

---

Correction des TD de théorie  
d'information

Année 2008-09

Version 1.1

---

Université de Montpellier  
Place Eugène Bataillon  
34095 Montpellier Cedex 5

RODOLPHE GIROUDEAU  
161, RUE ADA  
34392 MONTPELLIER CEDEX 5  
TEL : 04-67-41-85-40  
MAIL : RGIROU@LIRMM.FR

## 1 Mesure de l'information

### Exercice 1 – Entropie et codage

Une séquence expérimentale d'ADN est définie comme une suite de symboles certains ou incertains. Un symbole certain est un élément de l'alphabet :  $B = \{A, T, G, C\}$ . Un symbole peut être incertain d'ordre 2 ou incertain d'ordre 3. Un symbole incertain d'ordre 2 est un élément de l'ensemble des arrangements de 2 éléments de  $B$ ; il y a donc un ordre interprété comme le fait que la première lettre de l'arrangement est plus probable que la seconde. Par exemple :  $TA$  est un symbole incertain d'ordre 2;  $AT$  en est un autre. Un symbole incertain d'ordre 3 est un élément de l'ensemble des combinaisons de 3 éléments de  $B$ ; il n'y a donc pas d'ordre. Par exemple  $TGA$  ou  $AGT$  ou  $TAG$  etc... sont des notations d'un même symbole incertain d'ordre 3.

Une série d'expériences a montré les résultats suivants sur un ensemble de séquences :

1. les symboles certains sont équiprobables entre eux
2. tous les symboles incertains (ordre 2 et 3 confondus) sont aussi équiprobables entre eux
3. la probabilité d'un symbole certain dans une séquence est la même que celle d'un symbole incertain.
1. Calculer l'information d'un symbole certain et l'information d'un symbole incertain dans une séquence expérimentale.
2. Calculer l'entropie d'une séquence expérimentale.
3. Trouver un code optimal pour chaque symbole d'une séquence expérimentale.
4. Un génome humain est une suite de 3,5 milliards de symboles. En supposant cette suite composée de symboles indépendants, et en se plaçant dans le même cadre expérimental, que doit-on prévoir comme capacité de stockage sur un support exempt d'erreurs pour archiver la séquence expérimentale d'un génome?

### Correction exercice 1

1.  $\mathbb{P}(\text{symbole certain}) = 1/8$ . Symboles incertains d'ordre 2 :  $F_1 = \{AT, AG, AC, TA, TG, TC, GA, GT, GC, CA\}$  avec  $\text{Card}(F_1) = 12$ . Symboles incertains d'ordre 3 :  $F_2 = \{\bar{A}, \bar{T}, \bar{G}, \bar{C}\}$  avec  $\text{Card}(F_2) = 16$ .  $\mathbb{P}(F) = 1/32$ .  
Nous avons donc  $I(\text{symbole certain}, x \in B) = -\log_2(1/8) = 3$ .  $I(x \in F) = \log_2 2^5 = 5$ .

$$2. H(\text{séquence expérimentale}) = -\sum_{x \in B \cup F} \mathbb{P}(x) \log_2(1/\mathbb{P}(x)) = -\sum_{x \in B} \mathbb{P}(x) \log_2(1/\mathbb{P}(x)) - \sum_{x \in F} \mathbb{P}(x) \log_2(1/\mathbb{P}(x)) = 4.$$

$A$	$T$	$C$	$G$	$AT$	$AG$	$AC$	$TA$	$TG$	$TC$
000	001	010	011	1000	1001	1010	1011	10100	10101
$GA$	$GT$	$GC$	$CA$	$CT$	$CG$	$A$	$T$	$G$	$C$
10110	10111	11000	11011	11010	11011	11100	11101	11110	11111

$$4. 1,75 \text{ Go}, 3,5 \times 4 \text{ bits}$$

### Fin correction exercice 1

### Exercice 2 – Entropie et codage suite

1. Calculer l'entropie  $H_x$  d'une source binaire  $X = \{0, 1\}$  telle que  $\mathbb{P}(x = 0) = p$  et  $\mathbb{P}(x = 1) = 1 - p$ .
2. Étudier et interpréter  $H_x(p)$ .

### Correction exercice 2

$$1. H_X(p) = -p \log_2 p - (1 - p) \log_2(1 - p)$$

### Fin correction exercice 2

## 1.1 Capacité d'un canal

### Exercice 3 – Canal dissymétrique

On considère un canal binaire, dissymétrique, stationnaire et sans mémoire. La transmission d'un bit égal à 0 se fait sans erreur. La transmission d'un bit égal à 1 se fait avec une erreur aléatoire de 50%.

1. Calculer l'équivoque et la transformation pour une source équiprobable.
2. Calculer la capacité de ce canal.

### Correction exercice 3

1.  $H(X|Y) = \frac{1+p}{2} \log_2(1+p) - \frac{1-p}{2} \log_2(1-p) - p \log_2 p - p$
2.  $I(X; Y) = p - \frac{1-p}{2} \log_2(1-p) - \frac{1+p}{2} \log_2(1+p)$
3.  $C = \log_2 5 - 2$

**Fin correction exercice 3**

**Exercice 4 – Canal à « effacement »**

On considère un canal binaire symétrique sans erreur, mais avec pertes détectées par un mécanisme approprié. L'alphabet de réception comporte donc 3 éléments : 0, 1 et  $\epsilon$  qui désigne le bit détecté « perdu ». Soit  $p$  la probabilité de perte d'un bit.

1. Calculer l'équivoque et la transinformation pour une source équiprobable.
2. Calculer la capacité de ce canal pour  $p = 1/2$ .

**Correction exercice 4**

1.  $H(X|Y) = I(X; Y) = \frac{1-p}{2} \log_2\left(\frac{1}{1-p}\right) + \frac{p}{2} \log_2 \frac{1}{p}$ .
2.  $C = 1/2$

**Fin correction exercice 4**

**Exercice 5 – Transmission bruitée**

On s'intéresse à un nouveau standard de Télévision *TBD* : Très Basse Définition pour des applications militaires. Une image est composée de pixels. Chaque pixel est défini par de l'une des trois couleurs :  $R$ ,  $V$  et  $B$ . On a mesuré sur une grande série d'images la répartition en probabilités des trois couleurs :

$$R : 1/4 \quad V : 1/2 \quad B : 1/4$$

1. Première partie
  - (a) Quelle est la quantité d'information associée à chacune des 3 couleurs.
  - (b) Quelle est l'entropie de la source ?
  - (c) Donner le codage optimal d'un pixel.
2. La transmission a lieu dans un environnement très bruité. La matrice de transmission est ainsi définie : soit  $X$  le symbole émis et  $Y$  le symbole reçu, appartenant chacun à l'alphabet  $\{R, V, B\}$ .  $\mathbb{P}(Y|X) = 1/2$  si  $Y = X$ ,  $1/4$  sinon.
  - (a) Quelle est la matrice de réception  $\mathbb{P}(X|Y)$  donnant pour chaque symbole reçu la probabilité du symbole émis ?
  - (b) Calculer l'entropie de la source de chrominance après réception :  $H(X|Y)$ .
  - (c) Calculer la quantité d'information transmise  $I(X; Y)$ .

On donne  $\log_2 3 = 1,58$ ,  $\log_2 5 = 2,32$  et  $\log_2 7 = 2,81$

**Correction exercice 5**

1.  $I(R) = \log_2(1/(1/4)) = \log_2 4 = 2 = I(B)$  et  $I(V) = \log_2(1/(1/2)) = \log_2 2 = 1$ . Donc  $H(x) = 1,5$  bits.  $R = 00$   $B = 01$  et  $V = 1$ .
2.  $\mathbb{P}(Y|X) = 1/2$  si  $Y = X$

$X/Y$	$R$	$V$	$B$
$R$	1/2	1/4	1/4
$V$	1/4	1/2	1/4
$B$	1/4	1/4	1/2

	$R$	$V$	$B$
$\mathbb{P}(X)$	1/4	1/2	1/4

$\mathbb{P}(X \cap Y) = \mathbb{P}(Y X)$	$R$	$V$	$B$
	1/8	1/8	1/16
	1/16	1/4	1/16
	1/16	1/8	1/8

	$\mathbb{P}(Y) = \sum \mathbb{P}(X \cap Y)$
$R$	5/16
$V$	3/8
$B$	5/16

Et donc en utilisant que  $\mathbb{P}(X|Y) = \frac{\mathbb{P}(X \cap Y)}{\mathbb{P}(Y)}$

$Y X$	$R$	$V$	$B$
$R$	2/5	2/5	1/5
$V$	1/6	2/3	1/6
$B$	1/5	2/5	2/5

3.

$$\begin{aligned} H(X|Y) &= \sum_X \sum_Y \mathbb{P}(X \cap Y) \times \log_2 \frac{1}{\mathbb{P}(X \cap Y)} \\ &= 1,42 \leq H(X) \end{aligned}$$

Il ne passe que 0,008 bits (différence entre  $H(X)$  et  $H(X|Y)$ ).

