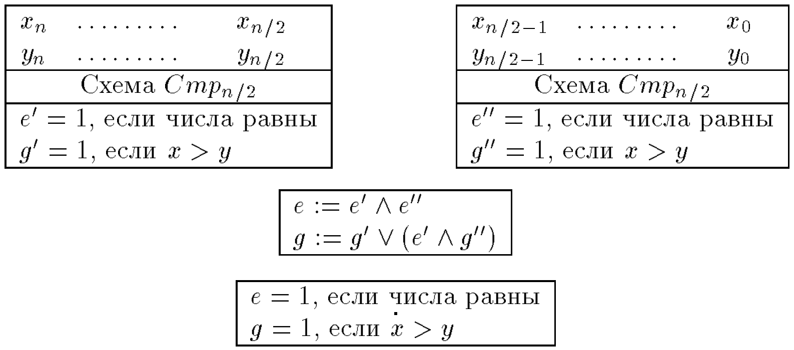


Рис. 9. *Схема Cmp_n для сравнения n-разрядных чисел (размер схемы $O(n)$, глубина — $O(\log n)$)*

Для простоты полагаем, что n является степенью двойки. Это не портит оценку в общем случае, потому что можно дополнить числа нулями слева, чтобы их длина стала степенью двойки. Размер входа увеличивается при этом не более чем вдвое.

Схему сравнения n-разрядных чисел будем собирать из двух схем, сравнивающих числа, образованные первыми $n/2$ разрядами и последними $n/2$ разрядами. Зная результаты сравнения этих двух пар чисел, можно восстановить и результат сравнения исходных чисел.

Конструкция схемы изображена на рис. 9. Оценим её размер и глубину. Выполняются следующие рекуррентные соотношения

$$s_n = 2s_{n/2} + 3, \qquad h_n = h_{n/2} + 2.$$

Из них получаем $s_n = O(n)$ и $h_n = O(\log n)$.

Замечание. Сравнение n-значных двоичных чисел x и y равносильно определению старшего разряда числа $x + (2^n - 1 - y)$, а число $2^n - 1 - y$ находится по y схемой глубины $O(1)$ линейного размера (нужно применить отрицание ко всем переменным). Поэтому достаточно было бы решить следующую задачу 1.15. Мы, наоборот, будем использовать сравнение чисел для сложения.

1.15. Введём обозначения: пусть $x_{n-1}, \ldots, x_0$ — двоичные разряды первого слагаемого; $y_{n-1}, \ldots, y_0$ — второго; $s_n, s_{n-1}, \ldots, s_0$ — разряды результата; $r_{n-1}, \ldots, r_0$ — биты переноса в следующий разряд. Введём дополнительные переменные $t_i = x_i \oplus y_i$, $t_0 = 0$. Тогда $s_0 = t_0$, $s_i = r_{i-1} \oplus t_i$ при $i > 0$; $r_{-1} = 0$, $r_i = (r_{i-1} \wedge t_i) \vee x_i$ при $i \geqslant 0$ (если биты в слагаемых различны, то бит переноса такой же, как в предыдущем разряде; если одинаковы — бит переноса совпадает с их (общим) значением).

Пока мы вводили обозначения, мимоходом решилась задача пункта а). Действительно, присваиваний по приведённым выше формулам нужно сделать $O(n)$ штук.

Для пункта б) используем предыдущую задачу.

Заметим, что если есть схема размера S и глубины H, вычисляющая биты переноса, то из неё легко строится схема размера $S + O(n)$ и глубины $H + O(1)$, вычисляющая сумму (все t_i могут быть найдены параллельно, и, при известных битах переноса r_i, все s_i также могут быть найдены параллельно).

Вычисление битов переноса равносильно сравнению, так что достаточно научиться сравнивать параллельно все «суффиксы» чисел, т.е. для каждого i сравнить числа $x_i x_{i-1} \dots x_0$ и $\neg y_i \neg y_{i-1} \dots \neg y_0$.

Вначале сравним числа $x_{n-1}x_{n-2}\dots x_0$ и $\neg y_{n-1}\neg y_{n-2}\dots\neg y_0$ по схеме, описанной в предыдущей задаче. Заметим, что при работе этой схемы на нижнем уровне мы сравниваем биты, на следующем — двузначные числа, затем — четырёхзначные и т. д. Получаем «сужающееся дерево». Оно даёт результаты сравнения блоков по 2^k битов, в частности, для суффиксов длин $1, 2, \dots, 2^m = n$. Это числа, двоичная запись которых содержит ровно одну единицу. Комбинируя результаты сравнения суффиксов длины 2^k и соседних с ними блоков, получим результаты сравнения суффиксов, двоичная запись длин которых содержит две единицы. Продолжая этот процесс, мы получим результаты сравнения всех суффиксов. Поскольку количество единиц в двоичной записи длины суффикса не превосходит $\log n$, глубина полученной схемы $O(\log n)$. Размер схемы линеен — помимо схемы сравнения x и $2^n - 1 - y$ (линейного размера) мы используем дополнительно для каждого суффикса один блок, изображённый в центре рис. 9, который комбинирует результаты предыдущих сравнений.

1.16. Для вычисления функции MAJ достаточно научиться подсчитывать число единиц среди значений переменных: дальше можно использовать схему из задачи 1.14. Общее число единиц равно сумме числа единиц среди значений переменных от x_1 до $x_{\lfloor n/2 \rfloor}$ и числа единиц среди значений переменных от $x_{\lfloor n/2 \rfloor + 1}$ до x_n. Представляя это в виде схемы, получим схему глубины $\log n$, элементами которой должны быть функции, вычисляющие сумму двух чисел. Поскольку эти числа не превосходят n, их двоичная длина не превосходит $\log n$. Из решения задачи 1.15 вытекает, что глубина таких схем $O(\operatorname{loglog} n)$. Глубина всей схемы поэтому $O(\log n \operatorname{loglog} n)$ (а размер $O(n)$).

1.17. Для графов можно определить операцию возведения в степень. В графе G^k столько же вершин, сколько и в G, а две вершины связаны ребром, если в G их можно соединить путём не длиннее k (в частности, $G^1 = G$).

Графы будем задавать *матрицами смежности.* Строки и столбцы матрицы смежности $A(G)$ графа G индексированы вершинами графа. Если (jk) — ребро G, то $a_{jk} = 1$, в противном случае $a_{jk} = 0$. Нам будет удобно полагать $a_{jj} = 1$ и считать элементы матрицы булевыми переменными.

Легко понять, что если между вершинами в графе G есть путь, то есть и путь не длиннее n, где n — число вершин в графе G. Так что для решения задачи достаточно построить схему, вычисляющую матрицу $A(G^k)$ для какого-нибудь $k \geqslant n$, и схему, выбирающую матричный элемент по заданным номерам строки и столбца.

Для любых положительных k, j, m, таких что $k = j + m$, справедливо тождество

$$A(G^k)_{uv} = \bigvee_{w \in V(G)} A(G^j)_{uw} \wedge A(G^m)_{wv}. \tag{$*$}$$

Если сказать словами, то это тождество означает, что на любом пути длины k в графе G, связывающем вершины u и v, есть вершина w такая, что длина пути от u до w равна j, а длина пути от w до v равна m. В обратную сторону, если есть пути между u и w, w и v длины j и m соответственно, то их объединение даёт путь длины $j + m = k$ из u в v.

Вычисление матричного элемента по формуле $(*)$ легко записать в виде схемы глубины $O(\log n)$. Вычисляя последовательность матриц $A(G^1)$, $A(G^2)$, $A(G^4)$,... последовательным возведением в квадрат, через $\lceil \log n \rceil$ шагов мы получим матрицу $A(G^{2^k})$, где $2^k \geqslant n$. Глубина построенной схемы $\log^2 n$, а размер полиномиален по n.

Теперь покажем, как извлекать из набора матричных элементов элемент a_{jk}, если j и k заданы двоичными представлениями $\overline{j_l \dots j_0}$, $\overline{k_l \dots k_0}$, $l = \lceil \log n \rceil$. Заметим, что равенство двух битов x и y проверяется в стандартном базисе формулой $x \overset{?}{=} y \overset{\text{def}}{=} (\neg x \wedge \neg y) \vee (x \wedge y)$. Поэтому значение, вычисленное по формуле

$$\bigvee_{s_l \dots s_0, t_l \dots t_0} a_{\overline{s_l \dots s_0}, \overline{t_l \dots t_0}} \bigwedge_{m=0}^{l} \left(s_m \overset{?}{=} j_m\right) \wedge \left(t_m \overset{?}{=} k_m\right), \tag{$**$}$$

всегда равно $a_{\overline{j_l \dots j_0}, \overline{k_l \dots k_0}}$. Размер формулы $(**)$ полиномиален по n. Легко представить эту формулу схемой глубины $O(\log n)$. Можно также сослаться на результат задачи 1.19.

1.18. Используя тождества де Моргана
$$\neg \bigvee x_i = \bigwedge \neg x_i, \qquad \neg \bigwedge x_i = \bigvee \neg x_i,$$
можно добиться, чтобы отрицания применялись первыми. А поскольку $\neg f = f \oplus 1$, применяя при необходимости тождества де Моргана ещё раз, можно добиться, чтобы последней применялась дизъюнкция.

После таких преобразований мы получим схему, которая есть ДНФ. И задача свелась к тому, чтобы убедиться, что количество конъюнктов (аргументов дизъюнкции) в любой ДНФ, задающей функцию $PARITY$, экспоненциально велико.

Легко понять, что любой конъюнкт в такой ДНФ должен содержать n сомножителей (в противном случае функция, которую задаёт эта ДНФ, иногда не будет меняться при изменении ровно одного из её аргументов). Конъюнкт, содержащий n сомножителей, равен 1 ровно на одном наборе значений переменных. Поэтому число конъюнктов не меньше числа единиц функции $PARITY$, которое равно 2^{n-1}.

Замечание. Можно доказать, что схемы любой фиксированной глубины из элементов NOT и OR, AND с произвольным числом входов, вычисляющие функцию $PARITY$, имеют экспоненциальный размер. Доказательство строится по индукции, основание которой дают схемы глубины 2 и 3 (изложенный выше случай).

Доказательство этого утверждения можно найти в [24]. Приведём краткое изложение основной идеи. Заметим, что применение дизъюнкции и конъюнкции в схеме минимального размера должно чередоваться. Заменяя дизъюнкцию конъюнкций на конъюнкцию дизъюнкций, можно уменьшить глубину схемы на 2. Однако размер схемы может при этом увеличиться квадратично, поэтому никакой разумной оценки так не получается. Для получения оценки *случайным* образом присвоим значения части переменных. Полученная функция от меньшего числа переменных — либо $PARITY$, либо её отрицание. Схемная сложность для отрицания $PARITY$ такая же, как и для самой $PARITY$. А с некоторой вероятностью наша схема упростится и перестановка конъюнкций и дизъюнкций не приведёт к большому увеличению размера схемы.

1.19. а) $\Longrightarrow$ б). Граф, представляющий формулу, можно сделать деревом, если размножить входные переменные. Размер при этом увеличится не более, чем вдвое.

Для построения схемы глубины $O(\log n)$, вычисляющей формулу X размера n, используем идею, применённую в решении задач 1.14 и 1.15.

Двигаясь от корня дерева, представляющего формулу, и выбирая каждый раз вершину, соответствующую подформуле бóльшего размера, мы найдём рано или поздно подформулу Z, размер которой лежит между $n/3$ и $2n/3$. Заменяя в формуле X подформулу Z независимой переменной z, получаем формулу Y с такими же оценками на размер.

Пусть для Z и Y есть вычисляющие их схемы глубины не больше h. Построим для всей формулы X схему глубины не больше $h+3$. Вычислим 3 переменные подсхемами глубины не больше h: y_0 — значение формулы Y при $z=0$, y_1 — значение формулы Y при $z=1$, s — значение формулы Z. Значение f всей формулы равно
$$f = (y_0 \wedge \neg s) \vee (y_1 \wedge s)\,.$$
Это формула глубины 3.

Итак, для $h(L)$ — минимальной глубины схемы, вычисляющей формулу размера L, выполнено рекуррентное соотношение:
$$h(L) \leqslant h(\frac{2L}{3}) + 3.$$
Из него следует $h(L) = O(\log L)$.

б) $\Longrightarrow$ а). Это совсем просто. Превратим граф схемы в дерево, размножая при необходимости вершины. Размер этого дерева не будет превышать количества ориентированных путей от выхода ко входам. А таких путей не более 2^h.

1.20. Вспомним конструкцию неразрешимого предиката $f_\varphi(x)$, принадлежащего P/poly, которая приведена в замечании 1.1 на с. 26.

Сейчас мы будем строить такой предикат $f_\varphi(x)$, чтобы он был разрешим, но не принадлежал P. Мы построим такую вычислимую функцию $\varphi(n)$, что любой алгоритм её вычисления работает дольше, чем 2^n. Другими словами, есть алгоритм распознавания принадлежности языку H, состоящему из двоичных записей тех чисел n, для которых $\varphi(n)=1$; но время работы в наихудшем случае любого такого алгоритма на словах длины m растёт быстрее, чем 2^{2^m}.

Докажем более общее утверждение. Пусть $f(n)$ — вычислимая функция. Обозначим через $\mathcal{M}_f$ язык, состоящий из таких пар $([M], x)$, что машина M на входе x останавливается за время $f(|x|)$. Принадлежность этому языку разрешима (запустим универсальную машину Тьюринга со входом $([M], x)$, отсчитаем $f(|x|)$ тактов работы M и посмотрим — остановилась ли M). Но быстро проверить эту разрешимость нельзя, как показывает следующее рассуждение, почти дословно повторяющее решение задачи 1.3.

Пусть машина A распознаёт принадлежность языку $\mathcal{M}_f$ слов длины n за время $T(n)$. Тогда есть и такая машина A', которая на входе X запускает A на входе (X, X), после чего в случае ответа «да» переходит в состояние, в котором головка двигается вправо (и машина не останавливается), а в случае ответа «нет» останавливается. Смоделировать работу A за время $T(n)$ можно за время $T'(n) = O(T^2(n))$. Если $T'(n) < f(n)$, то что скажет A о слове $([A'],[A'])$? Если «да», то приходим к противоречию с определением машины A', если «нет» — тоже приходим к противоречию. Поэтому $T'(n) \geqslant f(n)$, а $T(n) = \Omega(f^{1/2}(n))$.

Итак, мы доказали, что время работы любого алгоритма, распознающего принадлежность слова языку $\mathcal{M}_f$ не меньше, чем $\Omega(f^{1/2})$.

Взяв в качестве f функцию 2^{2^n}, получим решение задачи.

Из раздела 2

2.1. Напомним, что литералом называется переменная или её отрицание. Литералы будем обозначать $l_1, l_2, \ldots$. Алгоритм решения задачи 2-КНФ будет работать в три этапа.

