- TP 1. Calculs sur les rationnels. -

Le but de ce TP est de se constituer une petite bibliothèque sur les rationnels. Elle nous servira tout au long de notre implémentation du simplexe dont les calculs ne s'exprimeront ni en entier, ni en float pour des raisons d'arrondis.

Langage. Votre programme en C++ pourra commencer par :

```
#include <cstdlib>
#include <iostream>

using namespace std;

typedef struct {{ int p; int q;} rat;
```

Les rationnels seront donc des nombres de la forme a/b où a est un entier et b est un entier positif. Ecrire les fonctions suivantes :

rat convert(**int** a, **int** b) // Convertit un couple d'entiers en rationnels.

rat opposerat(**rat** z) // Retourne l'opposé de z.

rat inverserat(**rat** z) // Retourne l'inverse de z.

rat ratnull() // Retourne le rationnel nul.

bool negatif(**rat** z) // Teste si un rationnel est négatif.

bool negatifounul(**rat** z) // Teste si un rationnel est negatif ou nul.

bool pluspetit(**rat** x, **rat** y) // Teste si x est plus petit que y.

bool pluspetitouegal(**rat** x, **rat** y) // Teste si x est plus petit que y.

rat produitrat(**rat** u, **rat** v) // Retourne le produit de u et v.

rat sommerat(**rat** a, **rat** b) // Retourne la somme de a et b.

void simplify(**rat** *z) // Réduit la fraction (Attention ! Utiliser un calcul de pgcd pas trop lent...).

void afficherat(**rat** z) // Affiche le rationnel z. Par exemple 2/3 ou -2 ou encore -11/25.

void afficheratsigne(**rat** z) // Affiche z avec son signe. Par exemple +2/3 ou -2 ou encore -11/25.

- TP 2. Affichage pour le simplexe. -

Le but de ce TP est d'afficher les programmes linéaires et les dictionnaires que le simplexe va engendrer. Dans un premier temps, on se contentera d'un affichage console. Pour le lecteur exigeant, ne pas hésiter à utiliser Latex, avec un exemple proposé sur la page :

`http://www.lirmm.fr/~bessy/PL/accueil.html` (section Documents)

Rappelons que le programme linéaire (P) que l'on veut résoudre est de la forme :

$$\begin{array}{ll} \text{Maximiser} & c_1x_1 + \cdots + c_nx_n \\ \text{Sous} & \sum_{j=1}^{n} a_{ij}x_j \leq b_i \qquad i = 1, \ldots, m \\ & x_1, \ldots, x_n \geq 0 \end{array}$$

Tous les coefficients a_{ij}, b_i et c_j sont des rationnels.

Le codage de l'entrée se fait par deux tableaux **int** objectif$[2*n]$ et **int** contraintes$[m][2*n+2]$. La correspondance se fait ainsi :

- Le coefficient c_i est égal à objectif$[2*i-2]$/objectif$[2*i-1]$.
- Le coefficient a_{ij} est égal à contraintes$[i-1][2*j-2]$/contraintes$[i-1][2*j-1]$.
- Le coefficient b_i est égal à contraintes$[i-1][2*n]$/contraintes$[i-1][2*n+1]$.

- Exercice 1 - Affichage du programme linéaire.

Ecrire une fonction **void** afficheprogramme(**int** contraintes[m][2*n+2], **int** objectif[2*n]) qui affiche le programme linéaire.

Par exemple, pour les entrées :

int objectif$[6] = \{2, 3, 3, 1, -1, 5\}$;

int contraintes$[2][8] = \{\{2, 1, 0, 1, 6, 1, 2, 3\}, \{7, 1, 8, 1, -2, 1, 1, 4\}\}$;

l'affichage sera de la forme :

```
Maximiser  2/3x_1   +3x_2 -1/5x_3
Sous         2x_1           +6x_3    <= 2/3
             7x_1   +8x_2   -2x_3    <= 1/4
              x_1,x_2,x_3 positifs ou nuls
```

- Exercice 2 - Dictionnaires.