**Fin correction exercice 5**

**Exercice 6 – Transmission bruitée suite**

On considère une source binaire symétrique  $X = \{0, 1\}$ , donc  $p = 1/2$  émettant sur un canal bruité symétrique, stationnaire et sans mémoire vers un puits également et symétrique  $Y = \{0, 1\}$ . La matrice de bruit  $B_{Y|X}(q)$  est définie par la seule quantité  $q$  telle que :

- $\mathbb{P}(y = 1|x = 0) = \mathbb{P}(y = 0|x = 1) = q$
- $\mathbb{P}(y = 0|x = 0) = \mathbb{P}(y = 1|x = 1) = 1 - q$

1. Calculer les quantités :
  - (a)  $\mathbb{P}(x = 0|y = 0)$
  - (b)  $\mathbb{P}(x = 0|y = 1)$

- (c)  $\mathbb{P}(x = 1|y = 0)$
- (d)  $\mathbb{P}(x = 1|y = 1)$
- (e)  $\mathbb{P}(y = 0)$
- (f)  $\mathbb{P}(y = 1)$

- 2. En déduire  $H_Y, H_{X|Y}$  et  $I_{X;Y}$  en fonction de  $q$ .
- 3. Etudier et interpréter  $I_{X;Y}(q)$ .

---

Correction exercice 6

---

- 1.
- 2.

---

Fin correction exercice 6

---

## 2 Compression

### Exercice 7 – Sur l’existence d’un compresseur universel

- 1. Donner le nombre de mots pour coder un mot de longueur  $n$  sur l’alphabet  $\{0, 1\}$  ?
- 2. Donner le nombre de mots pour coder un mot de longueur au plus  $n - 1$  sur l’alphabet  $\{0, 1\}$  ?
- 3. Conclure.

---

Correction exercice 7

---

- 1. Il y a  $2^n$  mots
- 2. Il y a  $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ , soit un total de  $\frac{2^n - 1}{2 - 1}$  fichiers strictement moins que  $N$  bits.
- 3. Ceci prouve que soit au moins deux fichiers distincts de  $N$  bits seront compressés de manière identique, et donc il y aura eu perte. En effet, dans ce cas la compression il sera impossible de savoir vers lequel des fichiers initiaux décompresser. L’autre solution est que certains fichiers compressés font plus de  $N$  bits. Dans ce cas il n’y a pas eu compression.

---

Fin correction exercice 7

---

### Exercice 8 – Code préfixe

Soit  $S$  la source d’alphabet  $\{a, b, c, d\}$  et de probabilité  $P(a) = 0,5, P(b) = 0,25, P(c) = 0,125, P(d) = 0,125$ . On code  $S$  avec le code suivant :  $a = 0, b = 10, c = 110, d = 111$ .

- 1. Calculer  $adbccab$ . Décoder 1001101010.
- 2. Est-ce un code instantané ?
- 3. Calculer l’entropie de la source.

---

Correction exercice 8

---

- 1. Le mot de code pour  $adbccab$  est 011110110110010, et le mot de source pour 1001101010 est  $bacbb$
- 2. On vérifie facilement en comparant tous les mots deux à deux qu’aucun n’est préfixe d’un autre. C’est donc un code instantané.
- 3. L’entropie de la source est  $H = 1,75$ .

---

Fin correction exercice 8

---

### Exercice 9 – Compression

On veut construire un code compresseur binaire sur une source  $\mathcal{S} = (S, P)$ . (supposée infinie) où  $S = (0, 1)$  est l’alphabet source  $P = (P(0) = p, P(1) = 1 - p)$  est la loi de probabilité de  $\mathcal{S}$ . On propose le code suivant : on compte le nombre d’occurrences de « 0 » avant l’apparition d’un « 1 ». Les deux règles de codage sont :

- Une chaîne de quatre « 0 » consécutifs (sans « 1 ») est codée par 0.
- Si au moins de quatre « 0 » apparaissent avant le symbole « 1 » on code la chaîne par le mot de code «  $1e_1e_2$  » où  $E_1e_2$  est la représentation binaire du nombre de « 0 » apparus avant l’« 1 ». Par exemple l’apparition de quatre zéros consécutifs « 0000 » est codée par « 0 » tandis que l’occurrence « 001 » est codé par « 110 » car deux « 0 » sont apparus avant le « 1 » et que « 10 » est la représentation binaire de deux.

- 1. Expliciter les cinq mots de code. Le code possède-t-il la propriété du préfixe ?
- 2. Sachant que la probabilité d’apparition de deux symboles successifs  $s_1$  et  $s_2$  est, dans notre cas où l’on suppose la source sans mémoire,  $p(s_1) \times p(s_2)$ , calculer la probabilité d’occurrence dans  $S$  d’une chaîne de  $k$  « 0 » suivis d’un « 1 ».
- 3. Pour chaque mot de code, calculer le nombre de bits de code nécessaires par bit de source. En déduire le taux de compression de ce code, c’est à dire la longueur moyenne par bit de source.

---

Correction exercice 9

---

- 1. 0 n’est préfixe d’aucun autre mot de code, car ils commencent tous par 1. Tous les autres mots de code sont de même longueur, ils ne peuvent donc pas être préfixes les uns des autres. Autre preuve : on peut dessiner un arbre de Huffman contenant tous les mots de code
- 2.  $P(0 \dots 01) = P(0) \dots P(0)P(1) = p^k(1 - p)$ .

- Pour les mots de code (100, 101, 110, 111, 0), les rapports (nombres de bits de code par bit de source) sont (3/1; 3/2; 3/3; 3/4; 1/4). Le taux de compression est donc  $l = 3(1-p) + 3/2(1-p)p + (1-p)p^2 + 3/4(1-p)p^3 + 1/4p^4 = 3 - 3/2p - 1/2p^2 - 1/4p^3 - 1/2p^4$ .

---

Fin correction exercice 9

---

### Exercice 10 – Compression

- Compresser le mot ci-dessus avec la méthode « Run Length Coding » AAAABBCCCBCCCCAC
- Quelle est la taille du codage obtenu ?
- Quelle est la condition structurelle permettant de garantir qu'un mot est compactable ?
- Quelle est la condition fonctionnelle permettant de garantir qu'un mot est compactable ?

---

Correction exercice 10

---

- La solution est la suivante : #A4#B2#C3#B2#C3#A1#C1
- Les champs qui sont essentiels :
  - le nombre de caractères à coder (caractères de l'alphabet)
  - Donner la caractère à coder + le nombre de bits pour le coder + les bits de codages
- Il faut que le caractère # n'apparaissent par le mot compressé c'est à dire hors alphabet.
- Sur la coté fonctionnel, il faut que au moins que dans la séquence tros caractères identiques contigue. Il y a aussi d'autres conditions.

---

Fin correction exercice 10

---

### Exercice 11 – Méthode topologique

- Compresser la source composé par deux mots suivants avec la méthode topologique : AAAABBCC et CBBCCCAC.
- Quelle est la taille du code obtenu ?
- Quelle est la condition topologique permettant de garantir qu'un mot est compactable ?
- Peut-on recompresser, c'est à dire appliquer à nouveau la méthode topologique sur la nouvelle source obtenue ?

---

Correction exercice 11

---

Codage topologique. Il fonctionne sur les octets et s'applique sur les fichiers contenant un octet  $O$  sensiblement dominant. Le principe est le suivant : on lit les données par mots de 8 octets, et on code le mot par un octet topologique suivi d'un sous-mot. Dans l'octet topologique, les bits mis

à 1 désignent la position de l'octet dominant dans le bloc et les bits mis à 0 désignent la position des autres caractères reproduits dans le sous-mot. Par exemple, le mot AABDEACA sera codé 11000101 puis BDEC.

- Nous obtenons 11110000ABBCC, donc 6 octets au lieu de 8 octets. et nous obtenons 10011101CBBA sur 5 octets.
- oui, et voici le code obtenu :
  - 11110000A00001100B0000011C,
  - Il faut des longues séquences répétitives
- Oui on peut recompresser en 11110000ABBCC en 11110000A1100BCC et ainsi de suite

---

Fin correction exercice 11

---

### Exercice 12 – Méthode RLE

- Quel serait le codage RLE de l'image donnée par la figure ?? ?
- Y-a-t'il un gain ?
- Quel serait le codage sur bloc de 5 bits en 16 niveaux de gris ? ET le gain ? Peut-on compresser plus avec ce système ?
- Et en 255 niveaux de gris (avec au plus 16 pixels identiques consécutifs) ?
- Et en colonnes ?