Этап 1. Перебираем все пары дизъюнкций. Если встречаем пару $l_1 \vee x$, $l_2 \vee \neg x$, то добавляем к КНФ дизъюнкцию $l_1 \vee l_2$. Если все пары (включая добавленные дизъюнкции) просмотрены, то переходим к этапу 2.

Этап 2. Проверяем для каждой пары переменных, сколько дизъюнкций использует эту пару. Если каждая пара используется не более одного раза, то выдаём ответ «да» (КНФ выполнима). Если нашлась пара переменных, используемая по крайней мере в двух дизъюнкциях, выполняем сокращение: делаем подстановку равенств, указанных в третьей строчке следующей таблицы, во все дизъюнкции и если получаем противоречие $((x = 0) \wedge (x = 1)$ или $(x = \neg y) \wedge (x = y))$, то выдаём ответ «нет», в противном случае переходим к этапу 3.

$x \vee y$	$x \vee y$	$x \vee y$	$\neg x \vee y$	$\neg x \vee \neg y$	$\neg x \vee \neg y$
$\wedge$	$\wedge$	$\wedge$	$\wedge$	$\wedge$	$\wedge$
$x \vee \neg y$	$\neg x \vee y$	$\neg x \vee \neg y$	$x \vee \neg y$	$\neg x \vee y$	$x \vee \neg y$
$x = 1$	$y = 1$	$x = \neg y$	$x = y$	$x = 0$	$y = 0$

Этап 3. Решаем полученную на этапе 2 задачу с меньшим числом переменных и повторяем её ответ.

Корректность такого алгоритма вытекает из следующих наблюдений. Во-первых, в силу логического тождества

$$\big((l_1 \vee x) \wedge (l_2 \vee \neg x)\big) \Longrightarrow (l_1 \vee l_2),$$

на первом этапе мы добавляем следствия уже имеющихся конъюнкций. Во-вторых, для каждого столбца приведенной выше таблицы легко проверить, что конъюнкция первых двух строчек эквивалентна равенству в третьей строчке. Поэтому выполнимость исходной КНФ эквивалентна выполнимости сокращенной КНФ.

Наконец, докажем, что если на каждой паре переменных есть не более одной дизъюнкции, то КНФ, полученная ко второму этапу, выполнима. Присвоим значение переменной x_1 произвольным образом, затем учтём все следствия этого присвоения. Первой из переменных, которой не присвоено значение, также присвоим значение произвольным образом и т. д. Когда мы не получим выполняющего набора значений переменных? Только в том случае, если для некоторой переменной из значений ранее определённых переменных будут следовать и 0, и 1. Рассмотрим самое раннее такое противоречие. Оно означает, что имеются дизъюнкции $l_1 \vee x$ и $l_2 \vee \neg x$, а также, что $l_1 \vee l_2 = 0$. Но это невозможно, так как в этом случае дизъюнкция $l_1 \vee l_2$ также входит в КНФ после этапа 1.

Теперь оценим время работы алгоритма в худшем случае. Обозначим его через $T(n)$, где n — число переменных (число переменных заведомо не превосходит длины входа). По построению имеем следующее рекуррентное соотношение

$$T(n) \leqslant O(n^4) + T(n-1),$$

откуда следует $T(n) = O(n^5)$.

2.2. *Степенью* вершины в графе называется количество рёбер, выходящих из этой вершины. Необходимым условием существования эйлерова пути является *связность* графа: должен существовать путь из любой вершины в любую. Чтобы выяснить, существует ли эйлеров путь в связном графе, достаточно подсчитать количество вершин нечётной степени в этом графе. Если оно не превосходит 2, то эйлеров путь есть, в противном случае — нет.

Вторая часть этого утверждения очевидна: если есть эйлеров путь, то все вершины графа имеют чётную степень, кроме начальной и конечной вершин эйлерова пути (если начальная и конечная вершины совпадают, то степени всех вершин чётны).

Доказательство существования эйлерова пути основано на таком простом наблюдении: если есть два замкнутых пути без общих рёбер, но с общей вершиной, то их можно объединить в один путь (идём по первому пути до общей вершины, потом обходим второй путь и продолжаем движение по первому). Далее нужно применить индукцию по числу вершин и рёбер в графе. Кратко изложим это индуктивное рассуждение (читателю рекомендуется восстановить опущенные детали самостоятельно).

Если все вершины графа G чётной степени, то выберем в нём какой-нибудь замкнутый путь C. Выбросив рёбра C из графа G, получим некоторое количество связных графов $G_1, \ldots, G_r$, у которых все степени вершин чётные. Так что в каждом из этих графов есть эйлеров путь (по индукции). Каждый из таких графов имеет хотя бы одну вершину на C (иначе G был бы несвязным). Поэтому можно применить изложенную выше конструкцию, чтобы построить из C и эйлеровых путей в графах G_j эйлеров путь в G.

Если в графе G есть две[16] вершины нечётной степени, то выберем связывающий их путь C. Дальнейшее рассуждение остается прежним.

Осталось заметить, что и подсчёт степени вершины, и проверка связности графа могут быть выполнены за полиномиальное время.

2.3. Рассмотрим предикат $Q \in \mathrm{NP}$. По определению

$$Q(x) = \exists\, y\ \big((|y| < q(|x|)) \wedge R(x, y)\big),$$

где $q(\cdot)$ — полином, а $R(\cdot,\cdot) \in \mathrm{P}$. В другой интерпретации слово y — это сообщение Мерлина, доказывающее истинность $Q(x)$.

Предположим, что Артур полностью доверяет Мерлину, а тот имеет право отвечать только одним битом (и всегда даёт правильный ответ на поставленный вопрос). Тогда Артур может восстановить слово y, выясняя его биты от первого до последнего, следующим образом. Если уже известны первые k битов, образующие слово u, то Артур может поинтересоваться у Мерлина, существует ли слово $u0z$, такое что $R(x, u0z)$. При положительном ответе $(k+1)$-й бит полагается равным 0, при отрицательном — 1.

Если $\mathrm{P} = \mathrm{NP}$, то Артур может имитировать Мерлина.

2.4. Принадлежность задачи о паросочетаниях классу NP очевидна: Мерлину достаточно разбить мальчиков и девочек на пары, после чего Артур сможет убедиться, что пар ровно n и все выбранные пары согласны танцевать.

Доказательство принадлежности P будем проводить, переформулировав задачу в терминах теории графов. Есть *двудольный граф* (рёбра соединяют только вершины из разных долей), нужно проверить, существует ли *совершенное паросочетание*, т. е. такой набор рёбер, что каждая вершина инцидентна ровно одному ребру из набора (если каждая вершина графа инцидентна не более чем одному ребру из набора, то такой набор называется *паросочетанием*, размер паросочетания — количество входящих в него рёбер).

[16] Если у вас получился граф, у которого ровно одна вершина нечётной степени, то проверьте вычисления и найдите ошибку.

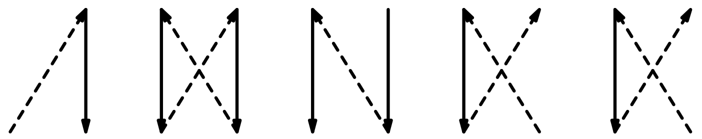


Рис. 10.

Мы рассмотрим алгоритм, последовательно увеличивающий текущее паросочетание, начиная с пустого. Построение завершается либо совершенным паросочетанием, либо доказательством максимальности размера текущего паросочетания (и тогда совершенного паросочетания в графе нет).

Алгоритм будет использовать для увеличения размера текущего паросочетания построение *чередующего пути* (это такой путь в графе, который начинается и заканчивается в вершинах, не инцидентных рёбрам паросочетания, и проходит попеременно по рёбрам, принадлежащим и не принадлежащим паросочетанию). Заменив в паросочетании рёбра чередующего пути, стоящие на чётных местах, на рёбра, стоящие на нечётных местах, мы увеличим размер паросочетания.

Обозначим текущее паросочетание через C, а множество вершин, инцидентных рёбрам из C, — через $W(C)$. Предположим, что в графе есть паросочетание C' бо́льшего размера. Рассмотрим граф A, образованный объединением рёбер C и C'. Степени вершин в таком графе не превосходят 2, поэтому его компоненты связности — это пути (возможно, длины 1, т. е. рёбра) и циклы (см. рис. 10, где сплошными линиями изображены рёбра из C, а штриховыми — из C', на стрелки пока не нужно обращать внимание). Так как C' имеет больше рёбер, чем C, то в A найдётся компонента связности, содержащая больше рёбер из C', чем из C. Такая компонента является, как нетрудно видеть, чередующим путём.

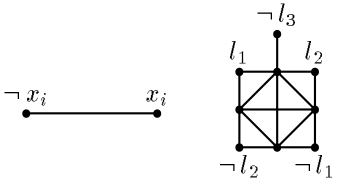
Итак, проверка максимальности заданного паросочетания C равносильна проверке существования чередующего пути. Последнюю можно сделать, например, так. Обозначим доли графа через V и U. *Ориентируем* рёбра графа согласно следующему правилу: рёбра, входящие в C, направлены из V в U, а все остальные рёбра направлены противоположно (эта ориентация также изображена на рис. 10). Тогда существование чередующего пути равносильно существованию такой пары вершин $u \in U \setminus W(C)$ и $v \in V \setminus W(C)$, что из u в v есть ориентированный путь. Существование ориентированного пути с началом и концом в заданных вершинах проверяется способом, описанным в решении задачи 1.17, который работает и в случае ориентированного графа (матрица смежности становится несимметричной, что никак не влияет на вычисления).

Итак, мы доказали, что задача о паросочетаниях принадлежит P. Описанный выше алгоритм не оптимален, читателю предлагается подумать, как можно его ускорить.

2.5. б) Удобнее описывать сведение 3-КНФ к задаче НЕЗАВИСИМОЕ МНОЖЕСТВО. В этой задаче по графу G и числу q требуется определить, существует ли в графе G множество вершин мощности q, никакая пара вершин которого не связана ребром (*независимое множество*).

Очевидно, что задача КЛИКА для графа G эквивалентна задаче НЕЗАВИСИМОЕ МНОЖЕСТВО для дополнительного графа $\bar{G}$ (в дополнительном графе рёбра и нерёбра меняются местами).

Возьмём 3-КНФ из m дизъюнкций, в которые входят n переменных. Граф, который сопоставляется этой КНФ, имеет $2n + 4m$ вершин. Каждой переменной x_i соответствуют две вершины, связанные ребром. Пометим их x_i и $\neg x_i$ (см. рис. 11а). Каждой дизъюнкции соответствуют четыре вершины, попарно связанные рёбрами между собой. Вершины, соответствующие переменным и дизъюнкциям, связаны между собой так, как показано на рис. 11б), (нарисован пример для дизъюнкции $l_1 \vee l_2 \vee l_3$).



переменная а) дизъюнкция б)

Рис. 11.

Очевидно, что такой граф строится по 3-КНФ полиномиальным алгоритмом.

По построению графа ясно, что любое независимое множество его вершин содержит не более $n + m$ элементов. Докажем, что независимые множества размера $n + m$ находятся во взаимно однозначном соответствии с выполняющими наборами значений для 3-КНФ, которое задаётся правилом: если из пары вершин, помеченных x_i и $\neg x_i$, в независимое множество входит вершина x_i, то в соответствующем этому независимому множеству наборе значений $x_i = 1$, в противном случае $x_i = 0$.

Из рис. 11б) легко усматривается корректность такого соответствия. В независимое множество размера $n + m$ обязана входить хотя бы одна вершина из четвёрки, соответствующей дизъюнкции. Но это означает, что хотя бы одна из вершин, помеченных $\neg l_1$, $\neg l_2$, $\neg l_3$, в это множество не входит. И наоборот, если взять множество S вершин, соответствующих выполняющему набору для 3-КНФ, то оно однозначно дополняется до независимого множества размера $n + m$ (проверьте по рис. 11б), что для каждого набора значений l_1, l_2, l_3 (кроме нулевого набора!) из четвёрки, соответствующей дизъюнкции, однозначно выбирается вершина, которую можно добавить к S).

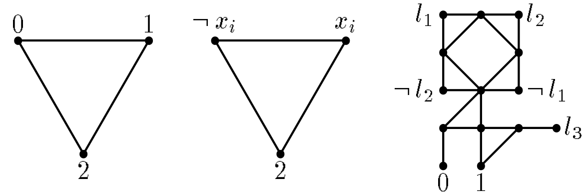
2.6. б). Заметим, что число раскрасок в 3 цвета по очевидным причинам кратно 6: перестановка цветов сохраняет правильную раскраску.

Пусть есть 3-КНФ из m дизъюнкций, в которые входят n переменных. Граф, который сопоставляется этой КНФ, имеет $7m + 2n + 3$ вершин. Для описания структуры графа удобно пометить часть вершин. Во-первых, пометим три вершины числами 0, 1, 2. Во-вторых, пометим каждым литералом (переменной или её отрицанием) по одной вершине. Остальные $7m$ вершин разобьём на группы по 7, каждая группа соответствует одной из дизъюнкций.

Теперь опишем рёбра этого графа. Как показано на рис. 12а), вершины 0, 1 и 2 соединены между собой рёбрами. На рис. 12б) показаны ещё n треугольников в этом графе. И, наконец, рис. 12в) показывает, как соединены рёбрами вершины, соответствующие каждой дизъюнкции. На этом рисунке l_1, l_2, l_3 обозначают литералы, входящие в дизъюнкцию. Заметим, что некоторые рёбра мы перечислили по нескольку раз.

Очевидно, что описанное выше построение можно выполнить за полиномиальное от n и m время. Докажем его корректность.

Рассмотрим, какие правильные раскраски в 3 цвета возможны для такого графа. Без ограничения общности можно считать, что вершины 0, 1 и 2 покрашены в цвета 0, 1 и 2 (из шести раскрасок, различающихся перестановками цветов мы выбрали одну). Тогда вершины, помеченные x_i и $\neg x_i$, покрашены в цвета 0 и 1, причём их цвета должны быть противоположны (см. рис. 12б).



константы а) переменная б) дизъюнкция в)

Рис. 12.

Прямым перебором вариантов можно проверить, что граф, изображённый на рис. 12в) удовлетворяет следующему свойству: если красить вершины, отмеченные литералами, в цвета 0 или 1, а вершины, отмеченные отрицаниями литералов, — в противоположные цвета 1 или 0, то правильная раскраска остальных вершин этого графа в 3 цвета существует (и единственна!) тогда и только тогда, когда хотя бы одно из значений литералов отлично от 0.

Поэтому правильные 3-раскраски построенного графа, для которых вершины 0, 1, 2 покрашены в цвета 0, 1, 2 соответственно, находятся во взаимно однозначном соответствии с выполняющими наборами значений переменных исходной 3-КНФ.

2.7. Данная задача содержит ограничения разного вида, бороться с которыми удобнее по отдельности. Поэтому мы сформулируем промежуточную задачу и построим цепочку полиномиальных сводимостей задач.

Итак, назовём данную задачу RP (“restricted puzzle”). Задача UP (“unrestricted puzzle”) получается, если отказаться от ограничения на число типов квадратиков и требовать сложить прямоугольник заданных размеров, причём не из всего набора квадратиков, а из некоторого его подмножества.