L'algorithme du simplexe manipule des dictionnaires de la forme :

$$\begin{array}{rrrr} x_3 = 2 & +x_4 & -x_2 & +5/2x_5 \\ x_1 = 3 & & -2/3x_2 & -1/3x_5 \\ \hline z = 4 & -x_4 & -2x_2 & +3x_5 \end{array}$$

Ils sont représentés par :

- Un tableau de rationnels **rat** dico$[m+1][n+1]$, ici **rat** dico[3][4] dont par exemple l'entrée dico[0][3] est le rationnel 5/2 et l'entrée dico[2][0] est le rationnel 4.
- Un tableau **int** variables$[n+m]$ qui énumère tout d'abord les indices des variables non basiques du dictionnaire (celles horizontales, ici x_4, x_2, x_5), et ensuite les variables basiques (celles verticales, ici x_3, x_1). Dans l'exemple, on aura variables$[5] = \{4, 2, 5, 3, 1\}$.

Ecrire une fonction d'affichage **void** affichedico(**rat** dico$[m+1][n+1]$,**int** variables$[m+n]$).

- TP 3. L'algorithme du simplexe en une phase. -

Le but de ce TP est d'implémenter la première phase de l'algorithme du simplexe.

L'algorithme du simplexe transforme le programme (P) en un dictionnaire initial, sur lequel des pivots sont itérés.

- Exercice 3 - Dictionnaire initial.
Ecrire une fonction

void initialisedico(**int** contraintes$[m][2*n+2]$, **int** objectif$[2*n]$, **rat** dico$[m+1][n+1]$, **int** variables$[n+m]$)

qui initialise les tableaux dico et variables représentant le dictionnaire initial du programme linéaire (P) donné par les tableaux contraintes et objectif.

- Exercice 4 - Pivot entrant.
Ecrire une fonction **int** pivotentrant(**rat** dico$[m+1][n+1]$) qui retourne le numéro de colonne entre 1 et n qui correspond à une variable entrante. Si aucune variable ne convient, pivotentrant retourne 0.

- Exercice 5 - Pivot sortant.
Ecrire une fonction **int** pivotsortant(**rat** dico$[m+1][n+1]$, **int** i) qui retourne le numéro de ligne entre 1 et m qui correspond à une variable sortante, et ceci lorsque i est la colonne de la variable entrante. Si aucune variable sortante ne convient, pivotsortant retourne 0.

- Exercice 6 - Pivoter.
Ecrire une fonction **void** pivoter(**rat** dico$[m+1][n+1]$, **int** variables$[n+m]$, **int** sort, **int** entr) qui effectue un pivot de simplexe sur le dictionnaire pour la variable entrante entr et la variable sortante sort.

- Exercice 7 - Simplexe.
Achever d'implémenter l'algorithme du simplexe en une phase. Votre algorithme devra retourner :

- soit une solution admissible optimale.
- soit le message : La fonction objectif n'est pas bornée.
- soit le message : Le dictionnaire initial de (P) n'est pas admissible.

- TP 4. Calculs avec le simplexe. Choix de pivot. Cyclage -

Si tout s'est bien passé aux TP précédents, vous disposez à présent d'un algorithme dont la sortie est, par exemple :

```
Le programme primal est :

Maximiser 5x_1 +4x_2 +3x_3
Sous      2x_1 +3x_2 +1x_3 <= 5
          4x_1 +1x_2 +2x_3 <= 11
          3x_1 +4x_2 +2x_3 <= 8
          x_1, x_2, x_3 positifs ou nuls

Le dictionnaire initial est :

x_4 = 5 -2x_1 -3x_2 -1x_3
x_5 = 11 -4x_1 -1x_2 -2x_3
x_6 = 8 -3x_1 -4x_2 -2x_3
----------------------------------------
z   = 0 +5x_1 +4x_2 +3x_3

La variable entrante est x_1
La variable sortante est x_4

x_1 = 5/2 -1/2x_4 -3/2x_2 -1/2x_3
x_5 = 1 +2x_4 +5x_2 +0x_3
x_6 = 1/2 +3/2x_4 +1/2x_2 -1/2x_3
----------------------------------------
z   = 25/2 -5/2x_4 -7/2x_2 +1/2x_3

La variable entrante est x_3
La variable sortante est x_6

x_1 = 2 -2x_4 -2x_2 +1x_6
x_5 = 1 +2x_4 +5x_2 +0x_6
x_3 = 1 +3x_4 +1x_2 -2x_6
----------------------------------------
z   = 13 -1x_4 -3x_2 -1x_6

La solution optimale est : x_1= 2 x_2= 0 x_3= 1
La valeur de la fonction objectif en cette solution est : 13
Le nombre de pivots effectues est : 2
```

Le but de ce TP est d'effectuer quelques calculs avec votre algorithme, ainsi que d'en définir les limites.