---

Correction exercice 12

---

- Par exemple, 680131414131413141312222611, où 68 ce cod pour les dimensions de l'image (6 × 8 pixels), puis chaque chiffre pour le nombre de pixels consécutifs de même couleur (alternance blanc-noir).
- 25 octes (ou 13 en limitant à des blocs de taille 16) au lieu de 6, plutôt une perte sur les petites images.
- 

---

Fin correction exercice 12

---

### Exercice 13 – Méthode Move-To-Front

Soit  $A$  l'alphabet sur 8 symboles suivant :  $A = (a, b, c, d, m, n, o, p)$ .

- On associe à chaque symbole un numéro entre 0 et 7, selon leur position alphabétique. Quels nombres représentent « abcdcdcbamnopnm » et « abcdmnoabcdmno » ?
- Coder les deux chaînes précédentes en utilisant la méthode Move-To-Front. Que constatez-vous ?

- Combien de bits sont nécessaires pour coder les deux premiers nombres, par Huffman par exemple ?
- Combien de bits sont nécessaires pour coder les deux nombres obtenus après Move-To-Front ? Comparer les entropies.
- Que donne Huffman étendu à deux caractères sur le dernier nombre ? Quelle est alors la taille de la table des fréquences ?
- Comment pourrait-on qualifier l'algorithme composé d'un Move-To-Front suivi d'un code statistique ?
- Le Move-ahead- $k$  est une variation du Move-To-Front où le caractère est seulement avancé de  $k$  positions au lieu du sommet de la pile. Encoder les deux chaînes précédentes avec  $k = 1$  puis  $k = 2$  et comparer les entropies.

---

**Correction exercice 13**

---

- « 0123321045677654 » et « 0123456701234567 »
- « 0123012345670123 » et « 0123456777777777 ». La deuxième chaîne a une entropie visiblement réduite.
- Les fréquences sont égales à  $1/8$ , l'entropie est donc maximale et 3 bits sont nécessaires pour chaque caractère. Il faut donc  $3 \times 16/8 = 6$  octets.
- Pour la première chaîne on obtient une entropie :  $H = 4(\frac{3}{16} \log_2(\frac{16}{3})) + 4(\frac{1}{16} \log_2(\frac{16}{1})) \equiv 1.81 + 1 = 2.81$ . Par Huffman le code est donc : 00, 01, 100, 101, 1100, 1101, 1110, 1111, d'où un codage sur seulement  $2 \times 3 \times 2 + 2 \times 3 \times 3 + 4 \times 1 \times 4 = 46 = 5,75$  octets.  
La deuxième chaîne s'y prête encore mieux :  $H = 7(\frac{1}{16} \log_2(\frac{16}{1})) + 7(\frac{9}{16} \log_2(\frac{16}{9})) \equiv 1.75 + 0.47 = 2.22$ . Par Huffman le code est donc : 00, 01, 100, 101, 1100, 1101, 1110, 111, 1, d'où un codage sur seulement  $30 = 3,75$  octets.
- Huffman donne : 000, 001, 010, 011, 1 pour respectivement 12, 34, 56, 67, 77. Ainsi,  $4 \times 3 + 4 \times 1 = 16 = 2$  octets sont nécessaires mais avec une table de taille 256 octets au lieu de 8 triplets soit 3 octets.
- Un Move-to-front suivi d'un code statistique est donc un code statistique localement adapté.
- Pour  $k = 1$  on obtient « 0123220345676644 » et « 0123456770123456 » d'entropies respectives  $H = 2,858$  et  $H = 2,953$ . Pour  $k = 2$ , cela donne : « 0123112345675552 » et « 0123456777012347 » d'entropies respectives  $H = 2,781$  et  $H = 2,875$ .

---

**Fin correction exercice 13**

---

**Exercice 14 – Analyse du code BWT**

- La dernière colonne  $L$  de la matrice triée de la transformation BWT contient des concentrations de caractères identiques, c'est pourquoi  $L$  se compresse bien. Toutefois, la première colonne  $F$  se compresserait encore mieux puisqu'elle est totalement triée. Pourquoi sélectionner  $L$  plutôt que  $F$  comme code ?

- Soit la chaîne  $S = \lll sssssssh \ggg$ . Calculer  $L$  et sa compression Move-To-Front.
- Implémentation pratique : BWT est efficace sur de longues chaînes  $S$  de taille  $n$ . En pratique, il est donc impensable de stocker toute la matrice  $n \times n$  des permutations. En fait, il faut seulement trier ces permutations et pour les trier, il suffit d'être capable de comparer deux permutations.
  - Donner un indice qui soit capable de désigner et différencier les permutations de la chaîne de départ.
  - Construire un algorithme *comparerpermutations* calculant une fonction booléenne qui a pour entrée une chaîne  $S$  et deux entiers  $i$  et  $j$ . cet algorithme doit décider si la chaîne décalée  $i$  fois est avant la chaîne décalée  $j$  fois, dans l'ordre obtenu en triant toutes les permutations par ordre alphabétique.
  - Conclure sur l'espace mémoire nécessaire pour calculer la BWT.
  - Comment calculer  $L$  et l'index primaire à partir du tableau des permutations ?

---

**Correction exercice 14**

---

- Car le message initial peut être reconstruit à partir de  $L$  et de l'index primaire, mais pas à partir de  $F$ .
- Ici  $S = L = "ssssssssh"$  Move-to-front donne  $C = (1, 0, 0, 0, 0, 0, 0, 1)$  ; qui est aussi la sortie de Huffman.
- Implémentation pratique
  - On peut se servir de la position du premier caractère du message source

---

**Algorithm 1** Solution à l'exercice Analyse move-to-front

---

```
(b) while on n'a pas de réponse do
    if  $S[i] < S[j]$  then
        Répondre  $i$  est avant
    else if  $S[i] > S[j]$  then
        Répondre  $j$  est avant
    else
         $i := (i + 1) \bmod n$  et  $j := (j + 1) \bmod n$ 
    end if
end while
```

---

- Il faut pouvoir stocker  $S$  le message source, puis  $T$  contenant les indices de permutations et enfin  $L$  que l'on calcule par  $S[T[i] - 1]$ .
- L'index primaire est celui pour lequel  $T[i] = 1$  puisque c'est là que se trouve la ligne 1.

---

**Fin correction exercice 14**

---

**Exercice 15 – Algorithme de Shannon-Fano**

1. Compresser la source suivante avec l'algorithme de Shannon-Fano AAAABBCCCBDDCAE.
2. Quelle est la taille du code obtenu ?
3. Quelle est la taille de l'en-tête ?
4. Peut-on ré-appliquer l'algorithme sur la nouvelle source obtenue (en-tête non comprise) ?

Correction exercice 15

1.  $f(A) = 5/15 = 0,3333, f(B) = 3/15 = 0,2, f(C) = 0,266, f(D) = 0,1333$  et  $f(E) = 0,06666$ .  
 En utilisant l'algorithme de Shannon-Fano,
  - nous obtenons  $A = B = 1$  et  $C = D = E = 0$
  - Ensuite, nous redécoupons chaque partie en deux, nous trouvons donc :  
 -  $A = 10$  et  $B = 11$  et  $C = 01$  et  $D = 001$  et  $E = 000$ .
2. La taille du code est 33 bits.
3. Pour coder l'entête nous allons utiliser la méthode suivante :
  - **Nombre de caractère à coder sur un octets**
  - **Pour chaque caractère à coder, nous considérons un triplet** (*caractrecoder, nombredebitspourcode:*
  - **Nombre de caractère à coder sur un octets 5**
  - **Pour chaque caractère à coder, nous considérons un triplet** (*caractrecoder, nombredebitspourcode:* (A, 10, 11)
  - **Pour chaque caractère à coder, nous considérons un triplet** (*caractrecoder, nombredebitspourcode:* (B, 10, 10)
  - **Pour chaque caractère à coder, nous considérons un triplet** (*caractrecoder, nombredebitspourcode:* (C, 10, 01)
  - **Pour chaque caractère à coder, nous considérons un triplet** (*caractrecoder, nombredebitspourcode:* (D, 11, 001)
  - **Pour chaque caractère à coder, nous considérons un triplet** (*caractrecoder, nombredebitspourcode:* (E, 11, 000)
 La taille de l'entête est de 72 bits donc,

Fin correction exercice 15

**Exercice 16 – Pile ou Face pour jouer au 421**

On souhaite jouer au lancé de dé, avec pour unique moyen une pièce de monnaie. On va donc chercher à coder un dé non pipé à 6 faces avec une pièce non pipée à deux faces.

1. Quelle est l'entropie d'un dé ?
2. Proposer un algorithme de codage.
3. Calculer la longueur moyenne de ce codage.
4. Ce codage est-il optimal ?