Опишем неформально цепочку сводимостей

$$3\text{-КНФ} \propto \mathrm{UP} \propto \mathrm{RP},$$

которая доказывает NP-полноту исходной задачи.

3-КНФ $\propto$ UP. Для описания этой сводимости будем считать квадратики роботами, которые могут получать и передавать сообщения через стороны. Пара букв допустима, если одна из них означает передачу сообщения, а вторая — приём того же сообщения.

Итак, пусть есть 3-КНФ, множество переменных которой обозначим через X, а множество дизъюнкций — через $\mathcal{D}$. Через $\mathcal{D}'$ обозначим множество выполнимых дизъюнкций, получаемых из $\mathcal{D}$ подстановкой значений некоторых переменных.

Будем теперь описывать множество квадратиков и их типы, из которых нужно будет складывать прямоугольник размера $|X| \times |\mathcal{D}|$. Одна сторона этого прямоугольника (для определённости — левая) соответствует переменным КНФ, а другая (нижняя) — дизъюнкциям. Роботы-квадратики получают слева сообщения «$x_j = 1$» или «$x_j = 0$», $x_j \in X$; передают вправо то же самое сообщение; получают снизу дизъюнкцию из $\mathcal{D}'$; подставляют в неё, если возможно, переменную x_j и передают полученную дизъюнкцию вверх, если не получено нулевое значение (другими словами, отсутствуют типы, получающие слева сообщения вида «$x_j = \alpha$», а снизу — «$\alpha \oplus x_j$»). Все эти типы не содержат граничных букв. А есть ещё 4 угловых типа, двум сторонам которых разрешено выходить на границу, и $2|X|+2|\mathcal{D}|$ граничных типов (одна сторона может выходить на границу), которые нумеруются переменными и дизъюнкциями соответственно. Буквы, написанные на сторонах квадратиков граничных типов, обеспечивают однозначную сборку контура прямоугольника. Квадратик каждого типа один. А всего типов $O(|X| \cdot |\mathcal{D}|)$.

Таким образом, выполнимость КНФ эквивалентна возможности сложить прямоугольник размера $(|X| + 2) \times (|\mathcal{D}| + 2)$ из описанного выше набора типов квадратиков.

UP $\propto$ RP. Пусть есть множество квадратиков, принадлежащих множеству типов T, каждого типа t по $n(t)$ штук ($\sum_{t \in T} n(t) = N$), из которых нужно сложить прямоугольник $m \times k$. Без ограничения общности можно считать, что $N > 5$. Добавим квадратиков: по 2 штуки из $m + k$ дополнительных типов c_j и r_l, чтобы выложить внешний контур прямоугольника $m \times k$, $4(N - mk)$ квадратиков ещё одного типа u, чтобы можно было использовать квадратики, не вошедшие в прямоугольник, и $(5N + 3)^2 - 5N - 2(m + k) + 4mk$ квадратиков типа v для получения квадрата. Поскольку $5N+3 > |T|+2(m+k)+2$, ограничение числа типов длиной стороны квадрата будет выполнено.

Сформулированные условия прямо переводятся на язык букв и их сочетаний. На сторонах квадратиков типа v написана одна и та же буква v, которая может соседствовать только сама с собой; на одной стороне квадратика типа u написана буква u, которая может соседствовать со всеми буквами, а на остальных — v; наконец, на одной стороне квадратиков типов c_j и r_j написана буква v, а на остальных сторонах написаны буквы, обеспечивающие сборку контура прямоугольника $m \times k$.

Таким образом, чтобы собрать квадрат $(5N+3) \times (5N+3)$ из нового набора квадратиков 1×1, необходимо и достаточно уметь собирать прямоугольник $m \times k$ из некоторого подмножества исходного набора.

2.8. Поскольку умножение чисел можно произвести за полиномиальное время, Мерлин может сообщить Артуру любое разложение числа n на два множителя.

2.9. Докажем принадлежность задачи ПРОСТОТА ЧИСЛА классу NP, построив для решения этой задачи недетерминированную машину Тьюринга N, которая рекурсивно вызывает саму себя.

На вход машине N подаётся двоичная запись числа p, простоту которого нужно проверить. Машина недетерминированно дописывает первообразный корень g по модулю p и разложение на простые множители числа $p-1$, т.е. такие числа $p_1, \ldots, p_s, \alpha_1, \ldots, \alpha_s$, что

$$p-1=\prod_{j=1}^{s} p_j^{\alpha_j}. \qquad (*)$$

После этого машина проверяет равенство $(*)$ (на это хватит времени $O(n^3)$, где $n=\log p$, даже если умножать большие числа «в столбик»), проверяет равенство $g^{p-1}\equiv 1 \pmod p$ и все неравенства

$$g^{(p-1)/p_j}\not\equiv 1 \pmod p.$$

Для этих проверок потребуется время $O(n^4)$, так как $s=O(n)$, а возведение в степень q требует $O(\log q)$ умножений. Далее машина проверяет простоту всех p_j, рекурсивно вызывая саму себя.

Описанные выше проверки гарантируют, что порядок числа g в группе $(\mathbb{Z}/p\mathbb{Z})^*$ равен $p-1$, что эквивалентно простоте p.

Оценим теперь время работы такой НМТ на входе длины n. Оно складывается из времени, затрачиваемого на детерминированные действия, и времени, затрачиваемого на недетерминированные действия. Как следует из приведенных выше оценок, количество детерминированных действий ограничено полиномом от длины записей всех чисел, проверяемых на простоту за время работы. Длина записей первообразных корней и кратностей также ограничена полиномом (и даже линейно) от длины записей всех чисел, проверяемых на простоту.

Таким образом, осталось оценить длину записей чисел, проверяемых на простоту. Если взять произведение всех чисел, проверяемых на простоту на k-м уровне рекурсии, то оно заведомо меньше исходного числа p. Поэтому суммарная длина записей этих чисел не более чем вдвое превышает длину записи p (длина записи произведения двух чисел разве что на 1 меньше суммы длин сомножителей). Максимальное из чисел, проверяемых на простоту на k-м уровне рекурсии, по крайней мере вдвое меньше, чем максимальное из чисел, проверяемых на $(k-1)$-м уровне. Поэтому максимальный уровень рекурсии не превосходит $\log p$. Следовательно, общая длина записей всех чисел, проверяемых на простоту при входе p, равна $O(\log^2 p)$.

Из раздела 4

4.1. Заметим прежде всего, что S заведомо не меньше длины входа в силу используемых определений.[17)]

[17)] Бывают и другие определения. Так, например, класс log-SPACE определяется с помощью двухленточных машин, у которых на одной ленте записан вход и с неё можно только читать, а на второй ленте используется память $O(\log n)$.

Предположим, что доказана верхняя оценка вида $2^{\mathrm{poly}(S)}$ на время работы недетерминированной машины. Тогда можно дословно повторить доказательство теоремы 4.2, построить игру длины $\mathrm{poly}(S)$, и вычислить её результат детерминированной машиной на памяти $\mathrm{poly}(S)$.

Пусть есть НМТ, работающая на памяти S. В процессе работы она может находиться не более чем в $N=|\mathcal{A}|^S\cdot|\mathcal{Q}|\cdot S$ состояниях, где $\mathcal{Q},\mathcal{A}$ — множество состояний управляющего устройства и внешний алфавит, соответственно. Хотя у НМТ может быть несколько вариантов перехода из данного состояния, из последовательности переходов всегда можно выбросить циклы (ни к чему делать вначале неправильный переход, а затем, вернувшись в то же состояние, — правильный). Так что время работы НМТ ограничено $2^{O(S)}$, как и для детерминированной МТ.

4.2. Легко сообразить, как имитировать оракул F из Σ_k, имея возможность заказывать оракул из Π_k. Нужно заказать оракул $\neg F$, а дальше брать отрицание от каждого результата его работы.

Класс P^{Σ_k}, как и P, замкнут относительно взятия дополнений. Так что осталось доказать включение $P^{\Sigma_k}\subseteq\Sigma_{k+1}$. Это удобно делать, используя игровое определение классов Σ_k.

Пусть есть предикат $F\in P^{\Sigma_k}$, а вычисляющий его полиномиальный алгоритм использует оракул $G\in\Sigma_k$. Игру, которая задаёт $G(x)$, обозначим $\mathcal{G}(x)$. Опишем теперь игру $\mathcal{F}(x)$, выигрыш белых в которой (точнее, наличие выигрышной стратегии) эквивалентен $F(x)$.

Первый ход белых в этой игре состоит в объявлении последовательности пар $(f_j,x_j)_{j=1}^q$. Белые утверждают, что эта последовательность есть протокол обращений к оракулу в процессе работы алгоритма с оракулом G, вычисляющего $F(x)$. Более точно это означает, что алгоритм обращается к оракулу q раз, j-й запрос делается о слове x_j, оракул отвечает на это запрос f_j, и окончательный результат $F(x)=1$. Следующим ходом чёрные объявляют индекс j и, если $f_j=0$, то дополнительно первый ход белых в игре $\mathcal{G}(x_j)$. Далее, если $f_j=1$, то разыгрывается игра $\mathcal{G}(x_j)$, а если $f_j=0$, то разыгрывается игра $\mathcal{G}(x_j)$ со сдвигом «на темп»: $(i+1)$-й ход белых в $\mathcal{F}(x)$ в этом случае соответствует i-у ходу чёрных в $\mathcal{G}(x_j)$, а $(i+1)$-й ход чёрных — $(i+1)$-у ходу белых в $\mathcal{G}(x_j)$, $(k+1)$-й ход чёрных на результат влияния не оказывает.

Белые выигрывают в этой игре, если $(x_j)_{j=1}^q$ — правильная последовательность аргументов при обращениях к оракулу, результат игры $\mathcal{G}(x_j)$ совпадает с f_j, а результат работы алгоритма для вычисления F на входе x с ответами оракула $(f_j)_{j=1}^q$ равен 1.

Достаточно ясно, что если $F(x)=1$, то у белых есть выигрышная стратегия в $\mathcal{F}(x)$: первым ходом сказать правду, а дальше играть в $\mathcal{G}(x_j)$ по стратегиям, существование которых вытекает из равенства $f_j = G(x_j)$. Если же $F(x)=0$, то в каком-то члене (f_m, x_m) последовательности $(f_j, x_j)_{j=1}^{q}$ белые должны отклониться от истины. Чёрные своим ходом должны объявить m (и, дополнительно, первый ход за белых в игре $\mathcal{G}(x_m)$ при необходимости), дальнейшая их стратегия состоит в том, чтобы доказывать $f_j \neq G(x_j)$, пользуясь соответствующими стратегиями для $\mathcal{G}(x_j)$.

Из раздела 6

6.1. Поскольку конъюнкция и отрицание образуют полный базис для обычных схем, из лемм 6.1, 6.2 следует, что достаточно реализовать функции $\neg_{\oplus}\colon (x,y)\mapsto (x, x\oplus y\oplus 1)$ и $\bigoplus$. Заметим, что $\neg_{\oplus}[1,2] = \neg[2]\bigoplus[1,2]$, поэтому достаточно реализовать $\bigoplus$. Для этого заведём вспомогательный бит u, содержащий константу 1 (для этого в самом начале и в самом конце применяем $\neg[u]$). Тогда $\bigoplus[1,2] = \wedge_{\oplus}[u,1,2]$.

Из раздела 7

7.1. Из доказательства теоремы 7.1 следует, что достаточно научиться реализовывать все операторы вида $\Lambda(X)$, $X\in \mathbf{U}(2)$ (управляемый двумя q-битами фазовый сдвиг на $-i$ является частным случаем: $\Lambda^2(-i) = \Lambda(K^{-1})$). Для алгоритма построения схемы требуется также конструктивное доказательство леммы 7.1.

Сперва реализуем управляемый фазовый сдвиг:

$$\Lambda(P(\varphi))[1,2] = E(\varphi)[1], \quad \text{где } P(\varphi) = \begin{pmatrix} e^{i\varphi} & 0 \\ 0 & e^{i\varphi}\end{pmatrix}, \quad E(\varphi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi}\end{pmatrix}.$$

Поскольку $\Lambda(XY) = \Lambda(X)\Lambda(Y)$, то остаётся реализовать операторы Y из $\mathbf{U}(2)/\mathbf{U}(1)$, где $\mathbf{U}(1)$ — подгруппа фазовых сдвигов. Можно считать, что $Y\in\mathbf{SU}(2)$. Эта реализация изображена на рис. 13. Используемые в ней операторы A и B должны удовлетворять уравнению

$$A\sigma^x A^{-1} B\sigma^x B^{-1} = Y. \qquad (*)$$

Геометрически уравнение $(*)$ эквивалентно такому утверждению: любое вращение трёхмерного пространства есть композиция двух поворотов на угол 180°. Доказательство этого утверждения можно усмотреть из рис. 14.

Осталось доказать лемму 7.1 конструктивно. Для начала заметим, что для любых чисел c_1, c_2 существует унитарная матрица V размера

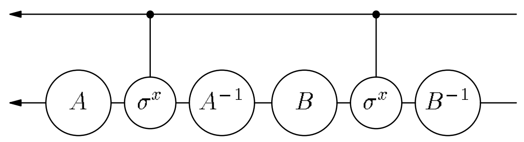


Рис. 13.

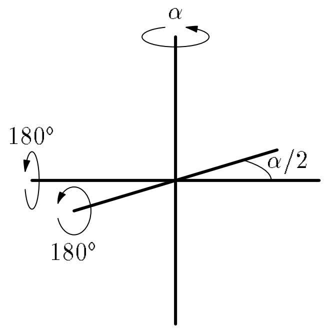


Рис. 14.

2×2 (эффективно вычислимая с любой заданной точностью δ), такая что

$$V\begin{pmatrix} c_1 \\ c_2\end{pmatrix} = \begin{pmatrix}\sqrt{|c_1|^2+|c_2|^2} \\ 0\end{pmatrix}.$$

Следовательно, для любого единичного вектора $|\xi\rangle\in\mathbb{C}^M$ существует последовательность унитарных матриц $V^{(1)},\dots,V^{(M-1)}$, такая что $V^{(1)}\cdot\ldots\cdot V^{(M-1)}|\xi\rangle = |1\rangle$, где $V^{(s)}$ действует на подпространстве $\mathbb{C}(|s\rangle, |s+1\rangle)$ (как матрицы в условии леммы 7.1) и оставляет неизменными остальные базисные векторы.

Пусть теперь задана унитарная матрица U размера $M\times M$. Умножая U^{-1} слева на подходящие матрицы $U^{(1,1)},\dots,U^{(1,M-1)}$, можно перевести первый столбец в вектор $|1\rangle$. При этом столбцы остаются ортогональными, поэтому первая строка переходит в $\langle 1|$. Действуя таким же образом с остальными столбцами, получаем набор матриц $U^{(j,s)}$ $(1\leqslant j\leqslant s\leqslant M-1)$ (где $U^{(j,s)}$ действует на $|s\rangle$ и $|s+1\rangle$), удовлетворяющий условию

$$U^{(M-1,M-1)}U^{(M-2,M-2)}U^{(M-2,M-1)}\cdot\ldots\cdot U^{(1,1)}\cdot\ldots\cdot U^{(1,M-1)}U^{-1} = I.$$

Этот набор строится алгоритмом сложности $O(M^3)\cdot\mathrm{poly}(\log(1/\delta))$.