- Exercice 8 - Application du simplexe.
Appliquer votre algorithme :

1. Aux programmes linéaires des exercices 1,2 et 3 de la fiche de TD.
2. À des programmes linéaires à n variables et n contraintes dont les coefficients sont tirés au hasard entre 1 et 10. Jusqu'à quelles valeurs de n votre algorithme vous permet-il de résoudre un programme linéaire ?
3. Votre algorithme est-il plus sensible au nombre n de variables ou au nombre m de contraintes ? Faire des simulations.

- Exercice 9 - Règle de Bland.
Ecrire une fonction **int** pivotentrantbland(**rat** dico$[m+1][n+1]$,**int** variables$[m+n]$) et **int** pivotsortantbland(**rat** dico$[m+1][n+1]$, **int** i,**int** variables$[m+n]$) qui choisit les pivots entrants et sortants de plus petit indice lorsqu'il y a plusieurs choix possibles.

- Exercice 10 - Un cas de cyclage.
Ecrire une fonction **int** pivotentrantmaxcoef(**rat** dico$[m+1][n+1]$) qui retourne, parmi les choix possibles de pivots entrants, celui qui a un coefficient maximum dans la fonction objectif.

Utiliser les fonctions pivotentrantmaxcoef et pivotsortantbland afin de résoudre l'exercice 43 de la fiche de TD.

- Exercice 11 - Les exemples de Klee-Minty.
Résoudre le problème linéaire suivant en utilisant l'algorithme du simplexe. Le choix de la variable entrante se fera avec pivotentrantmaxcoef.

$$\begin{array}{lrrrl} \text{Maximiser} & 100x_1 & +10x_2 & +x_3 & \\ \text{Sous} & x_1 & & & \leq 1 \\ & 20x_1 & +x_2 & & \leq 100 \\ & 200x_1 & +20x_2 & +x_3 & \leq 10000 \\ & & & x_1, x_2, x_3 & \geq 0 \end{array}$$

Généraliser cet exemple pour n variables. Quel est le nombre $f(n)$ de pivots effectués ?

- TP 5. Le problème d'affectation. -

Le but de ce TP est de résoudre le problème d'affectation (Cf cours, étude de cas 6) de t agents à t tâches. On se donne un tableau de rationnels assign$[t][t]$ dont chaque entrée assign$[i][j]$ représente la performance p_{ij} de l'agent i à réaliser la tâche j. Le problème est d'assigner une tâche à chaque agent afin de maximiser la somme des performances des agents à réaliser leurs tâches respectives.

La modélisation de ce problème se fait ainsi : Les variables de décision sont les x_{ij} qui représentent la part de la tâche j réalisée par l'agent i. La fonction à maximiser est la somme des $p_{ij}x_{ij}$. Les contraintes sont :

- Pour chaque agent i fixé, la somme des x_{ij} est au plus 1.
- Pour chaque tâche j fixée, la somme des x_{ij} est au plus 1.

Il y a donc $n = t^2$ variables de décision et $m = 2t$ contraintes, parmi lesquelles t proviennent des agents et t autres proviennent des tâches. Les variables d'écart provenant des agents seront notées $a_1, \ldots, a_t$ et celles provenant des tâches seront notées $t_1, \ldots, t_t$.

Le but est d'utiliser l'algorithme du simplexe en une phase afin de résoudre ce problème. Illustrons sur le cas (facile !) suivant :

performance	tâche 1	tâche 2
agent 1	3	2
agent 2	3	5

On voit clairement que la performance maximale sera obtenue en assignant l'agent 1 à la tâche 1 et l'agent 2 à la tâche 2. Mais on va faire le calcul en utilisant le simplexe. La principale difficulté est d'initialiser le dictionnaire correspondant au programme linéaire. On veut obtenir :

```
Le dictionnaire initial est :
 a_1 =  1  -x_11   -x_12
 a_2 =  1                  -x_21   -x_22
 t_1 =  1  -x_11           -x_21
 t_2 =  1          -x_12           -x_22
   z =  0  +3x_11  +2x_12  +3x_21  +5x_22
```