1. L'idée est de procéder par rejet (avec relance de la pièce si nécessaire). Pour cela, il faut générer 6 événements de probabilités uniformes avec une pièce en tirant 3 fois pile ou face et en associant les événements suivants à un jet de dé

Tirages	PPP	PPF	PPF	PPF	FPP	FPF
Jet de dé	1	2	3	4	5	6

Si on trouve FFP ou FFF ; on rejette le tirage (on peut donc s'arrêter après deux jets, dès que l'on a FF-) et on lance relance trois fois la pièce (tirages indépendants)

2. Le nombre  $N$  moyen de lancers de pièces est donc :

$$\begin{aligned}
 N &= \frac{6}{8} \times 3 + \frac{2}{8} \times \frac{6}{8} \times (2+3) + \left(\frac{2}{8}\right)^2 \times \frac{6}{8} \times (2+2+3) + \dots \text{ ou encore} \\
 N &= \frac{6}{8} \times \sum_{k=0}^{\infty} (2k+3) \left(\frac{2}{8}\right)^k \\
 N &= 3 \times \frac{6}{8} \times \frac{8}{6} + 2 \times \frac{6}{8} \times \frac{2}{8} \times \left(\frac{2}{8}\right)^2 \\
 N &= 3 + (2/3)
 \end{aligned}$$

Soit une moyenne de 11 pile ou face pour trois dés du 421.

Nous rappelons que

$$\begin{aligned}
 \sum_{j=0}^{\infty} (j+1)x^j &= \left(\sum_{j=1}^{\infty} x^k\right)' \\
 &= \left(\frac{1}{1-x}\right)' \\
 &= \frac{1}{(1-x)^2}
 \end{aligned}$$

3. L'entropie d'un dé non pipé  $6 \times (1/6) \log_2 6 = 2,58$
4. La longueur moyenne du coage est donc  $3 \times 2,58 = 7,755$ , ce codage n'est sans doute pas optimal. En effet, le cas de l'extension de source à 3 dés permet de faire un encodage sur 8 pile ou face, puisque  $6^3 = 216 < 256 = 2^8$ . Dans ce cas, la moyenne du nombre de tirages nécessaires avec rejets est plus proche de 8,5.

Fin correction exercice 16

**Exercice 17 – Algorithme de Huffman : aspect théorique**

Cet exercice introduit des éléments théoriques sur la valeur du code généré par l'algorithme de Huffman. Soit  $S = (S, P)$ , où  $S = (0, 1)$ , et  $P(0) = 0,99$  et  $P(1) = 0,01$ .

1. Calculer l'entropie de  $\mathcal{S}$ .
2. Donner le code généré par l'algorithme de Huffman sur la troisième extension  $\mathcal{S}^3$ . Quel est le taux de compression ?
3. Que pouvez-vous dire de l'optimalité de l'algorithme de Huffman en comparant les taux de compression obtenus avec ceux de l'exercice précédent ? Cela est-il conforme au théorème de Shannon ?

Correction exercice 17

1.  $H(X) = 0,99 \log_2 \frac{1}{0,99} + 0,01 \log_2 \frac{1}{0,01} = 0,0808$
2. Nous obtenons :
  - $\mathbb{P}(000) = \mathbb{P}(0) \times \mathbb{P}(0) \times \mathbb{P}(0) = 0,97$
  - $\mathbb{P}(001) = 0,0098$
  - $\mathbb{P}(010) = 0,0098$
  - $\mathbb{P}(011) = 0,000099$
  - $\mathbb{P}(100) = 0,0098$
  - $\mathbb{P}(101) = 0,000099$
  - $\mathbb{P}(110) = 0,000099$
  - $\mathbb{P}(111) = 0,000000001$

$S$	$C$
000	1
001	010
010	011
100	001
011	00011
101	00010
110	00001
111	00000

3. Nous obtenons
4. Nous obtenons la longueur moyenne :  $l = 1,05$  soit par bit de source  $l = 0,35$

Fin correction exercice 17

**Exercice 18 – Algorithme de Huffman**

On cherche à coder des jets successifs (en nombre supposé infini) d'un dé faussé. Les symboles de la source sont notés  $(1, 2, 3, 4, 5, 6)$ , et suivent la loi de probabilité d'apparition  $(0,12; 0,15; 0,16; 0,17; 0,18; 0,22)$ .

1. Quelle est l'entropie de cette source ?
2. Proposer un code ternaire (sur un alphabet à trois chiffres) issu de l'algorithme de Huffman pour cette source. Quelle est sa longueur moyenne ? Quelle longueur moyenne minimale peut-on espérer pour un tel code ternaire ?
3. Même question pour un code binaire

1.  $H(d) = 2,58$
2. sur  $\{0, 1, 2\}$ , Huffman construit le code (par exemple) :
  - $1 \rightarrow 10$
  - $2 \rightarrow 12$
  - $3 \rightarrow 01$
  - $4 \rightarrow 00$
  - $5 \rightarrow 02$
  - $6 \rightarrow 2$
 de longueur moyenne (fixe) égale à  $0,22 * 1 + (1 - 0,22) * 2 = 1,78$  chiffres. C'est le code optimal (par théorème), on ne pourra obtenir mieux (et toujours plus de  $2,58 / \log_2(3) = 1,62$ ) qu'en codant la source par séquences de chiffres.
3. Sur  $\{0, 1\}$ , un code possible est :
  - $1 \rightarrow 000$
  - $2 \rightarrow 001$
  - $3 \rightarrow 010$
  - $4 \rightarrow 011$
  - $5 \rightarrow 10$
  - $6 \rightarrow 11$
 de longueur moyenne 2,6. C'est aussi le code optimal.

Fin correction exercice 18

**Exercice 19 – Suite de l'exercice précédent**

L'algorithme de Huffman construit un code où des mots de source de longueur fixes sont codés par des mots de code de longueur variable. L'organisation de la mémoire où l'on stocke le code de longueur fixe aux mots de code, et on code alors des séquences de longueur variable des chiffres de la source.

Les codes dits de Tunstall sont des codes optimaux qui suivent ce principe. En voici la méthode de construction dans le cas d'un alphabet binaire. Si la longueur choisie des mots de code est  $k$ , il faut choisir les  $2^k$  séquences de chiffres de source qu'on va choisir de coder. Au départ, l'ensemble des candidats est l'ensemble des mots de longueur 1 (ici  $\{1, 2, 3, 4, 5, 6\}$ ). Soit dans cet ensemble le mot le plus probable (ici, 6). On construit toutes les séquences réalisables en rajoutant un chiffre à ce mot, et on remplace ce mot par ces séquences dans l'ensemble des candidats (ici on obtient alors  $\{1, 2, 3, 4, 5, 61, 62, 63, 64, 65, 66\}$ ). On recommence alors cette opération tant que la taille de l'ensemble des candidats n'est pas strictement supérieure à  $2^k$  (on arrête avant d'avoir dépassé cette valeur). On code alors toutes ces séquences par des mots de longueur  $k$ .

1. Sur un alphabet binaire, construire un code de Tunstall pour le dé, pour des longueurs de mots de code  $k = 4$ .
2. Comment est codée la séquence « 6664 » par Tunstall ? Par Huffman ?

3. Pour chaque mot de code, calculer le nombre de bits de code nécessaires par caractère de source. En déduire la longueur moyenne par bit de source code de Tunstall.

Correction exercice 19

- On code les séquences  $\{1, 2, 3, 4, 51, 52, 53, 54, 55, 56, 61, 62, 63, 64, 65, 66\}$  (au nombre de 16) avec les 16 mots de longueur 4.
- Selon le code choisi (dont ne dépend pas la longueur des équences), on peut avoir par exemple :
  - Tunstall :  $66 \rightarrow 1111$  et  $64 \rightarrow 1100$  : 11111100 et
  - Huffman 11|11|11|011
- Pour coder les équences de deux chiffres, on a besoin de 4 bits, soit un rendement de 2, et de même pour les mots de un chiffre, soit un rendement de 4. La longueur moyenne par bit de source est donc :  $2 * 0,22 + 2 * 0,18 + 4 * 0,6 = 3,2$ .

Fin correction exercice 19

Exercice 20 – Algorithme de Tunstall

Nous poursuivons l'étude de l'algorithme de Tunstall. Soit  $S = (S, P)$ , où  $S = (A, B, C)$ , et  $P(A) = 0,6$ ,  $P(B) = 0,3$  et  $P(C) = 0,1$ .

- Quelle est la relation entre  $N$  le nombre d'éléments dans  $S$ ,  $K$  le nombre d'itérations et  $k$ .
- Donner un 2-bit code tel que le décodage de la séquence suivante soit impossible AAABAABAABAABAAA.
- Donner un 3-bit code de Tunstall pour la source décrite ci-avant.
- Coder AAABABCBA.