7.2. Неравенство (7.6) следует из цепочки неравенств, справедливых для любого $|\xi\rangle$:

$$\big\|XY|\xi\rangle\big\| \leqslant \|X\|\cdot\big\|Y|\xi\rangle\big\| \leqslant \|X\|\cdot\|Y\|\cdot\big\||\xi\rangle\big\|.$$

Для доказательства равенства (7.7) заметим, что собственные числа операторов $XX^\dagger$ и $X^\dagger X$ совпадают.

И, наконец, равенство (7.8) следует из того, что собственные числа оператора $(X \otimes Y)^\dagger (X \otimes Y) = X^\dagger X \otimes Y^\dagger Y$ имеют вид $\lambda_j \mu_k$, где λ_j и μ_k — собственные числа $X^\dagger X$ и $Y^\dagger Y$, соответственно.

7.3. Достаточно проверить для двух сомножителей. Имеем

$$\tilde{U}_2 \tilde{U}_1 \left(|\xi\rangle \otimes |0^{N-n}\rangle\right) = \tilde{U}_2 \left(U_1|\xi\rangle \otimes |0^{N-n}\rangle + |\eta_1\rangle\right) = \\ = U_2 U_1 |\xi\rangle \otimes |0^{N-n}\rangle + |\eta_2\rangle + \tilde{U}_2 |\eta_1\rangle,$$

где $\big\| |\eta_j\rangle \big\| \leqslant \delta_j$ $(j = 1, 2)$. Поэтому

$$\left\| \tilde{U}_2 \tilde{U}_1 \left(|\xi\rangle \otimes |0^{N-n}\rangle\right) - U_2 U_1 |\xi\rangle \otimes |0^{N-n}\rangle \right\| \leqslant \delta_1 + \delta_2.$$

7.4. Обозначим $\mathcal{M} = \mathcal{B}^{\otimes n} \otimes |0^{N-n}\rangle$. Будем искать оператор в виде $W = (U \otimes I_{[n+1,\ldots,N]})\Pi_{\mathcal{M}} + \tilde{W}(I - \Pi_{\mathcal{M}})$, где унитарный оператор $\tilde{W}$ сохраняет $\mathcal{M}^\perp$. Для такого W, очевидно, выполняется равенство

$$W\left(|\xi\rangle \otimes |0^{N-n}\rangle\right) = (U|\xi\rangle) \otimes |0^{N-n}\rangle,$$

а $\|W - \tilde{U}\| \leqslant O(\delta)$ эквивалентно тому, что для всех $|\eta\rangle \in \mathcal{M}^\perp$ выполняется $\big\|(\tilde{W} - \tilde{U})|\eta\rangle\big\| = O(\delta)$. Представляя $\tilde{W} = X\tilde{U}$, получаем эквивалентные условия на унитарный оператор X:

$$\|X - I\| = O(\delta), \qquad X\mathcal{L}^\perp = \mathcal{M}^\perp,$$

где $\mathcal{L} = \tilde{U}\mathcal{M}$.

Теперь нам потребуется следующая лемма.

Лемма. *Пусть $\mathcal{L}$ и $\mathcal{M}$ — подпространства конечномерного пространства $\mathcal{N}$, такие что $\|\Pi_{\mathcal{L}} - \Pi_{\mathcal{M}}\| \leqslant \delta$, $\delta < 1/2$. Тогда найдётся унитарный оператор X, такой что $\|X - I\| = O(\delta)$ и $X\mathcal{L} = \mathcal{M}$. (Значит, и $X\mathcal{L}^\perp = \mathcal{M}^\perp$.)*

Доказательство. Возьмём оператор $Y = \Pi_{\mathcal{M}}\Pi_{\mathcal{L}} + (I - \Pi_{\mathcal{M}})(I - \Pi_{\mathcal{L}})$. Сразу видно, что он переводит $\mathcal{L}$ в $\mathcal{M}$ и $\mathcal{L}^\perp$ в $\mathcal{M}^\perp$. Для нормы $\|Y - I\|$ имеем оценку

$$\|Y - I\| = \|\Pi_{\mathcal{M}}\Pi_{\mathcal{L}} - \Pi_{\mathcal{M}} - \Pi_{\mathcal{L}} + \Pi_{\mathcal{M}}\Pi_{\mathcal{L}}\| \leqslant \\ \leqslant \|(\Pi_{\mathcal{M}} - \Pi_{\mathcal{L}})\Pi_{\mathcal{L}}\| + \|\Pi_{\mathcal{M}}(\Pi_{\mathcal{M}} - \Pi_{\mathcal{L}})\| \leqslant 2\delta < 1.$$

Оператор Y не унитарный. Но из приведенной оценки следует, что он невырожденный. Рассмотрим унитарный оператор $X = Y(Y^\dagger Y)^{-1/2}$. Оператор $Y^\dagger Y$ сохраняет подпространство $\mathcal{L}$, поэтому X переводит $\mathcal{L}$ в $\mathcal{M}$. Для оценки нормы X разложим $(Y^\dagger Y)^{-1/2}$ в ряд Тейлора

$$(Y^\dagger Y)^{-1/2} = I + \frac{1}{2}Z + \frac{3}{8}Z^2 + \ldots, \qquad \text{где } Z = I - Y^\dagger Y.$$

Поэтому $\|(Y^\dagger Y)^{-1/2} - I\| \leqslant (1 - \|Z\|)^{-1/2} - 1 = O(\delta)$, отсюда получаем $\|X - I\| = O(\delta)$. □

Чтобы применить лемму, необходимо оценить величину $\|\Pi_{\mathcal{L}} - \Pi_{\mathcal{M}}\|$. Имеем $\|(\tilde{U} - U \otimes I)\Pi_{\mathcal{M}}\| \leqslant \delta$ ($\tilde{U}$ приближает U в расширенном смысле с точностью δ). Обозначая $V = U \otimes I$, получаем

$$\|\Pi_{\mathcal{L}} - \Pi_{\mathcal{M}}\| = \|\tilde{U}\Pi_{\mathcal{M}}\tilde{U}^\dagger - V\Pi_{\mathcal{M}}V^\dagger\| \leqslant 2\delta.$$

Итак, условие леммы выполнено (и задача решена) при $\delta < 1/4$.

7.5. Схему для оператора $\Lambda(U)$ можно построить, используя *элемент Фредкина* $F = \Lambda(\leftrightarrow)$ — управляемый обмен битами. Элемент Фредкина задается соотношениями

$$F\colon |a, b, c\rangle \mapsto \begin{cases} |0, b, c\rangle & \text{если } a = 0, \\ |1, c, b\rangle & \text{если } a = 1. \end{cases}$$

Его можно реализовать следующим образом:

$$F[1,2,3] = \Lambda(\oplus)[1,2,3]\ \Lambda(\oplus)[1,3,2]\ \Lambda(\oplus)[1,2,3]$$

(заметим, что $\Lambda(\oplus) = \Lambda^2(\sigma_x)$ — это элемент Тоффоли).

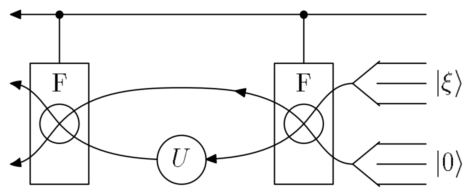


Рис. 15.

На рис. 15 показано, как из схемы для оператора U, сохраняющего $|0\rangle$, построить схему для $\Lambda(U)$. В прямоугольниках происходит управляемый обмен q-битами (параллельно действует нужное количество элементов Фредкина). Если управляющий q-бит равен $|1\rangle$, то на вход схемы, вычисляющей U, будет подан $|\xi\rangle$, в противном случае — $|0\rangle$.

7.6. Каждый из рассматриваемых поворотов порождает всюду плотное подмножество в подгруппе поворотов относительно фиксированной прямой. Поэтому осталось доказать, что повороты относительно двух различных прямых порождают $\mathbf{SO}(3)$. Для этого достаточно доказать, что подгруппа, порождённая всеми поворотами относительно двух различных прямых, действует транзитивно на сфере (или на проективной плоскости — множестве одномерных подпространств). Справедливость этого факта очевидна из рис. 16 (если можно перемещаться по двум семействам параллелей, то из любой точки на сфере можно попасть в любую другую). Строгое доказательство получается аналогично решению задачи 7.7.

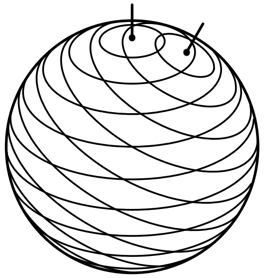


Рис. 16.

Замечание. Это решение неконструктивно: нельзя дать никакой верхней оценки на количество поворотов X, X^{-1}, Y, Y^{-1}, композиция которых приближает заданный элемент $U \in \mathbf{SO}(3)$ с заданной точностью δ. Причина неконструктивности состоит в следующем. Поворот на угол $2\pi\alpha$, где α — иррациональное, порождает всюду плотное подмножество в группе поворотов относительно фиксированной прямой (эта группа, очевидно, изоморфна $\mathbb{R}/\mathbb{Z}$). Однако число α может очень хорошо приближаться рациональными числами (это имеет место, когда коэффициенты цепной дроби, представляющей α, очень быстро растут). Тогда любое $r \in \mathbb{R}/\mathbb{Z}$ приближается элементами вида $n\alpha$ $(n \in \mathbb{Z})$ с любой точностью $\delta > 0$, но число n может быть сколь угодно велико: больше, чем любая наперёд заданная функция δ.

Конструктивное доказательство и эффективный (при фиксированных X и Y) алгоритм построения аппроксимаций довольно сложны [4].

7.7. Обозначим $|\xi'\rangle = V^{-1}|\xi\rangle$, тогда $H' = V^{-1}HV$ — стабилизатор $\mathbb{C}(|\xi'\rangle)$. Так что утверждение задачи приобретает вид: объединение стабилизаторов двух несовпадающих одномерных подпространств порождает $\mathbf{U}(\mathcal{M})$.

Достаточно показать, что группа G, порождённая $H \cup H'$, действует транзитивно на множестве единичных векторов. Действительно, пусть для каждого $|\psi\rangle \in \mathcal{M}$ найдётся оператор $U_\psi \in G$, такой что $U_\psi|\xi\rangle = |\psi\rangle$. Тогда

$$\mathbf{U}(\mathcal{M}) = \bigcup_{|\psi\rangle\in\mathcal{M}} U_\psi H.$$

Доказываем транзитивность действия группы G. Заметим, что

$$H|\psi\rangle = Q(\vartheta) \stackrel{\text{def}}{=} \{|\eta\rangle : |\langle\eta\,|\,\xi\rangle| = \cos\vartheta\},$$
$$H'|\psi\rangle = Q'(\vartheta') \stackrel{\text{def}}{=} \{|\eta\rangle : |\langle\eta\,|\,\xi'\rangle| = \cos\vartheta'\},$$

где ϑ, ϑ' обозначают углы между $|\psi\rangle$ и $|\xi\rangle$, $|\xi\rangle'$ соответственно: $\cos\vartheta = |\langle\psi\,|\,\xi\rangle|$, $\cos\vartheta' = |\langle\psi\,|\,\xi'\rangle|$, $0 \leqslant \vartheta, \vartheta' \leqslant \pi/2$. В последующих формулах используется также угол α между векторами $|\xi\rangle$ и $|\xi\rangle'$: $\cos\alpha = |\langle\xi\,|\,\xi'\rangle|$, $0 \leqslant \alpha \leqslant \pi/2$.

Можно проверить, что при $\dim\mathcal{M} \geqslant 3$

$$HQ'(\vartheta') = \bigcup_{|\alpha-\vartheta'|\leqslant\vartheta\leqslant\min(\alpha+\vartheta',\pi/2)} Q(\vartheta),$$
$$H'Q(\vartheta) = \bigcup_{|\alpha-\vartheta|\leqslant\vartheta'\leqslant\min(\alpha+\vartheta,\pi/2)} Q'(\vartheta').$$

Поэтому

$$H'|\xi\rangle = Q'(\alpha),$$
$$HH'|\xi\rangle = \bigcup_{0\leqslant\vartheta\leqslant\min(2\alpha,\pi/2)} Q(\vartheta),$$
$$H'HH'|\xi\rangle = \bigcup_{0\leqslant\vartheta'\leqslant\min(3\alpha,\pi/2)} Q'(\vartheta'),$$

и т. д. Таким образом, действуя на вектор $|\xi\rangle$ попеременно элементами из H' и H достаточное количество раз, можно получить любой единичный вектор $|\psi\rangle$.

7.8. Поскольку $\sigma^x = HK^2H$, то стандартный базис содержит полный базис для классических обратимых вычислений (см. задачу 6.1). Это, благодаря задаче 7.5, позволяет реализовать операторы $\Lambda(U)$ для всех элементов базиса, кроме H.

Теперь рассмотрим оператор $X = \Lambda(HKH) = H\Lambda(K)H$, который, в силу сказанного, реализуется в стандартном базисе (оператор K сохраняет $|0\rangle$). Подействуем им на $\mathcal{B}^{\otimes 2}$ двумя возможными способами: $X_1 = X[1,2]$, $X_2 = X[2,1]$. Операторы $Y_1 = X_1X_2^{-1}$, $Y_2 = X_2^{-1}X_1$ также реализуются в стандартном базисе.

Заметим, что операторы X_1, X_2 (следовательно, и Y_1, Y_2) сохраняют векторы $|00\rangle$ и $|\eta\rangle = |01\rangle + |10\rangle + |11\rangle$. Кроме того, вычислениями проверяется, что Y_1, Y_2 не коммутируют и имеют, помимо 1, собственные числа $\lambda_{1,2} = (1 \pm \sqrt{-15})/4 = e^{\pm i\varphi/2}$. В $\mathbf{SO}(3) \cong \mathbf{U}(2)/\mathbf{U}(1)$ оператору с такими собственными числами соответствует поворот на угол φ. Поскольку $\lambda_{1,2}$ не являются корнями из 1 (и даже целыми алгебраическими числами, так как их след равен $1/2$), угол φ несоизмерим с π. А поскольку эти операторы не коммутируют, им соответствуют повороты вокруг различных прямых. Поэтому Y_1, Y_2 порождают всюду плотное подмножество в $\mathbf{U}(\mathcal{L})/\mathbf{U}(1)$, где $\mathcal{L} = \mathbb{C}(|00\rangle, |\eta\rangle)^{\perp}$ (см. задачу 7.6).