Ceci se fait par l'intermédiaire d'une fonction **void** dicoinitassign(rat assign$[t][t]$, rat dico$[m+1][n+1]$) qui initialise dico comme dictionnaire initial du problème d'assignation. L'affichage des variables se fera avec une fonction **void** affichevar(**int** i) qui au lieu d'afficher simplement `x_i` affichera `x_11, x_12, x_21, x_22, a_1, a_2, t_1, t_2`, pour i variant de 1 à $n + m$.

La suite ne pose pas de problème, il suffit d'exécuter le simplexe.

```
La variable entrante est x_11.
La variable sortante est a_1.
 x_11 =  1  -a_1   -x_12
  a_2 =  1                 -x_21   -x_22
  t_1 =  0  +a_1   +x_12  -x_21
  t_2 =  1         -x_12          -x_22
    z =  3  -3a_1  -x_12  +3x_21  +5x_22
La variable entrante est x_22.
La variable sortante est a_2.
```

```
 x_11 =  1  -a_1   -x_12
 x_22 =  1                 -x_21   -a_2
  t_1 =  0  +a_1   +x_12   -x_21
  t_2 =  0         -x_12   +x_21   +a_2
    z =  8  -3a_1  -x_12   -2x_21  -5a_2
```

```
La solution optimale est : x_11= 1 x_12= 0 x_21= 0 x_22= 1.
La valeur de la fonction objectif en cette solution est : 8.
Le nombre de pivots effectues est : 2.
```

Vous pourrez modifier la conclusion en :

```
L'assignation est 11, 22, qui réalise une performance de 8.
Le certificat d'optimalité est obtenu en affectant les poids suivants sur les agents
et les tâches: a_1=3, a_2=5, t_1=0, t_2=0.
```

Une fois votre algorithme réalisé, testez-le sur des valeurs aléatoires de assign$[t][t]$. Quelle taille t de problèmes peut-on atteindre ? Comparer avec le simplexe sur des programmes aléatoires.

- TP 6. Quelques directions. -

- Exercice 12 - Exemple d'affectation.
Résoudre les questions c. et d. de l'exercice 5 de la fiche de TD.

- Exercice 13 - Le cas non borné.
Lorsque votre algorithme ne trouve pas de candidat pour le pivot sortant, le problème est non borné. Plutot qu'un simple message `Problème non borné`, retourner un ensemble de solutions dépendant d'un paramêtre t qui certifie bien que le problème est non borné. Par exemple dans le programme suivant

$$\begin{array}{lrrl} \text{Maximiser} & x_1 & +x_2 & \\ \text{Sous} & x_1 & -x_2 & \leq 1 \\ & -x_1 & +x_2 & \leq 1 \\ & & x_1, x_2 & \geq 0 \end{array}$$

La solution pourrait être :

`Problème non borné lorsque` $x_1 = t$ `et` $x_2 = t$`, avec` $t >= 0$

- Exercice 14 - Certificat d'optimalité.
Lorsque votre algorithme du simplexe trouve une solution optimale, vous pouvez justifier son optimalité grace au dual, ou ce qui est équivalent par une combinaison linéaire de contraintes. En plus de la solution optimale, retourner cette combinaison. Par exemple :

$$\begin{array}{lrrl} \text{Maximiser} & x_1 & +x_2 & \\ \text{Sous} & 2x_1 & +x_2 & \leq 3 \\ & x_1 & +2x_2 & \leq 2 \\ & & x_1, x_2 & \geq 0 \end{array}$$

La solution pourrait être :

`La solution optimale est` $x_1 = 4/3$, $x_2 = 1/3$ `pour une valeur de 5/3.`
`Le certificat d'optimalité est :` $1/3$ `contrainte 1 +` $1/3$ `contrainte 2, pour un résultat de` $x_1 + x_2 \leq 5/3$

- Exercice 15 - Réseau de transport.
Modifier votre algorithme du TP3 sur le problème d'affectation afin de résoudre les problèmes de réseaux de transport.

- Exercice 16 - Simplexe en deux phases.
Coder le simplexe en deux phases.