Correction exercice 20

- $N + K(N - 1) \leq 2^k$ . On rajoute  $N - 1$  éléments.
- Il suffit de considérer :

Sequence	Mot de code
AAA	00
ABA	01
AB	10
B	11

Nous devons être capable de parser la source de telle manière que les séquences obtenus par l'algorithme apparaissent dans le mot à coder.

3. Nous faisons le produit des probabilité pour le caractère AA

Lettre	Probabilité	Lettre	Probabilité
A	0,6	B	0,3
B	0,3	C	0,1
C	0,1	AA	0,36
		AB	0,18
		AC	0,06

Voici la solution finale :

Lettre	Mots de codes
B	000
C	001
AB	010
AC	011
AAA	100
AAB	101
AAC	110

Fin correction exercice 20

Exercice 21 – Algorithme de Huffman

Considérons la source suivante : « Le president est entre dans la salle » Nous avons supprimé les accents, mais les espaces sont comptés.

- Construire la table des fréquences.
- Construire l'arbre de Huffman.
- Compresser la source avec le codage Huffman.
- Calculer la taille de l'en-tête.
- Quel est le gain de compression ?

Correction exercice 21

1. Nous obtenons les codes suivants :

- I = 00000
- D = 0001
- A = 0011
- T = 0111
- S = 101
- E = 11
- P = 00001
- R = 0010
- N = 0110
- L = 100
- = 010
- Soit un total de 118 bits, donc un gain  $118/288 = 0,59$ , environ 60%.

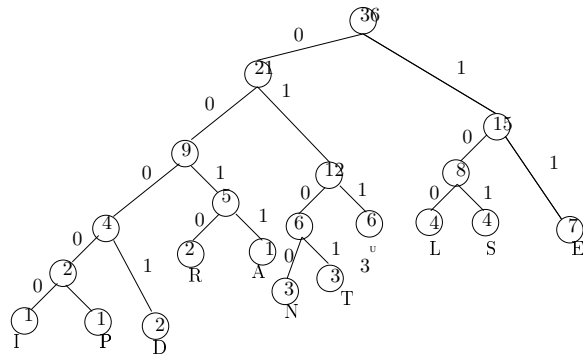


FIG. 1 – Arbre de Huffman

2. L'entropie est  $H(X) = \sum p(x) \log_2(\frac{1}{p(x)}) = 1,84$

#symboles	symboles	# de bits	codage
11	I	0101	00000
	D	0100	0001
	A	0100	0011
	T	0100	0111
	S	0100	101
	E	0010	11
	P	0101	00001
	R	0100	0010
	N	0010	11
	L	0011	100
		0100	010

3. L'en tête est donné par le tableau suivant :

Avec l'entête on obtiens,  $181 + 118 = 299/288 = 1.01$ . En fait on peut gagner un peu encore il suffit d'enlever un bit à la colonne # bits (ici 11) et on trouve  $181 - 11 + 118 = 288$ .

Fin correction exercice 21

### Exercice 22 – Compression par Huffman

Une source émet des symboles de l'alphabet  $A = \{a, b, c, d, e\}$  avec les probabilités :

- $\mathbb{P}(a) = 0,15$
- $\mathbb{P}(b) = 0,04$
- $\mathbb{P}(c) = 0,26$
- $\mathbb{P}(d) = 0,05$
- $\mathbb{P}(e) = 0,5$

1. Calculer l'entropie de cette source.

2. Trouver un code de Huffman pour cette source.
3. Calculer la longueur moyenne du code de Huffman.

Correction exercice 22

- 1.
- 2.
3. 1,83 bits

Fin correction exercice 22

### Exercice 23 – Algorithme de Huffman

Considérons la source suivante : *GATTACA* répété deux fois.

1. Construire la table des fréquences.
2. Calculer l'entropie de la source.
3. Compresser la source avec le codage Huffman.
4. Quelle est la taille du code compressé (sans l'entête) ?
5. Quelles sont les tailles obtenues si le motif est répété 5, 10, 20 fois ? Calculer le gain avec et sans entête avec les répétitions.

Correction exercice 23

1. L'entropie est  $H(X) = \sum p(x) \log_2(\frac{1}{p(x)}) = 1,84$

#symboles	symboles	# de bits	codage
4	G	11	000
	C	11	001
	T	10	01
	A	01	1

2. L'en tête est donné par le tableau suivant :

3. La solution via un arbre de Huffman est donnée par la figure 2

# répétitions	1	5	10	20
#symboles	7	35	70	140
code avec entête	70	122	187	317
code sans entête	13	65	130	160
gain	10	3,5	2,5	2,2
gain	1,85	1,85	1,85	1,85

4. Le calcul du gain en fonction du nombre

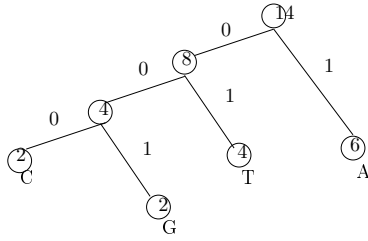


FIG. 2 – Arbre de Huffman pour le mot « Gataca »

Fin correction exercice 23

#### Exercice 24 – Huffman dynamique

Nous considérons la séquence  $aaa\ aaa\ aaa\ bcd\ bcd\ bcd$ .

1. Quel serait le codage de Huffman dynamique de cette séquence ?
2. Quel serait le codage de Huffman de l'extension à trois caractères de cette séquence ?

Correction exercice 24

1.  $@a|1|0|0|0|0|0|0|0|0|@b|@c|@d$  à l'arbre dynamique est
  - $a \rightarrow 0$
  - $@ \rightarrow 10$
  - $b \rightarrow 110$
  - $c \rightarrow 1110$
  - $d \rightarrow 1111$

la suite de la compression est donc

Fin correction exercice 24

#### Exercice 25 – Compression par LZ77

Compresser la séquence suivante  $(ab)^{16}$  avec LZ77 avec :

1.  $N = 12$  et  $F = 5$
- 2.

Correction exercice 25

1. Nous trouvons :  $(0, 0, a), (0, 0, b), (2, 2, a), (4, 4, b), (6, 5, b), (6, 5, b), (6, 5, b), (4, 4, \text{""})$ .
2. La solution est  $(0, 0, a)$  et  $(0, 0, b)$  car ces deux caractères sont encore inconnus. Ensuite, le début de la répétition est positionné sur le  $ab$  suivant. La plus longue correspondance est alors de 14 répétitions de  $ab$ , deux caractères avant. La compression se termine par le triplet  $(2, 28, \text{""})$ .

Fin correction exercice 25

#### Exercice 26 – Compression par LZ77

Reprendre l'exemple du cours avec les paramètres  $N = 12$  et  $F = 5$ .

Correction exercice 26

Nous obtenons la suite de triplets suivants :

$(0, 0, l), (0, 0, e), (3, 1, m), (0, 0, a), (0, 0, g), (5, 2, d), (0, 0, i), (0, 0, t), (4, 1, a), (0, 0, b), (0, 0, r), (3, 1, c), (2, 1, d), (7,$

Fin correction exercice 26

#### Exercice 27 – Compression par LZ78

Considérons la source suivante : « MAMAN AIME LES MANGUES » Nous avons supprimé les accents, mais les espaces sont comptés.

1. Compresser la source avec le codage LZ78.
2. Construire l'arbre de décompression.
3. Quel est le gain de compression ?

Correction exercice 27

0	1	2	3	4	5	6	7
ε	M	A	MA	N		AI	ME
8	9	10	11	12	13	14	15
L	E	S	M	AN	G	U	ES
0	1	2	3	4	5	6	7
ε	0M	0A	1A	1N	0	2I	1E
8	9	10	11	12	13	14	15
5L	0E	0S	5M	2N	0G	0U	9S

1. Déterminons le dictionnaire

Coder chaque lettre par son codage ascii, et coder chaque entier apparaissant dans le *nieme* bloc en  $\lceil \log_2 n \rceil$  bits.

2. L'arbre de décode est donné par la figure 3
3. Le nombre de bits initial est  $176 = 22 \times 8$  et le fichier compressé (les entiers sont codés sur  $\lceil \log_2 n \rceil$  bits donc  $15 \times 8 + 15 \times 4 = 180$ ). Donc aucune compression.

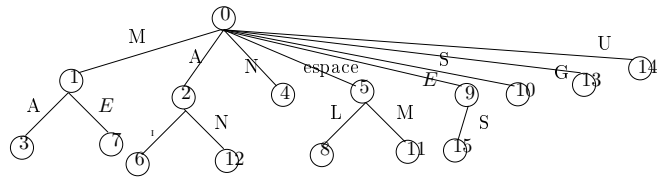


FIG. 3 – Arbre de décodage pour LZ78

Fin correction exercice 27

**Exercice 28 – Compression suite**

Considérons la source composé par la motif « ATGC » répété deux fois.