Для завершения доказательства дважды применим результат задачи 7.7. Операторы Y_1, Y_2 порождают всюду плотное множество в $\mathbf{U}(\mathcal{L})/\mathbf{U}(1)$, оператор $V = \Lambda(K)$ сохраняет $\mathbb{C}(|00\rangle)$ и не сохраняет $\mathbb{C}(|\eta\rangle)$. Так что Y_1, Y_2, $V^{-1}Y_1V$, $V^{-1}Y_2V$ порождают всюду плотное множество в $\mathbf{U}\big(\mathcal{L}\oplus\mathbb{C}(|\eta\rangle)\big)/\mathbf{U}(1)$. Оператор $H[1]$ не сохраняет $\mathbb{C}(|00\rangle)$; применяя результат задачи 7.7 ещё раз, получаем всюду плотное множество в $\mathbf{U}(\mathcal{B}^{\otimes 2})/\mathbf{U}(1)$.

7.9. Из предыдущей задачи следует, что можно реализовать оператор $\Lambda(c)$ с точностью до фазового множителя, $\Lambda(c) = e^{i\varphi}U$. Оператор σ^x реализуется точно. Возьмём дополнительный q-бит в состоянии $|0\rangle$ и применим $\sigma^x U \sigma^x U^{-1}\colon |0\rangle \mapsto c|0\rangle$. Неизвестный фазовый множитель сокращается.

7.10. В базисе $|\xi\rangle = \dfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|\eta\rangle = \dfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ оператор σ^x диагонализуется, так что в этом базисе R имеет вид

$$R = -i\exp(\pi i\alpha\sigma^x) = -i\begin{pmatrix} e^{\pi i\alpha} & 0 \\ 0 & e^{-\pi i\alpha}\end{pmatrix}.$$

А раз α — иррациональное число, степенями R можно приближённо реализовать любой оператор вида $i^s\exp(i\varphi\sigma^x)$, который в указанном выше базисе имеет матрицу $i^s\begin{pmatrix} z & 0 \\ 0 & z^{-1}\end{pmatrix}$, где $s = 0,1,2,3$, $|z| = 1$. В геометрической интерпретации эти операторы соответствуют поворотам вокруг оси x. При $s = 3$ и $z = i$ получаем элемент Тоффоли: $\big(\Lambda^2(R)\big)^k \approx \Lambda^2(\sigma^x)$ (при подходящем k), так что уже имеем полный классический базис. При $s = 1$ и $z = 1$ получаем $\Lambda^2(i) = \Lambda(K)$ (на третий q-бит этот оператор действует тождественным образом). Из $\Lambda(K)$ можно сделать K, подавая на управляющий q-бит константу $|1\rangle = \sigma^x|0\rangle$. С точностью до фазового множителя K — это поворот на $90°$ вокруг оси z, а композициями поворотов вокруг x и одного поворота вокруг z представляются все элементы $\mathbf{SO}(3)$ (аналогично задаче 7.7).

Итак, мы получили реализацию всех операторов из $\mathbf{U}(2)$ с точностью до фазового множителя. Осталось использовать задачу 7.9 для того, чтобы реализовать H.

7.11. Любое вращение трёхмерного пространства представляется как композиция трёх поворотов: на угол α вокруг оси z, затем на угол β вокруг оси x, затем на угол γ вокруг оси z. Поэтому любой оператор, действующий на одном q-бите, представляется в виде

$$U = e^{i\varphi}e^{i(\gamma/2)\sigma^z}e^{i(\beta/2)\sigma^x}e^{i(\alpha/2)\sigma^z}. \qquad (*)$$

Каждый из операторов в правой части $(*)$ выражается через H и управляемые фазовые сдвиги:

$$e^{i\varphi} = \Lambda(e^{i\varphi})\sigma^x\Lambda(e^{i\varphi})\sigma^x, \qquad e^{i\varphi\sigma^z} = \Lambda(e^{-i\varphi})\sigma^x\Lambda(e^{i\varphi})\sigma^x,$$
$$\sigma^x = H\Lambda(e^{i\pi})H, \qquad e^{i\varphi\sigma^x} = He^{i\varphi\sigma^z}H.$$

Таким образом, для решения задачи достаточно построить схему, представляющую управляемый фазовый сдвиг $\Lambda(e^{i\theta})$ с точностью $O(\delta)$.

Выберем такое $q = 2^n$, что $1/\delta \leqslant q < 2/\delta$. Предположим, что у нас в распоряжении есть n-битовый регистр в состоянии

$$|\psi_n(q,k)\rangle = \frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}\exp\big(2\pi i\frac{kj}{q}\big)|j\rangle.$$

Заметим, что $|\psi_n(q,k)\rangle$ — собственный вектор классического оператора $V : |j\rangle \mapsto |(j-1) \bmod q\rangle$:

$$V^l|\psi_n(q,k)\rangle = e^{2\pi i(kl/q)}|\psi_n(q,k)\rangle.$$

Применяя к $|\psi_n(q,k)\rangle$ оператор V^l, управляемый дополнительным q-битом, получаем искомый фазовый сдвиг на этом q-бите:

$$\Lambda(V^l)\big(|\xi\rangle\otimes|\psi_n(q,k)\rangle\big) = \Lambda(e^{2\pi i(kl/q)})|\xi\rangle\otimes|\psi_n(q,k)\rangle.$$

(Классический) оператор $\Lambda(V^l)$ можно задать схемой линейного размера в стандартном базисе. Если k — нечётно, то выбором подходящего l можно представить оператор $\Lambda(e^{i\theta})$ с точностью $2\pi/q = O(\delta)$.

Вместо того, чтобы строить схему, порождающую $|\psi_n(q,k)\rangle$, будем брать смесь $|\psi_n(q,k)\rangle$ при разных k, измерять значение k и выбирать l, соответствующее этому измеренному значению. Опишем требуемые действия.

1. Создаём вектор

$$\sigma^z[1]H[1]|0^n\rangle = |\eta\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|q/2\rangle = \frac{1}{\sqrt{q/2}}\sum_{s=0}^{q/2-1}|\psi_n(q,2s+1)\rangle.$$

   Заметим, что $\sigma^z = \Lambda(i)^2 = K^2$ реализуется точно в стандартном базисе.

2. Произведём измерение k с вероятностью ошибки $\varepsilon \leqslant \delta^2$; оно может быть представлено оператором

$$W = \sum_{k=0}^{q-1}|\psi_n(q,k)\rangle\langle\psi_n(q,k)|\otimes W_k,$$

   где $W_k|0\rangle = \sqrt{1-\varepsilon}|k\rangle\otimes|\text{мусор}(k)\rangle + \sqrt{\varepsilon}|\zeta(k)\rangle$, а векторы $|\text{мусор}(k)\rangle$

и $|\zeta(k)\rangle$ — единичной длины. Заметим, что точность — квадратный корень из вероятности ошибки:

$$\big\| W_k|0\rangle - |k\rangle \otimes |\text{мусор}(k)\rangle \big\| = \sqrt{\left(1-\sqrt{1-\varepsilon}\right)^2 + \sqrt{\varepsilon}^2} = O(\delta).$$

3. Найдём такое $l(k)$, что $|kl(k)/q - \theta/(2\pi)| \leqslant 1/q$.
4. Применим $\Lambda(V^l)$ к рабочему n-битовому регистру, используя бит, в котором нужно сделать фазовый сдвиг, в качестве управляющего.
5. Обратим вычисления, сделанные на шагах 1–3.

Чтобы вероятность ошибки была меньше δ^2, нужны $O(\log(1/\delta))$ элементарных измерений для каждого из операторов V, V^2, ..., $V^{2^{n-1}}$. Поскольку $n = O(\log(1/\delta))$, общий размер схемы — $O(\log^3(1/\delta))$.

Из раздела 8

8.1. Пусть $\sum_z \left|\langle F(x), z|\, U\, |x, 0^{N-n}\rangle\right|^2 = 1 - \varepsilon_x$, и $\varepsilon_x \leqslant \varepsilon < 1/2$ для всех x. Нам нужно оценить величину

$$p(x) = \sum_{\substack{f_1,\dots,f_k\\ z_1,\dots,z_k}} \left|\langle f_1, z_1, \dots, f_k, z_k, F(x), 0^s|\, MU^{\otimes k}\, |(x, 0^{N-n})^k, 0^{m+s}\rangle\right|^2,$$

где M — оператор, реализующий применение функции MAJ к соответственным битам k регистров ответа исходной схемы и записывающий значение этой функции в регистр окончательного ответа. (Длина ответа равна m, а s дополнительных битов используются при вычислении MAJ).

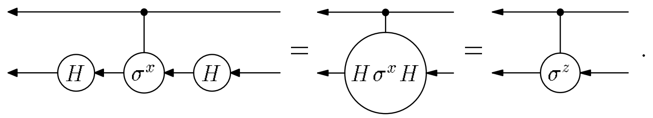
Если более половины регистров ответа исходной схемы содержат $F(x)$, то результатом применения M обязательно будет $F(x)$. Поэтому, аналогично (3.1) на с. 38, имеем

$$1 - p(x) \leqslant$$

$$\leqslant \sum_{\substack{S\subseteq\{1,\dots,k\},\\ |S|\leqslant k/2}} \; \sum_{\substack{f_1,\dots,f_k,\\ f_j = F(x) \Leftrightarrow j\in S}} \; \sum_{z_1,\dots,z_k} \left|\langle f_1, z_1, \dots, f_k, z_k| U^{\otimes k} |(x, 0^{N-n})^k\rangle\right|^2 =$$

$$= \sum_{\substack{S\subseteq\{1,\dots,k\},\\ |S|\leqslant k/2}} (1-\varepsilon_x)^{|S|} \varepsilon_x^{k-|S|} < \lambda^k, \quad \text{где } \lambda = 2\sqrt{(1-\varepsilon)\varepsilon}.$$

8.2. Поскольку $H^2 = I$, имеем цепочку равенств:



Оператор $\Lambda(\sigma^z)$ умножает $|1,1\rangle$ на -1, а остальные базисные векторы не меняет.

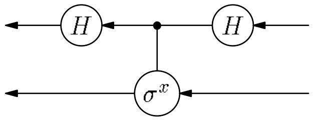
Если сделать замену базиса только в управляющем q-бите, как показано на рисунке ниже, то получится оператор, который является произведением отрицаний в обоих q-битах и «оператора диффузии» (на с. 70 мы обозначали его V). Действительно,

$$H[1]\Lambda(\sigma^x)[1,2]H[1]|a,b\rangle = H[1]\Lambda(\sigma^x)[1,2]\frac{1}{\sqrt{2}}\sum_c (-1)^{ac}|c,b\rangle =$$

$$= H[1]\frac{1}{\sqrt{2}}\sum_c (-1)^{ac}|c, b\oplus c\rangle = \frac{1}{2}\sum_{c,d}(-1)^{(a+d)c}|d, b\oplus c\rangle =$$

$$= \frac{1}{2}\sum_{c,b'}(-1)^{(a+d)(b+b')}|d, b'\rangle.$$

На рисунке слева показана схема вычисления такого оператора, а справа — его матрица в стандартном базисе:



$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & -1\\ 1 & 1 & -1 & 1\\ 1 & -1 & 1 & 1\\ -1 & 1 & 1 & 1 \end{pmatrix}.$$

8.3. BPP $\subseteq$ BQP. Классическое вероятностное вычисление можно представить обратимой схемой $(U_1, \dots, U_L)$, которая, наряду со входом x, использует случайную последовательность нулей и единиц $r \in \mathbb{B}^s$. (Кроме полезного ответа, схема может создавать мусор — это неважно). Заменим перестановки U_j на соответствующие унитарные операторы $\hat{U}_j$, а вместо случайного слова r приготовим состояние

$$|\psi\rangle = H^{\otimes s}|0^s\rangle = 2^{-s/2}\sum_{r\in\mathbb{B}^s}|r\rangle.$$

BQP $\subseteq$ PP. Пусть схема $(U_1, \dots, U_L)$ вычисляет предикат $F(x)$ с вероятностью ошибки $\leqslant 1/3$, общее число битов в схеме равно N, а $|x| = n$. Вероятность получения ответа 1 выражается через проектор $\Pi^{(1)} = |1\rangle\langle 1|$, применённый к первому q-биту:

$$\begin{aligned} p(x) &= \langle x, 0^{N-n}|U_1^\dagger U_2^\dagger \cdot \ldots \cdot U_L^\dagger\, \Pi^{(1)}[1]\, U_L U_{L-1} \cdot \ldots \cdot U_1|x, 0^{N-n}\rangle = \\ &= 2^{-h}\langle x, 0^{N-n}|V_L V_{L-1} \cdot \ldots \cdot V_{-L+1} V_{-L}|x, 0^{N-n}\rangle. \end{aligned} \qquad (*)$$

Здесь $V_L, \ldots, V_{-L}$ — перенумерованные операторы $U_1^\dagger, \ldots, \Pi^{(1)}[1], \ldots$ $\ldots, U_1$ с одним исключением: если $U_k = H[m]$ (или $U_k^\dagger = H[m]$), то соответствующий оператор V_j равен $\sqrt{2}H[m]$; количество элементов H в схеме обозначено через h.

Матричные элементы операторов $V_j \in \{\sqrt{2}H, K, K^\dagger, \Lambda(\sigma^x), \Lambda^2(\sigma^x), \Pi^{(1)}\}$ принадлежат множеству

$$M = \{0, +1, -1, +i, -i\}.$$

Перемножая матрицы, мы получаем сумму чисел из множества M. Поскольку интересующая нас величина $p(x)$ вещественная, мы можем ограничиться суммированием ± 1. Кратности слагаемых будут выражаться в виде

$$\#_a(x) = |\{w : C_a(x,w) = 1\}|, \quad a \in \{\pm 1\},$$

где предикаты $C_a(x,w) \in \mathrm{P}$ определены ниже. Получаем представление

$$p(x) = 2^{-h}(\#_1(x) - \#_{-1}(x)).$$

Дальнейшее приведение условия $(*)$ к виду из определения класса PP уже не будет использовать никакой квантовой специфики.

Теперь опишем предикаты $C_a(x,w)$ формально. Матричные элементы произведения $V_L \cdot \ldots \cdot V_{-L}$ можно выразить по формуле (5.1)

$$(V_L \cdot \ldots \cdot V_{-L})_{xy} = \sum_{x_{L-1}, \ldots, x_{-L+1}} (U_L)_{xx_{L-1}} \cdot \ldots \cdot (U_{-L})_{x_{-L+1}y}.$$

По определению, $C_a(u_{-L}, \ldots, u_L)$ равно 1, если и только если

$$u_{-L} = u_L = (x, 0^{N-n}), \qquad \prod_{j=-L+1}^{L} (V_j)_{u_j u_{j-1}} = a.$$

Легко видеть, что $C_a \in \mathrm{P}$: нужно представлять матричные элементы как степени i и суммировать показатели степеней по модулю 4.