- Calculer l'entropie de la source.
- Compresser la source avec le codage LZ78.
- Quelle est la taille du code compressé ?
- Quelles sont les tailles obtenues si le motif est répété 3 , 4 5 fois ? Donner les arbres de décompression pour chaque cas.

Correction exercice 28

1. nous avons  $p(A) = p(T) = p(G) = p(C) = 1/4$ . Nous avons également  $I(A) = I(T) = I(G) = I(C) = \log 4 = 2$ . D'où  $H(X) = 2$ .

2. nous obtenons le tableau suivant :

0	1	2	3	4	5	6
$\epsilon$	A	T	G	C	AT	GC
	0A	0T	0G	0C	1T	3C
codage	A	0T	00G	00C	001T	011C

3. taille code compressé = 60 bits + entête. Code non compressé 64 bits

4. Si on considère le motif trois fois

0	1	2	3	4	5	6	7	8
$\epsilon$	A	T	G	C	AT	GC	ATG	C
codage	A	0T	00G	00C	001T	011C	101G	000C

la taille est 88 bits. 96 sans compression

5. Si on considère le motif quatre fois

0	1	2	3	4	5	6	7	8	9	10
$\epsilon$	A	T	G	C	AT	GC	ATG	CA	TG	C
codage	A	0T	00G	00C	001T	011C	101G	100A	0010G	0000C

la taille est 110 bits

6. Si on considère le motif cinq fois

0	1	2	3	4	5	6	7	8	9	10	11
$\epsilon$	A	T	G	C	AT	GC	ATG	CA	TG	CAT	GC
codage	A	0T	00G	00C	001T	011C	101G	100A	1000T	1100C	110C

la taille est 132 bits

7. L'évolution de l'arbre est donnée par la figure 4.

Fin correction exercice 28

**Exercice 29 – Compression**

Une source émet des symboles de l'alphabet  $A = \{a, b, c\}$  avec les probabilités :

- $\mathbb{P}(b|a) = 0,5$
- $\mathbb{P}(c|a) = 0,5$
- $\mathbb{P}(a|b) = 1$
- $\mathbb{P}(a|c) = 0,5$
- $\mathbb{P}(b|c) = 0,5$

La source émet 4 symboles et commence par émettre « a ».

- Quelles sont les séquences possibles émises par la source ?
- Calculer la longueur d'un code LZ78 obtenu pour chaque séquence.
- En déduire la longueur moyenne du codage LZ78 pour cette source.

Correction exercice 29

1. abab, abac, acac, acab, acba

2. 27 bits

3. idem

$\epsilon$	1	2	3
	a	b	ac
4.	0a	0b	1c
	a	0b	01c

Fin correction exercice 29

**Exercice 30 – Compression fin**

Alice, Bob et Dominique sont co-locataires d'un grand appartement. Ils mettent en commun leur importante collection de CD et cherchent un moyen de coder chacun de leur CD de telle sorte qu'il soit facile de trouver à qui appartient le CD et de quelle genre il est parmi : Classique, Jazz, Variétés.

- Alice possède 256 CD : 1/2 de classique, 1/4 de jazz et 1/4 de Variétés.
- Bob possède 512 CD : 1/4 de classique, 1/4 de jazz et 1/2 de Variétés.

– Diminiquè possède 256 CD : 1/4 de classique, 1/2 de jazz et 1/4 de Variétés.

Nous notons ainsi la collections :  $A$  Alice,  $BBOB$ ,  $D$  Dominique,  $C$  classique,  $J$  Jazz,  $V$  variétés,  $\Gamma$  tout

Chaque CD est représenté par deux lettres : le nom de la personne suivi du genre. Par exemple  $AC$  est un CD Classique de Alice.

1. Calculer l'entropie de la collection  $\Gamma$ .
2. Trouver un codage de Huffman pour la collection  $\Gamma$ .

---

**Correction exercice 30**

---

- 1.

---

**Fin correction exercice 30**

---

### Exercice 31 – Compression

Une source émet un ensemble de symboles  $\mathcal{A} = \{a, b, c, d\}$  avec les probabilités :  $p(a) = 0,5, p(b) = 0,25, p(c) = 0,2, p(d) = 0,05$ .

Nous supposons que le  $a$  apparait en une longue série de taille  $n$ . Les autres lettres apparaissent aléatoirement.

1. Calculer l'entropie de cette source
2. Trouver un code de Huffman pour cette source.
3. Si l'on code la source avec NRZ puis Huffman, pour quelle valeur de  $n$  obtient-on un gain de compression ?

---

**Correction exercice 31**

---

- 1.

---

**Fin correction exercice 31**

---

## 3 Codes correcteurs et correcteurs d'erreurs

### Exercice 32 – Mesures des erreurs de transmissions

On a mesuré que les erreurs de transmission sur une boucle locale hertzienne pouvaient être modélisées par un canal binaire symétrique et sans mémoire, avec une probabilité d'erreur de  $1/1000$  sur chaque bit transmis. On transmet sur ce canal des trames d'une longueur de 1000 bits. Sachant que les erreurs sont indépendantes pour chaque bit :

1. Calculer le taux d'erreur d'une trame (probabilité d'au moins un bit erroné).  $\ln(1 + \epsilon) \equiv \epsilon$

2. Calculer le taux résiduel d'erreur après utilisation d'un code de parité simple. L'efficacité du code de parité sera calculée en faisant le quotient de la probabilité de détection d'une seule erreur sur la probabilité d'occurrences d'une ou deux erreurs.

---

**Correction exercice 32**

---

1. soit  $\tau = 1 - (1 - p)^n = 1 - (1 - \frac{1}{1000})^{1000}$ . En passant par le logarithme. On pose  $q = (1 - \frac{1}{1000})^{1000}$  donc  $\ln q = 1000 \times \log(1 - 1/1000) = -1000/1000 = -1$  donc  $qe^{-1} = 1/e$ , donc  $\tau = 1 - 1/E = \frac{e-1}{e} = 0,63$

2. Le calcul de  $E$  donne :

$$\begin{aligned} E &= \frac{np(1-p)^{n-1}}{np(1-p)^{p-1} + \frac{n(n-1)}{2}p^2(1-p)n-2} \\ &= \frac{1}{1 + \frac{(n-1)p}{2(1-p)}} \\ &= 2/3 \end{aligned}$$

$$Q = \tau(1 - E) = \frac{e-1}{3e} = 0,21 \text{ Le taux d'erreur a été divisé par trois.}$$

---

**Fin correction exercice 32**

---

### Exercice 33 – Construction empirique d'un code correcteur

On souhaite construire un code susceptible de corriger jusqu'à deux erreurs, pour une trame de deux bits.

1. Quelle doit être la distance minimum de ce code ?

Soit  $k$  le nombre de bits à coder,  $r$  le nombre de bits de contrôle à ajouter à  $k$  pour la correction des erreurs et  $n = k + r$ . La théorie des codes de Hamming nous indique que pour pouvoir corriger une erreur, les  $r$  bits de contrôle doivent pouvoir coder soit l'absence d'erreur, soit la position de l'erreur quand il y en a une. Il faut donc que  $r \geq \log_2(n + 1)$ .

2. En s'inspirant des codes de Hamming, exprimer l'inéquation permettant de calculer la valeur minimum à donner à  $r$  pour pouvoir corriger jusqu'à deux erreurs. On calculera pour cela le nombre de possibilités d'avoir 0, 1 ou 2 erreurs.
3. En utilisant les résultats des questions précédentes plus un peu d'imagination de d'I.N. (Intelligence Naturelle), construire un code éventuellement systématique, de longueur  $n = 8$  bits pouvant corriger deux erreurs pour un bloc à coder de dimension  $k = 2$  bits. Justifier la construction et les propriétés de ce code.

---

**Correction exercice 33**

---

- $D = 5$
- la relation  $r \geq \log_2(n+1)$  doit être vu comme  $r = \log(C_n^2 + C_n^1 + C_n^0)$  comme étant 2 erreurs parmi  $n$  plus 1 erreur parmi  $n$  + aucune erreur parmi  $n$ .

$$r = \log(C_n^2 + n + 1) \text{ la première correspond 2 erreurs après une et aucune erreur}$$

$$2^r \geq \frac{n(n-1)}{2} + n + 1$$

$$2^{r+1} \geq n^2 + n + 2 \text{ or } n = k + r = r + 2$$

$$2^{r+1} \geq (r+2)^2 + (r+1) + 2 = r^2 + 5r + 8$$

Il suffit ensuite de faire ds tests à la main pour trouver le bon  $r$ . On trouve donc  $r = 5$ .

- Il n'est pas possible de trouver un code avec  $D = 5, k = 2$  et  $n = 7$ , en prenant les codes

donnés par la figure 5. La table des codes est la suivante :

00	000000
01	111011
10	111100
11	000111

Si nous prenons  $D = 5, k = 2$  et  $n = 7$ , les codes sont données par la figure 6

$k = 2$	$r = 6$
00	000000
01	111011
10	111100
11	000111

La table des codes est la suivante :

Quand  $Y$  reçoit 11110100  $Y$  corrige en 10111100 (le mot de code le plus proche) et donc  $X$  a envoyé 10.

---

**Fin correction exercice 33**

---

**Exercice 34 – Codes linéaires**

On considère le code linéaire  $C$  dont la base est formée par trois mots :

- 00111
- 10101
- 11011

- Vérifier que ces mots forment bien une base.
- Que se passe-t'il si nous rajoutons le vecteur  $X_4 = 01110$  ?
- Quelles sont les valeurs  $k$  (dimension de  $C$ ,  $n$  longueur de  $C$  et  $R$  redondance de  $C$  ?
- Construire la matrice génératrice  $G$  de  $C$ .
- Construire les mots du code  $C$ .
- Montrer que la distance de ce code est égale à son poids.
- En conservant le même code, construire la matrice  $G'$  permettant un codage systématique (ou séparable). Expliquer sa construction.
- Construire la matrice de vérification  $H'$  correspondant à  $G'$ .

- Calculer à nouveau à partir de  $H'$  la distance  $d(\mathbb{C})$
- Est-ce un code de Hamming? Justifier.
- Le récepteur reçoit le  $Y_1 = 11111$ . Quel est son syndrome? Quel mot à été émis?
- Même question avec le mot  $Y_2 = 10011$  ?
- Quelle corrélation y a t'il avec la distance du code?
- Combien de code permet-il de détecter et de corriger d'erreurs?

---

**Correction exercice 34**

---

- Il faut résoudre le système suivant

- $X_1 = 00111$
- $X_2 = 10101$
- $X_3 = 11011$

$$\alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 = 0$$

- Si nous rajoutons  $X_4 = 01110$ , les quatres vecteurs ne sont linéairement indépendants.
- Nous avons  $k = 3, n = 5$  et  $R = 2/3$ .

- 

$$G = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

- Les mots de code sont obtenus en procédant au produit matricielle des mots de trois bits avec la matrice  $G$ . Nous obtenons le tableau suivant :

000	00000	code A
001	11011	code D
010	10101	code C
011	01110	code G
100	00111	code B
101	11100	code F
110	10010	code H
111	01001	code E

Ce qui revient à 0,  $X_1, X_2, X_3, X_1 + X_2, X_1 + X_3, X_2 + X_3, X_1 + X_2 + X_3$

- $\text{poids}(\mathbb{C}) =$  le mot qui a le mot de 1.

	code A	code B	code C	code D	code E	code F	code G	code H
code A	0	3	3	4	2	3	3	2
code B	3	0	2	3	3	4	2	3
code C	3	2	0	3	3	2	4	3
code D	4	3	3	2	0	3	3	4
code E	2	3	3	2	0	3	3	4
code F	3	4	2	3	3	0	2	3
code G	0	3	3	4	2	3	3	2
code H	2	3	3	2	4	3	3	0

Ainsi nous trouvons  $d(\mathbb{F}) = 2$ .

7. Il suffit d'utiliser le pivot de Gauss ; on trouve donc

$$G' = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

IL est important de noter que la matrice est unique, et que tout code linéaire admet une matrice systématique.

8. La matrice  $H'$  est égale à la transposée de  $G'$  (après la partie identité) et on rajoute la matrice identité pour compléter la dimension

$$H' = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

9. A partir de  $H$  on peut calculer la distance en calculant le nombre de colonnes dépendantes. Ici on les colonnes 1 et 4 qui dépendantes donc  $d(\mathbb{F}) = 2$

10. Ce n'est pas un code de Hamming car  $d(\text{code de Hamming}) \geq 3$ .

11. Soit  $Y_1 = 11111$ . Calculons  $S(Y_1) = H \cdot Y_1 = H' = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Le vecteur résultant est différent du vecteur nul alors on regarde la colonne correspondante au syndrome. Ici le problème se porte sur le troisième bit. Sachant que  $d(\mathbb{F}) < 3$ , il existe des erreurs qui ne peuvent être corrigées.

12. Avec  $Y_2 = 10011$ , en calculant le syndrome nous arrivons à  $S(Y_2) = H \cdot Y_2 = H' = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Son erreur est incorrigible.

- 13.

14.  $d(\mathbb{F}) = 2$  implique que l'on peut détecter toujours au moins une erreur et n'en corrige aucune.

Fin correction exercice 34

### Exercice 35 – Codes polynomiaux

On considère le code cyclique  $C(7, 4)$  engendré par le polynôme générateur  $g(x) = x^3 + x + 1$ .

1. Rappler le sens de  $\mathbb{C}(n, k)$ .

2. Montrer que le polynôme générateur est bien celui d'un code cyclique.

3. Ce code peut-il détecter (justifier chacune des réponses. On pourra utiliser des théorèmes connus en les citant) :

- (a) toutes les erreurs ?  
 (b) toutes les erreurs doubles ?  
 (c) toutes les erreurs de poids impair ?

4. Soit  $f(x) = x^3 + x$  un bloc à coder

- (a) Quel est le mot du code  $c(x)$  correspondant ? On considère le polynôme  $h(x)$  tel que  $g(x).h(x) = x^7 + 1$  On appelle syndrome d'un polynôme  $a(x)$  l'expression  $S(a(x)) = (h(x).a(x)) \pmod{x^7 + 1}$ .

- (b) Montrer que le syndrome de  $c(x)$  est nul.

5. Soit  $e(x) = x^2(x^2 + x + 1)$  un polynôme d'erreur.

- (a) Quel est le polynôme  $d(x)$  reçu pour  $c(x)$  transmis avec l'erreur  $e(x)$  ?  
 (b) Quel est le syndrome de  $d(x)$  ?  
 (c) L'erreur  $e(x)$  est-elle détectable ? Justifier.

6. Le code engendré par  $g(x)$  par multiplication polynomiale est-il systématique ? Pourquoi ?

7. Comment engendrer un code systématique à partir de  $g(x)$  ?

8. Quel est, dans ce cas, le mot de code  $c'(x)$  engendré pour le bloc à coder  $f(x)$  précédent ?

9. Peut-on détecter l'erreur  $e(x)$  précédente ? Si oui comment ? Justifier.

### Correction exercice 35

1. LE code  $\mathbb{C}(n, k)$  avec  $n$  la longueur du code et  $k$  la longueur de l'information utile. L'émetteur envoie le  $t(x) = f(x)g(x)$  où  $f(x)$  est le polynôme engendré par la code utile (par exemple :  $1011 \rightarrow 1 + x + x^3$ ). Donc le receveur reçoit le polynôme  $t'(x)$  et il divise  $t'(x)$  par  $g(x)$  et si le reste de la division est différent de zéro il en déduit une erreur de transmission.

2. Nous utilisons le théorème suivant :  $g(x)$  génère un code cyclique ssi  $g(x)$  divise  $x^n + 1$ . Ici  $n = 7$ , ainsi nous devons procéder à la division de  $x^7 + 1$  par  $x^3 + x + 1$ .

3. Sur la nombre de détéctions :

- (a) tout code polynomial a une distance supérieure strictement à 1. Les erreurs simples sont détectées. Nous avons  $t(x) = f(x)g(x) \rightarrow t'(x) = f(x)g(x) + e(x)$   $e(x)$  est détecté si  $e(x)$  n'est pas un multiple de  $g(x)$ .

- (b) Si il y a deux erreurs  $e(x) = x^i + x^j = x^i(1 + x^{j-i})$ . Nous devons regarder si  $(1 + x^l)$  est un multiple de  $g(x)$  pour  $d \leq l \leq n - 1$  avec  $d = \text{degré de } g(x)$ . Donc pour détecter deux erreurs il faut que ni  $1 + x^3$ , ni  $1 + x^4$ , ni  $1 + x^5$  ni  $1 + x^6$  ne soient des multiples de  $g(x)$ . Il suffit de vérifier.

(c) Peut-on détecter trois erreurs? C'est impossible car  $d(y, z) = 3$  avec

1011	111111 = $y$
1001	1100101 = $z$

Nous avons  $e(x) = x^2 + x^3 + x^5 = x^2(X^3 + x + 1)$

Pour résister à un nombre impair d'erreurs on met dans  $g$  un nombre pair de monômes.

-  $g(1) = 0 \Rightarrow g(x) = (1+x)g'(x)$

-  $e(1) = 1 \Rightarrow e(x)$  n'est pas un multiple de  $(1+x)$

- tout ceci est vrai si nombre pair d'erreurs

Donc  $e(x)$  ne peut pas être un multiple de  $g(x)$ .