Если $F(x) = 0$, то $p(x) \leqslant 1/3$; если $F(x) = 1$, то $p(x) \geqslant 2/3$. Итак, $F(x) = 1$ тогда и только тогда, когда

$$p(x) = 2^{-h}(\#_1(x) - \#_{-1}(x)) > \frac{1}{2}.$$

Это эквивалентно условию

$$\#_{-1}(x) + 2^{h-1} < \#_1(x).$$

Записанное неравенство почти соответствует определению класса PP: остаётся лишь проверить, что левая часть представима в виде $f(x) = |\{y : R(x,y) = 1\}|$, $R(\cdot,\cdot) \in \mathrm{P}$ (для правой части это уже доказано). Функции f такого вида образуют так называемый класс #P.

Покажем, что этот класс замкнут относительно сложения. Пусть $g(x) = |\{y : Q(x,y) = 1\}|$, $Q(\cdot,\cdot) \in \mathrm{P}$, тогда

$$f(x) + g(x) = |\{y : T(x, zy) = 1\}|,$$
$$\text{где } T(x, 0y) = R(x,y),\ T(x,1y) = Q(x,y).$$

Доказательство закончено.

PP $\subseteq$ PSPACE. Это очевидно. Заведём два счётчика: один для R_0, другой — для R_1. Перебираем все возможные значения y и увеличиваем значения счётчиков для R_k, если $R_k(x,y) = 1$. Потом сравниваем значения счётчиков.

8.4. Пункт а) следует из пункта б). Для б) приведём схему, которая даёт приближённое решение. Прежде всего, запишем рекуррентную формулу

$$|\psi_n(q)\rangle = \cos\vartheta |0\rangle \otimes |\psi_{n-1}(q')\rangle + \sin\vartheta |1\rangle \otimes |\psi_{n-1}(q'')\rangle, \qquad (*)$$

где

$$\begin{array}{llll} q' = 2^{n-1}, & q'' = q - 2^{n-1}, & \vartheta = \arccos\sqrt{q'/q}, & \text{если } q > 2^{n-1}; \\ q' = q, & q'' = 1, & \vartheta = 0, & \text{если } q \leqslant 2^{n-1}. \end{array}$$

Организуем рекурсивное вычисление, используя формулу $(*)$.

1. Вычисляем q', q'', ϑ/π, последнее представляем приближённо l двоичными цифрами. Запоминаем результаты вычисления в дополнительных q-битах.
2. Применяем к первому q-биту регистра $|0^n\rangle$, в котором нужно создать $|\psi_n(q)\rangle$, оператор
$$R(\vartheta) = \begin{pmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{pmatrix}.$$
3. В остальных $n-1$ битах создаём состояние, зависящее от значения первого бита: если он равен 0, то создаём состояние $|\psi_{n-1}(q')\rangle$, в противном случае создаём $|\psi_{n-1}(q'')\rangle$.
4. Проводим вычисление, обратное к шагу 1, чтобы очистить дополнительную память.

Оператор $R(\vartheta)$ реализуется приближённо. Пусть $\vartheta/\pi = \sum_{k=1}^{l} a_k 2^{-k}$. Тогда $R(\vartheta) \approx R(\pi/2^l)^{a_l} \cdot \ldots \cdot R(\pi/2)^{a_1}$ с точностью $O(2^{-l})$. Итак, приближённо оператор $R(\vartheta)$ представляется произведением операторов $\Lambda(R(\pi/2^k))$, где k-й разряд числа ϑ/π управляет применением оператора $R(\pi/2^k)$.

Общая точность такой схемы равна $\delta = O(n2^{-l})$; размер, выраженный через длину входа и точность, — $\mathrm{poly}(n + \log(1/\delta))$.

Пункт в). Приведём реализацию преобразования Фурье, найденную Копперсмитом и, независимо, Дойчем, в изложении П. Шора [39].

Занумеруем q-биты в убывающем порядке от $n-1$ до 0. Обозначим

$$H_j = H[j], \qquad S_{j,l} = \Lambda^2\big(e^{i\pi/2^{l-j}}\big)[j,l] \quad (j<l).$$

Тогда оператор

$$V_k = H_0 S_{0,1} S_{0,2} \cdot \ldots \cdot S_{0,n-2} S_{0,n-1} H_1 S_{1,2} \cdot \ldots \cdot S_{1,n-1} \cdot \ldots \\ \ldots \cdot H_{n-3} S_{n-3,n-2} S_{n-3,n-1} H_{n-2} S_{n-2,n-1} H_{n-1}$$

даёт почти то, что нужно: $U_k = RV_k$, где R — (классический) оператор, переписывающий двоичное слово в обратном порядке. Размер такой схемы $O(n^2)$.

Легко видеть, что модули матричных элементов V_k определяются количеством операторов H_j, так что они равны $1/\sqrt{k}$, как и требуется. Осталось проверить фазовые множители. Пусть

$$(V_k)_{(x_{n-1}\ldots x_0, y_{n-1}\ldots y_0)} = \frac{1}{\sqrt{k}} \exp(i \cdot 2\pi\varphi(x_{n-1},\ldots,x_0,y_{n-1},\ldots,y_0)).$$

Заметим, что каждый матричный элемент V_k есть произведение матричных элементов сомножителей. Применение H_j меняет фазу на π тогда и только тогда, когда $x_j = y_j = 1$; применение $S_{j,l}$ добавляет к фазе $\pi/2^{l-j}$ только в том случае, когда $x_j = y_l = 1$. Изменение фазы на 2π ни на что не влияет, поэтому вычислим φ по модулю 1.

$$\varphi(x_{n-1},\ldots,x_0,y_{n-1},\ldots,y_0) = \sum_{j=0}^{n-1} \frac{x_j y_j}{2} + \sum_{0\leqslant j<l<n} \frac{x_j y_l}{2^{l-j+1}} =$$

$$= \sum_{0\leqslant j\leqslant l<n} \frac{x_j y_l}{2^{l-j+1}} = \sum_{0\leqslant j+m<n} \frac{x_j y_{n-1-m}}{2^{n-j-m}} =$$

(здесь равенство по модулю 1)

$$= \sum_{j,m=0}^{n-1} \frac{x_j y_{n-1-m}}{2^{n-j-m}} = 2^{-n} \sum_{j=0}^{n-1} 2^j x_j \sum_{m=0}^{n-1} 2^m y_{n-1-m}.$$

После того, как мы перепишем слово y в обратном порядке, последнее выражение превращается в $xy/2^n$.

Из раздела 9

9.1. Пусть $\rho = \sum_k p_k |\xi_k\rangle\langle\xi_k|$. Проверим условия 1)–3) для ρ.

Условие 1): очевидно.

Условие 2): $\langle\eta|\,\rho\,|\eta\rangle = \sum_k p_k \langle\eta|\xi_k\rangle\langle\xi_k|\eta\rangle = \sum_k p_k \,|\langle\eta|\xi_k\rangle|^2 \geqslant 0$.

Условие 3): $\operatorname{Tr}\rho = \sum_k p_k \langle\xi_k|\xi_k\rangle = \sum_k p_k = 1$.

И наоборот, если ρ удовлетворяет 1)–3), то $\rho = \sum_k \lambda_k |\xi_k\rangle\langle\xi_k|$, где λ_k — собственные числа, а $\{|\xi_k\rangle\}$ — ортонормированный базис из собственных векторов.

9.2. Вектору $|\psi\rangle \in \mathcal{N}\otimes\mathcal{F}$ можно естественным образом сопоставить оператор $\Psi\colon \mathcal{F}^* \to \mathcal{N}$. Пусть p_j — ненулевые собственные числа оператора

$$\rho = \Psi\Psi^\dagger = \operatorname{Tr}_{\mathcal{F}}(|\psi\rangle\langle\psi|) \in \mathbf{L}(\mathcal{N})$$

(каждое собственное число учитывается столько раз, какова его кратность). Поскольку ρ является матрицей плотности, $0 < p_j \leqslant 1$. Положим $\lambda_j = \sqrt{p_j}$, а в качестве множества $\{|\xi_j\rangle\}$ возьмём любую ортонормированную систему собственных векторов, соответствующих собственным числам p_j.

Оператор Ψ можно представить в виде

$$\Psi = \sum_j \lambda_j |\xi_j\rangle\langle\nu_j|,$$

где $|\nu_j\rangle = \lambda_j^{-1}\Psi^\dagger|\xi_j\rangle \in \mathcal{F}^*$. Соответственно, $\langle\nu_j| \in \mathcal{F}^{**} = \mathcal{F}$. Переобозначив $\langle\nu_j|$ через $|\eta_j\rangle$, получаем искомое разложение Шмидта.

9.3. Условие $\operatorname{Tr}_{\mathcal{F}}(|\psi_1\rangle\langle\psi_1|) = \operatorname{Tr}_{\mathcal{F}}(|\psi_2\rangle\langle\psi_2|)$, как следует из решения предыдущей задачи, позволяет выбрать разложения Шмидта для $|\psi_1\rangle$ и $|\psi_2\rangle$ с одинаковыми λ_j и $|\xi_j\rangle$. Запишем эти разложения

$$|\psi_k\rangle = \sum_j \lambda_j |\xi_j\rangle \otimes |\eta_j^{(k)}\rangle, \qquad k = 1,\, 2.$$

Поскольку $\{|\eta_j^{(k)}\rangle\}$ — ортонормированные семейства, существует унитарный оператор U, такой что $U|\eta_j^{(1)}\rangle = |\eta_j^{(2)}\rangle$ для всех j. Тогда

$$(I_{\mathcal{N}}\otimes U)|\psi_1\rangle = \sum_j \lambda_j |\xi_j\rangle \otimes U|\eta_j^{(1)}\rangle = \sum_j \lambda_j |\xi_j\rangle \otimes |\eta_j^{(2)}\rangle = |\psi_2\rangle.$$

Из раздела 10

10.1. Утверждение задачи вытекает из следующей леммы, которая будет полезна и в дальнейшем.

Лемма. *Физически реализуемые преобразования матриц плотности имеют вид*

$$\rho \mapsto \sum_{m=1}^{s} A_m \rho A_m^\dagger, \qquad \sum_m A_m^\dagger A_m = I, \qquad (*)$$

который будем называть разложением в операторную сумму. А всякое разложение в операторную сумму можно представить в виде $\rho \mapsto$ $\mapsto \operatorname{Tr}_{\mathcal{F}}(V\rho V^\dagger)$, где V — изометрическое вложение.

Доказательство. План доказательства следующий:
а) докажем, что изометрическое вложение и взятие частичного следа имеют разложение в операторную сумму;
б) докажем, что операторные суммы замкнуты относительно композиции;
в) найдём представление произвольной операторной суммы в виде $\rho \mapsto \operatorname{Tr}_{\mathcal{F}}(V\rho V^\dagger)$.

а) Условие изометричности вложения записывается как $V^\dagger V = I$. Это означает, что изометрическое вложение представимо в виде операторной суммы из одного слагаемого.

Для частичного следа имеется следующее разложение в операторную сумму:

$$\operatorname{Tr}_{\mathcal{F}}(\rho) = \sum_m W_m \rho W_m^\dagger, \quad \text{где } W_m = I_{\mathcal{N}} \otimes \langle m|\colon \mathcal{N}\otimes\mathcal{F} \to \mathcal{N}. \qquad (**)$$

Заметим, что $W_m(|j,k\rangle) = \delta_{mk}|j\rangle$, а $W_m^\dagger(|j\rangle) = |j,m\rangle$.

б) Пусть $\sum\limits_m A_m\rho A_m^\dagger$, $\sum\limits_k B_k\rho B_k^\dagger$ — разложения двух преобразований в операторные суммы. Тогда их композиция также разлагается в операторную сумму:

$$\sum_k B_k\Big(\sum_m A_m\rho A_m^\dagger\Big)B_k^\dagger = \sum_{k,m}(B_kA_m)\rho(B_kA_m)^\dagger,$$

$$\sum_{k,m}(B_kA_m)^\dagger(B_kA_m) = \sum_m A_m^\dagger\Big(\sum_k B_k^\dagger B_k\Big)A_m = \sum_m A_m^\dagger A_m = I.$$

в) Пусть преобразование разложено в операторную сумму $(*)$, а $\mathcal{F}$ — s-мерное пространство, базисные векторы в котором обозначим $|m\rangle$. Отображение

$$V = \sum_m A_m \otimes |m\rangle\colon\ |\xi\rangle \mapsto \sum_m A_m|\xi\rangle\otimes|m\rangle$$

является изометрическим вложением, поскольку

$$V^\dagger V = \sum_{l,m}\big(A_l^\dagger\otimes\langle l|\big)\big(A_m\otimes|m\rangle\big) = \sum_{l,m} A_l^\dagger A_m|k\rangle\cdot\langle l|m\rangle = I.$$

Осталось заметить, что операторная сумма представляется в виде $\rho \mapsto \operatorname{Tr}_{\mathcal{F}}(V\rho V^\dagger)$. Действительно,

$$\operatorname{Tr}_{\mathcal{F}}(V\rho V^\dagger) = \sum_{m,l}\operatorname{Tr}_{\mathcal{F}}\Big(A_m\rho A_l^\dagger\otimes|m\rangle\langle l|\Big) = \sum_m A_m\rho A_m^\dagger. \qquad \square$$

10.2. Пусть $\rho\in\mathbf{L}(\mathcal{N}\otimes\mathcal{F})$, $U|j\rangle = |\xi_j\rangle$, $Y|k\rangle = |\eta_k\rangle$. Тогда

$$\begin{aligned}&\operatorname{Tr}_{\mathcal{F}}\big((U\otimes Y)\rho(U\otimes Y)^\dagger\big) =\\ &= \operatorname{Tr}_{\mathcal{F}}\Big((U\otimes Y)\sum_{j,k,j',k'}\rho_{jkj'k'}|j,k\rangle\langle j',k'|(U\otimes Y)^\dagger\Big) =\\ &= \operatorname{Tr}_{\mathcal{F}}\Big(\sum_{j,k,j',k'}\rho_{jkj'k'}|\xi_j\rangle\langle\xi_{j'}|\otimes|\eta_k\rangle\langle\eta_{k'}|\Big) = \sum_{jj'k}\rho_{jkj'k}|\xi_j\rangle\langle\xi_{j'}| =\\ &= U(\operatorname{Tr}_{\mathcal{F}}\rho)U^\dagger.\end{aligned}$$

10.3. Пусть $\sum_m A_m\rho A_m^\dagger$ — разложение в операторную сумму преобразования T (см. задачу 10.1). Тогда

$$T_{(j'j)(k'k)} = \langle j'|\,T\big(|j\rangle\langle k|\big)\,|k'\rangle = \sum_m\langle j'|A_m|j\rangle\cdot\langle k|A_m^\dagger|k'\rangle.$$

Такое представление позволяет легко проверить сформулированные в условии задачи свойства а)–в).

Свойство а): $$\sum_{k'}T_{(k'j)(k'k)} = \sum_{k'm}\langle k'|A_m|j\rangle\cdot\langle k|A_m^\dagger|k'\rangle = \sum_{k'm}\langle k|A_m^\dagger|k'\rangle\langle k'|A_m|j\rangle = \sum_m\langle k|A_m^\dagger A_m|j\rangle = \langle k|j\rangle.$$

Свойство б): $$T^*_{(j'j)(k'k)} = \sum_m\big(\langle j'|A_m|j\rangle\langle k|A_m^\dagger|k'\rangle\big)^* = \sum_m\langle j|A_m^\dagger|j'\rangle\langle k'|A_m|k\rangle = T_{(k'k)(j'j)}.$$

Свойство в): $$\sum_{j',j,k',k}T_{(j'j)(k'k)}|j',j\rangle\langle k',k| = \sum_m|\psi_m\rangle\langle\psi_m|,$$

где $$|\psi_m\rangle = \sum_{j',j}\langle j'|A_m|j\rangle\,|j',j\rangle.$$

И наоборот, всякий неотрицательный оператор можно представить в виде $T = \sum_m|\psi_m\rangle\langle\psi_m|$, где $|\psi_m\rangle$ — подходящим образом нормированные собственные векторы T, отвечающие положительным собственным числам. Обозначим

$$|\psi_m\rangle = \sum_{m,j',j}a_{mj'j}|j',j\rangle, \qquad A_m\colon|j\rangle\mapsto\sum_{j'}a_{mj'j}|j'\rangle,$$

тогда $T_{(j'j)(k'k)} = \sum\limits_m a_{mj'j}a^*_{mk'k}$. Из условия $\sum\limits_{k'}T_{(k'j)(k'k)} = \delta_{jk}$ следует

$$\langle k| \Big(\sum_m A_m^\dagger A_m \Big) |j\rangle \;=\; \langle k| \Big(\sum_{m,k'} a_{mk'k}^* a_{mk'j} \Big) |j\rangle \;=\; \sum_{k'} T_{(k'j)(k'k)} \;=\; \delta_{jk}.$$

Осталось проверить, что $T\big(|j\rangle\langle k|\big) = \sum_m A_m |j\rangle\langle k| A_m^\dagger$:

$$\sum_m A_m |j\rangle\langle k| A_m^\dagger = \sum_{j',k'} \sum_m a_{mj'j} a_{mk'k}^* |j'\rangle\langle k'| = \sum_{j'k'} T_{(j'j)(k'k)} |j'\rangle\langle k'| = \\ = T\big(|j\rangle\langle k|\big).$$

10.4. Свойства а) и б) эквивалентны свойствам а) и б) из предыдущей задачи.

Пусть есть физически реализуемое преобразование матриц плотности $T\colon \rho \mapsto \mathrm{Tr}_{\mathcal{F}}(V\rho V^\dagger)$. Тогда $T \otimes I_{\mathbf{L}(\mathcal{G})}\colon \rho \mapsto \mathrm{Tr}_{\mathcal{F}}\big((V \otimes I_{\mathcal{G}})\rho(V \otimes I_{\mathcal{G}})^\dagger\big)$ также является физически реализуемым преобразованием и поэтому обладает разложением в операторную сумму. Следовательно, $T \otimes I_{\mathbf{L}(\mathcal{G})}$ переводит неотрицательные операторы в неотрицательные.

Для доказательства утверждения в другую сторону выведем из свойства в) данной задачи свойство в) предыдущей задачи.

Чтобы доказать неотрицательность матрицы $T_{(j'j)(k'k)}$ по парам индексов, взятых в скобки, покажем, что она является матрицей оператора вида $(I \otimes T)|\psi\rangle\langle\psi|$, где $|\psi\rangle \in \mathcal{N} \otimes \mathcal{N}$ имеет вид $|\psi\rangle = \sum_j |j\rangle \otimes |j\rangle$. Действительно,

$$(I \otimes T)|\psi\rangle\langle\psi| = \sum_{j',j,k',k} T_{(j'j)(k'k)} |j\rangle\langle k| \otimes |j'\rangle\langle k'| = \sum_{j',j,k',k} T_{(j'j)(k'k)} |j'j\rangle\langle k'k|.$$

10.5. Воспользуемся результатом задачи 10.1. Представим $T\rho$ в виде $\mathrm{Tr}_{\mathcal{F}'}(V\rho V^\dagger)$. Возьмём $|\psi\rangle \in \mathcal{N}$. Поскольку состояние

$$\mathrm{Tr}_{\mathcal{F}\otimes\mathcal{F}'}\big(|V\psi\rangle\langle V\psi|\big) = \mathrm{Tr}_{\mathcal{F}}\big(\mathrm{Tr}_{\mathcal{F}'}\big(V|\psi\rangle\langle\psi|V^\dagger\big)\big) = \mathrm{Tr}_{\mathcal{F}}\big(T|\psi\rangle\langle\psi|\big) = |\psi\rangle\langle\psi|$$

чистое, $|V\psi\rangle = |\psi, \xi(\psi)\rangle$ (это следует из замечания, сделанного после формулировки задачи 9.2, см. с. 80). Из линейности V следует, что $|\xi(\psi)\rangle = |\xi\rangle$ не зависит от $|\psi\rangle$. Поэтому $TX = X \otimes \gamma$, где $\gamma = \mathrm{Tr}_{\mathcal{F}'}(|\xi\rangle\langle\xi|)$.

10.6. Будем считать, что сразу же после измерения измеряемые q-биты выбрасываются в «мусорную корзину». Это соответствует преобразованию двух квантовых битов в классические:

$$T\colon \rho \mapsto \sum_{a,b} \langle \xi_{ab}|\rho|\xi_{ab}\rangle (a,b).$$

Чтобы реализовать преобразование T, нужно сначала подействовать унитарным оператором

$$H[1]\Lambda(\sigma^x)[1,2]\colon\; |\xi_{ab}\rangle \mapsto |b,a\rangle,$$

а затем произвести измерение в базисе $\{|0\rangle, |1\rangle\}$; после этого q-биты выбрасываются.

Без ограничения общности, первый q-бит находится в чистом состоянии $|\psi\rangle = z_0|0\rangle + z_1|1\rangle$. (Если мы построим восстанавливающую процедуру для чистых состояний, то она по линейности будет продолжаться на смешанные). На третий q-бит измерение не действует, поэтому можно записать

$$(T \otimes I_{\mathbf{L}(\mathcal{B})}) \Big(|\psi\rangle\langle\psi| \otimes |\xi_{00}\rangle\langle\xi_{00}| \Big) = \sum_{a,b} (a, b, |\psi_{ab}\rangle\langle\psi_{ab}|),$$

где $|\psi_{ab}\rangle = \big(\langle\xi_{ab}| \otimes I_{\mathcal{B}}\big)\big(|\psi\rangle \otimes |\xi_{00}\rangle\big)$. Здесь $\langle\xi_{ab}|$ рассматривается как оператор $\mathcal{B}^{\otimes 2} \to \mathbb{C}$, поэтому $\langle\xi_{ab}| \otimes I_{\mathcal{B}}\colon \mathcal{B}^{\otimes 3} \to \mathcal{B}$. Заметим, что квадрат нормы вектора $|\psi_{ab}\rangle$ равен вероятности получения пары (a, b).

Теперь запишем явное выражение для $|\psi_{ab}\rangle$:

$$|\psi_{ab}\rangle = \frac{1}{\sqrt{2}} \sum_d \big(\langle\xi_{ab}| \otimes I_{\mathcal{B}}\big)\big(|\psi\rangle \otimes |d,d\rangle\big) = \frac{1}{\sqrt{2}} \sum_d \langle\xi_{ab}|\psi, d\rangle\, |d\rangle = \\ = \frac{1}{\sqrt{2}} \sum_{c,d} z_c \langle\xi_{ab}|c, d\rangle\, |d\rangle = \frac{1}{2} \sum_{c,d} (-1)^{bc} \delta_{c\oplus a,d} z_c |d\rangle = \frac{1}{2} \sum_c (-1)^{bc} z_c |a \oplus c\rangle.$$

Из этого выражения сразу следует, что

$$(\sigma^z)^b\, (\sigma^x)^a\, |\psi_{ab}\rangle = \frac{1}{2} |\psi\rangle.$$

Так что состояние $|\psi\rangle$ в третьем q-бите получится применением операторов σ^x и σ^z с классическим управлением: управляющими параметрами являются измеренные значения a и b.

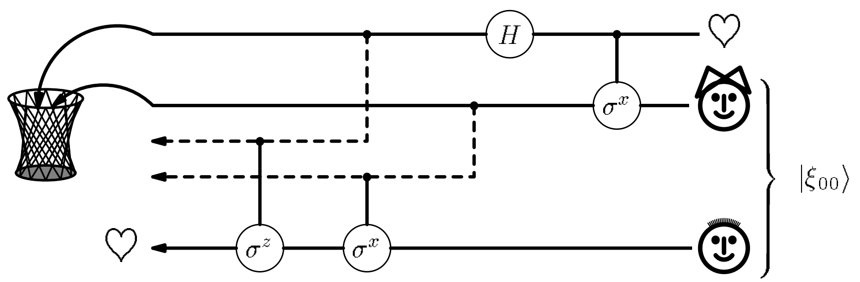


Рис. 17. *Схема квантовой телепортации*

Схема квантовой телепортации изображена на рис. 17. Значком ♡ обозначен первый бит,  — второй (q-бит Алисы),  — третий (q-бит Боба). Когда Алиса хочет передать q-бит ♡ Бобу, она совершает измерения над ним и своим q-битом (дальше эти q-биты не используются, и она выбрасывает их в мусорную корзину). Результаты измерений она сообщает Бобу по классическому каналу связи (телефону). Боб, используя сообщение Алисы, превращает свой q-бит в q-бит $\heartsuit$.

Из раздела 11

11.1. По определению квантовой вероятности имеем

$$\begin{aligned}
&\mathbf{P}\Big(W\big(|0\rangle\langle 0|\otimes\rho\big)W^\dagger,\, \mathbb{C}(|k\rangle)\otimes\mathcal{N}\Big) = \\
&= \operatorname{Tr}\Big(\Pi_{\mathbb{C}(|k\rangle)\otimes\mathcal{N}} W\big(|0\rangle\langle 0|\otimes\rho\big)W^\dagger\Big) = \\
&= \sum_j \operatorname{Tr}\Big(\big(|k\rangle\langle k|\big)R_j\big(|0\rangle\langle 0|\big)R_j^\dagger \otimes \Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}\Big) = \\
&= \sum_j \operatorname{Tr}\Big(\big(|k\rangle\langle k|\big)R_j\big(|0\rangle\langle 0|\big)R_j^\dagger\Big)\operatorname{Tr}\big(\Pi_{\mathcal{L}_j}\rho\Pi_{\mathcal{L}_j}\big) = \\
&= \sum_j |\langle k|R_j|0\rangle|^2 \mathbf{P}(\rho,\mathcal{L}_j) \;=\; \sum_j \mathbf{P}(k|j)\mathbf{P}(\rho,\mathcal{L}_j).
\end{aligned}$$

11.2. Если $W = \sum_{k=1}^t \Pi_{\mathcal{L}_k}\otimes V_k$, то $W^{-1} = \sum_{k=1}^t \Pi_{\mathcal{L}_k}\otimes V_k^{-1}$. Поэтому искомая квантовая схема имеет вид $W^{-1}YW$, где оператор Y копирует в дополнительный регистр «полезный результат»:

$$Y : |y,z,v\rangle \mapsto |y,z,v\oplus y\rangle.$$

Оценка точности делается так же, как в задаче 7.11.

Из раздела 12

12.1. Интересующую нас вероятность обозначим через $p(X,l)$. Если $h_1,\ldots,h_l$ не порождают всю группу X, то они содержатся в некоторой максимальной собственной подгруппе $Y\subset X$. Для каждого Y вероятность такого события не превосходит 2^{-l}, поскольку $|Y|\leqslant |X|/2$. Таким образом, мы имеем оценку

$$p(X,l)\geqslant 1-K(X)\cdot 2^{-l},$$

где $K(X)$ — число максимальных собственных подгрупп в группе X.

Подгруппы абелевой группы X находятся во взаимнооднозначном соответствии с подгруппами группы характеров X^*, при этом максимальным собственным подгруппам отвечают минимальные ненулевые подгруппы. Каждая такая подгруппа порождается одним элементом, поэтому $K(X)\leqslant |X^*| = |X|$.

12.2. Построим классический оператор $V_b\in\mathbf{L}(\mathcal{B}\otimes\mathcal{B}^{\otimes n})$ (базисные векторы в $\mathcal{B}^{\otimes n}$ занумерованы от 0 до 2^n-1), такой что

$$V_b|0,0\rangle = |0,1\rangle,\qquad V_b|1,0\rangle = |1,b\rangle.$$

Тогда схема $V^{-1}[0,B]\,U[B,A]\,V[0,B]$ реализует оператор $\Lambda(U_b)[0,A]$ в расширенном смысле.

12.3. Обозначим образ вектора $|x\rangle$ при преобразовании Фурье через $|\psi_n(k,x)\rangle$. В задаче 8.4 мы научились строить вектор $|\psi_n(k,0)\rangle$. Как уже отмечалось в решении задачи 7.11, $|\psi_n(k,x)\rangle$ — собственный вектор классического оператора $V\colon |j\rangle\mapsto|(j-1)\bmod k\rangle$, собственное число которого равно $\exp(2\pi i x/k)$.

Используя эти соображения и результат задачи 11.2, построим следующую схему для квантового преобразования Фурье.

1. В дополнительном, изначально нулевом, регистре построим вектор $|\psi_n(k,0)\rangle$. Общее состояние получается
$$\frac{1}{\sqrt{k}}\sum_y |x,y\rangle.$$
2. Сделаем фазовый сдвиг на $\exp(2\pi i x y/k)$. Теперь получаем состояние $|x\rangle\otimes|\psi_n(k,x)\rangle$.
3. Измеряя обратимым образом (см. задачу 11.2) собственное число оператора V, действующего на второй регистр, прибавляем результат измерения x к первому регистру. (Здесь имеется в виду побитовое сложение по модулю 2). В первом регистре получается $|0\rangle$, а во втором — требуемый результат $|\psi_n(k,x)\rangle$.

Из раздела 14

14.1. Оператор A можно представить в виде

$$A=\sum_j \lambda_j|\xi_j\rangle\langle\eta_j|,\qquad \lambda_j>0,\qquad \langle\xi_j|\xi_k\rangle=\langle\eta_j|\eta_k\rangle=\delta_{jk}.$$

Здесь λ_j^2 — ненулевые собственные числа оператора $AA^\dagger$, $|\xi_j\rangle$ — его собственные векторы, а $|\eta_j\rangle = \lambda_j^{-1}A^\dagger|\xi_j\rangle$. Тогда $\|A\|_{\mathrm{tr}}=\sum_j\lambda_j$.

Для любого оператора X

$$|\operatorname{Tr}AX|\leqslant\sum_j\lambda_j\big|\operatorname{Tr}|\xi_j\rangle\langle\eta_j|X\big|\leqslant\sum_j\lambda_j\|X\| = \|A\|_{\mathrm{tr}}\|X\|.$$

С другой стороны, если взять $X=\sum_j|\eta_j\rangle\langle\xi_j|$, то $\|X\|\leqslant 1$, а $\operatorname{Tr}AX = \|A\|_{\mathrm{tr}}$.