4. Soit  $f(x) = x^3 + x$  un bloc à coder

(a)  $c(x) = f(x)g(x) = (x^3 + x)(x^3 + x + 1) = x^6 + x^4 + x^3 + x^4 + x^2 + x = x^6 + x^3 + x^2 + x$

(b) Donc

$$\begin{aligned} S(c(x)) &= h(x)c(x) \pmod{1+x^7} \\ &= (h(x)f(x)g(x)) \pmod{1+x^7} \\ &= (1+x^7)f(x) \pmod{1+x^7} \\ &= 0 \end{aligned}$$

5.  $d(x) = x^6 + x^4 + x$

6.  $s(d(x)) = x^4 + x^2 + x + 1$

7. L'erreur est détectable, son syndrome n'est pas nul.  $\mathbb{C}$  détecte les paquets d'erreurs de longueur trois.

8. C'est car la matrice  $G$  défini ci-dessous n'est pas diagonale gauche

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

9. On pose  $c'(x) = x^3 \cdot f(x) + x^3 \cdot (f(x) \pmod{g(x)})$

10.  $c'(x) = x^6 + x^4 + x + 1$

11. oui il suffit de calculer  $s(d'(x)) = x^4 + x^2 + x + 1$

Fin correction exercice 35

### Exercice 36 – Codes linéaires

On considère le code suivant :

$$C = \{00000000, 00011111, 11111000, 11100111\}$$

1. Calculer

(a) Sa longueur  $n$

(b) Sa dimension  $k$

(c) Sa redondance  $R$

(d) Sa distance  $D$

(e) Le nombre d'erreurs détectables  $Ed$

(f) Le nombre d'erreurs corrigibles  $Ec$

2. Montrer que ce code est linéaire. Construire une matrice génératrice  $G$

3. Montrer que ce code n'est pas systématique. Construire une matrice génératrice  $G$  d'un code systématique équivalent. Construire la matrice de vérification  $H$  correspondante

4. Le mot 11 est émis puis codé à partir du code systématique de matrice  $G$ ? Une malencontreuse série de 2 erreurs transforme ces 2 bits en 00 Ces erreurs sont-elles détectables? Justifier Ces erreurs sont-elles corrigibles? Sinon pourquoi? Si oui comment?

Correction exercice 36

1.

Fin correction exercice 36

### Exercice 37 – Sur les codes cycliques linéaires

Montrer tous les codes cycliques linéaires possibles de longueur 6. Indiquer pour chacun d'eux leurs caractéristiques essentielles : polynôme générateur, dimension, matrice de codage, distance, capacité de détectations d'erreurs. Quels sont, parmi ces codes, ceux permettant de détecter une erreur double sur 2 bits consécutifs, ceux permettant de détecter une erreur quadruple sur 4 bits consécutifs?

Mêmes questions avec des codes de longueur 8, puis 5.

Correction exercice 37

1.

Fin correction exercice 37

### Exercice 38 – Sur les codes cycliques linéaires

On cherche à construire des codes polynomiaux de redondance minimale pour une dimension supérieure à 2, pouvant détecter 2 erreurs indépendantes et des erreurs indépendantes en nombre impair

1. Expliquez comment construire ces codes et exprimer pour chacun d'eux son polynôme générateur et sa distance
2. Choisir un de ces codes et indiquer, en les justifiant, ses propriétés :
  - (a) Quelle longueur maximum d'un paquet d'erreur peut-il détecter ?
  - (b) Est-ce un code de Hamming ?
  - (c) Est-ce un code parfait ?
  - (d) Pour quelle longueur minimum est-il cyclique ?

---

Correction exercice 38

---

- 1.

---

Fin correction exercice 38

---

### Exercice 39 – Sur les codes cycliques polynomiaux

On considère les codes polynomiaux suivants :  $C_1(5,4)$  défini par le polynôme générateur :  $g_1(x) = x + 1$  et  $C_2(5,1)$  défini par le polynôme générateur :  $g_2(x) = x^4 + x^3 + x^2 + x + 1$

Caractériser ces deux codes

1.  $C_1$  est-il cyclique ? Pourquoi ?
2.  $C_2$  est-il cyclique ? Pourquoi ?
3. Existe-t-il d'autres codes cycliques de même longueur ? Lesquels ?
4. Pour chacun de ces 2 codes  $C_1$  et  $C_2$  :
  - (a) construire les matrices génératrices et de contrôle pour des codes systématiques équivalents. Justifier leur construction
  - (b) calculer leurs distances et capacités de détection et de correction

---

Correction exercice 39

---

- 1.

---

Fin correction exercice 39

---

### Exercice 40 – Sur les codes cycliques linéaires

Le code  $C(7,4)$  engendré par le polynôme  $g(x) = 1 + x^2 + x^3$

1. est-il équivalent à un code de Hamming ? Justifiez.

2. est-ce un code cyclique ? Justifiez.
3. On notera les polynômes et la matrice génératrice  $G$  de  $C$  avec les poids faibles en tête. On construira la matrice  $G'$  équivalente à  $G$  diagonale-gauche.

---

Correction exercice 40

---

- 1.

---

Fin correction exercice 40

---

### Exercice 41 – Sur les codes cycliques linéaires

On considère le code cyclique  $C(7,4)$  engendré par le polynôme générateur  $g(x) = x^3 + x^2 + 1$

1. Montrer que le polynôme générateur est bien celui d'un code cyclique.
2. Ce code peut-il détecter :
  - (a) toutes les erreurs simples ?
  - (b) toutes les erreurs doubles ?
  - (c) toutes les erreurs de poids impair ? Justifier chacune des réponses. On pourra utiliser des théorèmes connus en les citant.
3. Soit  $f(x) = x^3 + 1$  un bloc à coder
  - (a) Quel est le mot du code  $c(x)$  correspondant ?
  - (b) Calculer  $h(x)$  tel que :  $g(x) \times h(x) \pmod{x^7 + 1} = 0$
4. On appelle syndrome d'un polynôme  $a(x)$  l'expression :  $s(a(x)) = h(x) \times a(x) \pmod{x^7 + 1}$ . Calculer le syndrome de  $c(x)$
5. Soit  $e(x) = (x^5 + x^3)$  un polynôme d'erreur.
  - (a) Calculer le polynôme  $d(x)$  reçu pour  $c(x)$  transmis avec l'erreur  $e(x)$
  - (b) Calculer le syndrome de  $d(x)$
  - (c) L'erreur  $e(x)$  est-elle détectable ? Justifier

---

Correction exercice 41

---

- 1.

---

Fin correction exercice 41

---

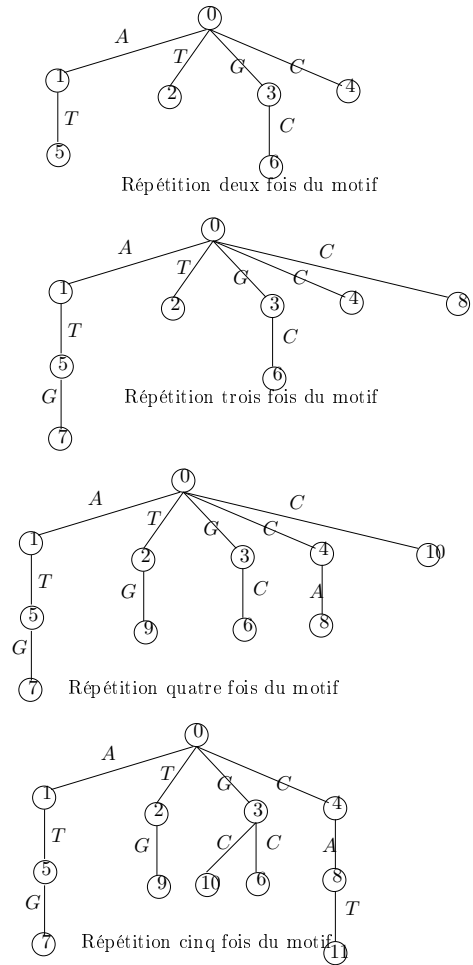


FIG. 4 – Evolution de l'arbre de décompression selon le nombre de répétition

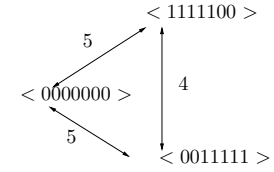


FIG. 5 – Calcul des distances

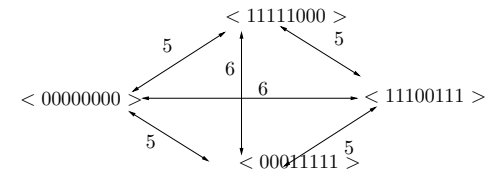


FIG. 6 – Calcul des distances