Из доказанного представления для $\|\cdot\|_{\mathrm{tr}}$ легко следует неравенство треугольника, а положительность и однородность $\|\cdot\|_{\mathrm{tr}}$ очевидны.

14.2. Свойство а):

$$\|AB\|_{\mathrm{tr}} = \sup_{X\neq 0} \frac{|\operatorname{Tr} ABX|}{\|X\|} \leqslant \sup_{X\neq 0} \frac{\|A\|_{\mathrm{tr}}\|BX\|}{\|X\|} \leqslant \|A\|_{\mathrm{tr}}\|B\|.$$

Аналогично доказывается и свойство б) (воспользуйтесь равенством $\operatorname{Tr} ABC = \operatorname{Tr} CAB$).

Свойство в):

$$|\operatorname{Tr}(A)| = \frac{|\operatorname{Tr}(AI)|}{\|I\|} \leqslant \|A\|_{\mathrm{tr}}.$$

Свойство г): для любого $A \in \mathbf{L}(\mathcal{N}\otimes\mathcal{M})$

$$\|\operatorname{Tr}_{\mathcal{M}} A\|_{\mathrm{tr}} = \sup_{X\neq 0} \frac{\big|\operatorname{Tr}\big((\operatorname{Tr}_{\mathcal{M}} A)X\big)\big|}{\|X\|} = \sup_{X\neq 0} \frac{\big|\operatorname{Tr}\big(A(X\otimes I_{\mathcal{M}})\big)\big|}{\|X\otimes I_{\mathcal{M}}\|} \leqslant \|A\|_{\mathrm{tr}}.$$

Свойство д):

$$\|A\otimes B\|_{\mathrm{tr}} = \operatorname{Tr}\sqrt{(A\otimes B)^{\dagger}(A\otimes B)} = \operatorname{Tr}\left(\sqrt{A^{\dagger}A}\otimes\sqrt{B^{\dagger}B}\right) = \|A\|_{\mathrm{tr}}\|B\|_{\mathrm{tr}}.$$

14.3. Пусть $\mathcal{F}$ — пространство состояний q-битов из A, а $\mathcal{N}$ — пространство состояний остальных q-битов. Обозначим

$$\mathcal{D} = I_{\mathcal{N}}\otimes\mathcal{F}^{*}\colon\ \mathcal{N}\otimes\mathcal{F}\to\mathcal{N}.$$

Если $X, Y \in \mathcal{D}$, то $Y^{\dagger}X \in I_{\mathcal{N}}\otimes\mathbf{L}(\mathcal{F}) = \mathcal{E}(A)$. Следовательно, код $\mathcal{M}$ исправляет ошибки из $\mathcal{D}$.

Преобразование $T\colon \rho\mapsto \operatorname{Tr}_{\mathcal{F}}\rho$ может быть разложено в операторную сумму $(**)$, см. решение задачи 10.1. Операторы W_m из этого разложения принадлежат пространству $\mathcal{D}$, поэтому $T\in\mathcal{D}\cdot\mathcal{D}^{\dagger}$. Осталось воспользоваться теоремой 14.2.

14.4. Пространство F, соответствующее искомому коду, порождается строками таблицы

1	0	1	0	0	1	0	1	0	0
1	0	0	1	1	0	0	0	0	1
0	1	0	0	0	1	1	0	0	1
0	1	0	1	0	0	0	1	1	0

Можно проверить, что $\omega(f_j, f_k) = 0$ для любых двух строк f_j, f_k. Заметим, что столбцы в таблице разбиты на пары. Если взять любые две пары, то соответствующие 4 столбца линейно независимы. Следовательно, из строк всегда можно составить линейную комбинацию, которая в двух заданных парах позиций содержит заданные числа. Поэтому условия $\omega(f_j, g) = 0$ $(j = 1,2,3,4)$ при $|g| \leqslant 2$ могут выполняться только для $g = 0$.

14.5. (См. [22, 33].) Допустим, что $\mathcal{M}$ — код типа $(4,1)$, исправляющий одну ошибку. Тогда он должен обнаруживать по крайней мере две

ошибки, в частности, ошибки в q-битах $[1,2]$, а также в q-битах $[3,4]$. Это означает, что произвольное состояние $\rho\in\mathbf{L}(\mathcal{M})$ можно восстановить как по первым, так и по последним двум q-битам (см. задачу 14.3). Покажем, что это невозможно.

Пусть $\mathcal{N}_1$ — пространство состояний q-битов $[1,2]$, а $\mathcal{N}_2$ — пространство состояний q-битов $[3,4]$, тогда $\mathcal{M}$ — это подпространство в $\mathcal{N}_1\otimes\mathcal{N}_2$. Вложение $\mathcal{M}\to\mathcal{N}_1\otimes\mathcal{N}_2$ обозначим через V (это изометрический оператор). Пусть также $T_1\colon \rho\mapsto \operatorname{Tr}_{\mathcal{N}_2}\rho$ и $T_2\colon \rho\mapsto\operatorname{Tr}_{\mathcal{N}_1}\rho$ — преобразования ошибок, а $P_1\colon \mathcal{N}_1\to\mathcal{M}$ и $P_2\colon\mathcal{N}_2\to\mathcal{M}$ — соответствующие исправляющие преобразования. Тогда преобразование $P = (P_1\otimes P_2)(V\cdot V^{\dagger})\colon \mathcal{M}\to\mathcal{M}\otimes\mathcal{M}$ обладает следующим свойством: для любого $\rho\in\mathcal{M}$

$$\operatorname{Tr}_{\mathcal{N}_2} P\rho = \operatorname{Tr}_{\mathcal{N}_2}\big((P_1\otimes P_2)(V\rho V^{\dagger})\big) = P_1T_1(V\rho V^{\dagger}) = \rho,$$
$$\operatorname{Tr}_{\mathcal{N}_1} P\rho = \operatorname{Tr}_{\mathcal{N}_1}\big((P_1\otimes P_2)(V\rho V^{\dagger})\big) = P_2T_2(V\rho V^{\dagger}) = \rho.$$

Согласно задаче 10.5 первое тождество означает, что $P\rho = \rho\otimes\gamma_2$, где γ_2 не зависит от ρ. Из второго тождества следует, что $P\rho = \gamma_1\otimes\rho$. Получили противоречие: $\gamma_1\otimes\rho = \rho\otimes\gamma_2$, где ρ —любое.

14.6. Опишем кратко идею решения этой задачи.

Достаточно рассмотреть одно из двух прямых слагаемых торического кода. Компонента синдрома равна 1 для такого узла решётки, в звезду которого входит нечётное число рёбер с ненулевыми весами в 1-цепи, соответствующей вектору ошибки $g^{(z)}$.

Поэтому получаем такую задачу. Задано некоторое множество D узлов решётки. Из всех 1-цепей C, граница которых совпадает с D, нужно выбрать ту, в которой наименьшее число рёбер ненулевого веса. Нетрудно сообразить, что такая 1-цепь распадается в объединение путей, соединяющих узлы из множества D (любые два различных пути не имеют общих рёбер), причём эти пути можно считать кратчайшими. Так что задача определения ошибки по синдрому сводится к *задаче о взвешенном паросочетании*: дан граф G (в нашем случае полный), каждому его ребру приписан вес (в нашем случае — расстояние между узлами по решётке), нужно найти паросочетание, на котором достигается минимум суммы весов по рёбрам, входящим в паросочетание.

Для задачи о взвешенном паросочетании известны полиномиальные алгоритмы (см., например, [11, гл. 11], где описан алгоритм, основанный на идеях линейного программирования).

Литература

[1] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

[2] Виноградов И. М. Основы теории чисел. Изд. 8-е, испр. М.: Наука, 1972.

[3] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

[4] Китаев А. Ю., Квантовые вычисления: алгоритмы и исправление ошибок // УМН, №6, 1997.

[5] Клини С. Математическая логика. М.: Мир, 1973.

[6] Клини С. Введение в метаматематику. М.: ИЛ, 1957.

[7] Кнут Д. Искусство программирования на ЭВМ. В 3 т. М.: Мир, 1977.

[8] Кострикин А. И., Манин Ю. И. Линейная алгебра и геометрия. М.: Наука, 1986.

[9] Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А. Теория кодов, исправляющих ошибки. М.: Связь, 1979.

[10] Мальцев А. И. Алгоритмы и рекурсивные функции. М.: Наука, 1965.

[11] Пападимитриу Х., Стаглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1985.

[12] Прасолов В. В. Задачи и теоремы линейной алгебры. М.: Наука. Физматлит, 1996.

[13] Роджерс Х. Теория рекурсивных функций и эффективная вычислимость. М.: Мир, 1972.

[14] Схрейвер А. Теория линейного и целочисленного программирования. В 2 т. М.: Мир, 1991.

[15] Шафаревич И. Р. Основные понятия алгебры // Алгебра-1. Итоги науки и техники. Сер. Совр. проблемы математики. Фундаментальные направления. Т. 11. М.: ВИНИТИ, 1986.

[16] Шенфилд Дж. Р. Математическая логика. М.: Наука, 1975.

[17] Шенфилд Дж. Р. Степени неразрешимости. М.: Наука, 1977.

[18] Эрдёш П., Спенсер Дж. Вероятностные методы в комбинаторике. М.: Мир, 1976.

[19] Aharonov D., Ben-Or M. Fault Tolerant Quantum Computation with Constant Error. LANL e-print quant-ph/9611025 (extended version: Fault-Tolerant Quantum Computation With Constant Error Rate. LANL e-print quant-ph/9906129), `http://xxx.lanl.gov`

[20] Aharonov D., Kitaev A., Nisan N. Quantum Circuits with Mixed States // STOC'29, 1997. (electronic version: LANL e-print quant-ph/9806029, `http://xxx.lanl.gov`)

[21] Bennett C., Brassard G., Crépeau C., Jozsa R., Peres A., Wootters W. Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channel // Phys. Rev. Lett. Vol. 70, 1993. P. 1895–1899.

[22] Bennett C., DiVincenzo D., Smolin J., Wootters W. Mixed state entanglement and quantum error correction // Phys. Rev. Vol. A 54, 1996. P. 3824–3851. (electronic version: LANL e-print quant-ph/9604024, `http://xxx.lanl.gov`)

[23] Boneh D., Lipton R. Quantum Cryptoanalysis of Hidden Linear Functions // Proceedings of Advances in Cryptology — CRYPTO-95, Lecture Notes in Computer Science. Vol. 963. Springer-Verlag, 1995. P. 424–437.

[24] Boppana R., Sipser M. The Complexity of Finite Functions // Handbook of Theoretical Computer Science. Volume A, Algorithms and Complexity, Ch. 14. J. van Leeuwen, ed., Amsterdam et al.: Elsevier / Cambridge, MA: MIT Press, 1990. P. 757–804.

[25] Calderbank A. R., Shor P. W. Good quantum error-correcting codes exist // Phys. Rev. Vol. A 54, 1996. P. 1098-1106. (electronic version: LANL e-print quant-ph/9512032, `http://xxx.lanl.gov`)

[26] Calderbank A. R., Rains E. M, Shor P. W., Sloane N. J. A. Quantum Error Correction and Orthogonal Geometry // Phys. Rev. Lett. Vol. 78, 1997. P. 405-408 (electronic version: LANL e-print quant-ph/9605005, `http://xxx.lanl.gov`)

[27] Deutsch D. Quantum theory, the Church-Turing principle and the universal quantum computer // Proc. R. Soc. Lond. Vol. A 400, 1985. P. 97.

[28] Deutsch D. Quantum computational networks // Proc. Roy. Soc. Lond. Vol. A 425,1989. P. 73.

[29] Feynman R. P. Quantum mechanical computers // Optics News, February 1985. Vol. 11. P. 11.

[30] Fortnow L., Sipser M. Are there interactive protocols for Co-NP-languages // Inf. Proc. Letters. Vol. 28, 1988. P. 249–251.

[31] Grover L. A fast quantum mechanical algorithm for database search // STOC'28, 1996. P. 212–219.

[32] Kitaev A. Yu. Fault-tolerant quantum computation by anyons. LANL e-print quant-ph/9707021, `http://xxx.lanl.gov`

[33] Knill E., Laflamme R. A theory of quantum error-correcting codes. LANL e-print quant-ph/9604034, `http://xxx.lanl.gov`

[34] Knill E., Laflamme R., Zurek W. Threshold Accuracy for Quantum Computation. LANL e-print quant-ph/9610011, `http://xxx.lanl.gov`

[35] Lautemann C. BPP and the polynomial hierarchy // Inf. Proc. Lett. Vol. 17, no. 4, 1983. P. 215–217.

[36] Shamir A. IP=PSPACE // Journal of the ACM, Vol. 39, no 4, 1992. P. 869–877.

[37] Shen A. IP=PSPACE: simplified proof // Journal of the ACM, Vol. 39, no 4, 1992. P. 878–880.

[38] Shor P. W. Algorithms for Quantum Computation: Discrete log and Factoring // FOCS'35, 1994. P. 124.

[39] Shor P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer // SIAM Journal of Computing. Vol. 26, 1997. P. 1484. (electronic version: LANL e-print quant-ph/9508027, `http://xxx.lanl.gov`)

[40] Shor P. W. Scheme for reducing decoherence in quantum memory // Phys. Rev. Vol. A 52, 1995. P. 2493–2496.

[41] Shor P. W. Fault-tolerant quantum computation // FOCS'37, 1996. P. 56–65. (electronic version: LANL e-print quant-ph/9605011, `http://xxx.lanl.gov`)

[42] Simon D. On the Power of Quantum Computation // FOCS'35, 1994. P. 116–123.

[43] Sipser M. Introduction to the Theory of Computation. Boston: PWS Publishing Company, 1997.

[44] Steane A. M. Multiple particle interference and quantum error correction // Proc. Roy. Soc. Lond. Vol. A 452, 1996. P. 2551. (electronic version: LANL e-print quant-ph/9601029, `http://xxx.lanl.gov`)

[45] Watrous J. PSPACE has 2-round quantum interactive proof system. LANL e-print cs.CC/9901015, `http://xxx.lanl.gov`

[46] Yamakami T., Yao A. C. $\mathrm{NQP_C} = \text{co-}\mathrm{C_=}P$. LANL e-print quant-ph/9812032, `http://xxx.lanl.gov`

[47] Yao A. C.-C. Quantum circuit complexity // FOCS'34, 1993. P. 352.

[48] Zalka C. Grover's quantum searching algorithm is optimal. LANL e-print quant-ph/9711070, `http://xxx.lanl.gov`

STOC = Proceedings of the Annual. ACM Symposium on Theory of Computing. New York: ACM Press.

FOCS = Proceedings of the Annual Symposium on the Foundations of Computer Science. Los Alamitos, CA: IEEE Computer Society Press.

Предметный указатель