Algorithmique (AMD5)
Devoir maison : tournoi de multibôle

Ce devoir maison est à réaliser en **binôme** ou **seul** et à rendre en TD, à votre chargé de TD, la semaine du **20 novembre**.
Les algorithmes du cours que vous utiliserez devront être rappelés (donnés avec leur code). Par contre vous n'aurez pas besoin de prouver de nouveau leur correction et leur complexité. Il vous suffira, le cas échéant, de rappeler leur complexité.
Pour tout nouvel algorithme que vous présenterez, il faudra justifier sa **correction** et sa **complexité**.
Vous pourrez utiliser des structures de données usuelles, mais dans ce cas, il faudra être explicite sur les fonctions usuelles d'accès à ses structures que vous invoquerez, c'est-à-dire les nommer et préciser leur rôle.

Présentation du problème

Le multibôle est un jeu où plusieurs joueurs s'affrontent. Tandis que les sports habituels (tennis, boxe...) sont des duels entre deux joueurs, avec un vainqueur et un vaincu, le multibôle se joue avec $k \geq 2$ joueurs. À l'issue d'un match, on obtient un classement, avec un premier, un deuxième, etc. jusqu'au $k$ième.
De plus, un match de multibôle dure très peu de temps. Ainsi, dans un tournoi, on peut jouer de grandes quantités de matchs. S'il y a n joueurs en tout, on va essayer de jouer r fois tous les matchs possibles de k joueurs. Par exemple pour $k = 3$, $n = 4$ (les joueurs s'appellent A, B, C et D) et $r = 2$, on joue 8 matchs et on peut obtenir les classements suivants :

$$ABC, ADB, CDA, BCD, BAC, ABD, CAD, BCD \tag{1}$$

On a donc, pour un tournoi complet avec r fois chaque match, $m = r\binom{n}{k} = r\frac{n!}{k!(n-k)!}$ matchs.

- k nombre de joueurs qui s'affrontent dans un match
- n nombre de participants à un tournoi. $n \geq k$.
- r nombre de matchs identiques joués (qui ont les même joueurs)
- m nombre total de matchs d'un tournoi. $m = r\binom{n}{k} = r\frac{n!}{k!(n-k)!}$

Résumé des paramètres

Le but de ce DM est de réfléchir à comment calculer le vainqueur d'un tournoi de multibôle.

Notons $w(A, B)$ le nombre de matchs où A a été classé devant B. On dit que le joueur A *domine* le joueur B si $w(A, B) \geq w(B, A)$. Un joueur qui domine tous les autres est dit un *vainqueur absolu*.

Le *graphe de tournoi* est un graphe complet orienté (il y a donc un arc dans chaque sens entre tout couple de sommets. Mais pas de boucle). Le long d'un arc $A \to B$ on note le nombre $w(A, B)$. Dans le *graphe de tournoi réduit*, on ne laisse qu'au plus un seul arc entre A et B : celui qui a la plus forte valeur (donc si $w(A, B) > w(B, A)$ il y a un seul arc, de A à B). En cas d'égalité (même valeur dans chaque sens), on enlève les deux arcs.

Exercice 1 : vainqueur absolu

1. Construire le graphe de tournoi, puis le graphe réduit pour le tournoi en 8 matchs et les classements décrits en (1) page précédente. Est-ce qu'on a un vainqueur absolu ?
2. En changeant le résultat du moins de matchs possibles dans (1), faire en sorte que le tournoi
   a) n'a pas de vainqueur absolu
   b) a plusieurs vainqueurs absolus
3. Comment voit-on si un joueur est un vainqueur absolu sur le graphe de tournoi réduit ?
4. Que vaut la somme des valeurs sur l'arc $A \to B$ plus celle sur l'arc $B \to A$ sur le graphe non réduit en fonction de n, k et r ?
5. Proposer un algorithme qui prend comme paramètre le graphe réduit et renvoie la liste des vainqueurs absolus (qui peut être vide). Quelle est la complexité de votre algorithme en fonction du nombre de joueurs et du nombre d'arcs du graphe réduit, s'il est donné sous la forme d'une liste d'adjacence ? Comparer à la complexité quand il est donné sous la forme d'une matrice d'adjacence.

Exercice 2 : algorithme – cas des graphes réduits acycliques

On suppose, dans cet exercice seulement, que le graphe de tournoi réduit G obtenu est acyclique.

1. Montrer que dans ce cas il existe au moins un vainqueur absolu.
2. Dans ce cas on définit le résultat du tournoi comme une fonction `rang` sur l'ensemble des joueurs :

$$\texttt{rang(A)} = \begin{cases} 1, & \text{si } A \text{ est un vainqueur absolu,} \\ 1 + \max\{\texttt{rang(B)} \mid \texttt{B} \to \texttt{A} \text{ est un arc de } \texttt{G}\}, & \text{sinon .} \end{cases}$$

   Proposer un algorithme le plus efficace possible qui calcule le rang des joueurs. Prouver sa correction, donner sa complexité et justifier.

La *force d'un chemin* entre deux joueurs A et B, pour un chemin $A = C_1 \to C_2 \to ... \to C_{l-1} \to C_l = B$ de $l-1$ arcs ($l \geq 2$) du graphe de tournoi réduit, est la valeur minimum apparaissant sur un arc. Par exemple la force du chemin $A \xrightarrow{5} C_2 \xrightarrow{7} C_3 \xrightarrow{3} C_4 \xrightarrow{9} B$ est 3.

La *force de A sur B* est la plus grande force possible pour un chemin de A à B. Par exemple, s'il y a 3 chemins entre A et B, le premier de force 3, le deuxième de force 8, et le troisième de force 5, alors la force de A sur B est 8. S'il n'y a pas de chemin de A à B, on dit alors que la force de A sur B est 0. On dira que *A est plus fort que B* si la force de A sur B est *supérieure ou égale* à celle de B sur A, et que *A est strictement plus fort que B* si la force de A sur B est *supérieure* (mais pas égale) à celle de B sur A,
Un *vainqueur par force* est un joueur V tel que V est plus fort que tous les autres joueurs.

Exercice 3 : exemple sans vainqueur absolu

Soit $n = 4$, $k = 3$ et $r = 5$. On nomme A, B, C et D les joueurs et après le tournoi de multibôle, on obtient les classements suivants :

ABC ACB BAC CAB CAB	ADB ADB BDA BDA DBA
CAD DAC DAC DAC DAC	BCD BCD BCD CDB CDB

Construire le graphe de tournoi, puis le graphe réduit. Montrer qu'il n'y a pas de vainqueur absolu. Déterminer le(s) vainqueur(s) par force en calculant toutes les forces.

Exercice 4 : propriétés de la force

1. Montrer qu'un vainqueur absolu est vainqueur par force.
2. Montrer que si A est strictement plus fort que B et B est strictement plus fort que C, alors A est strictement plus fort que C.
3. Montrer que dans tout graphe de tournoi réduit, on a au moins un vainqueur par force.

Exercice 5 : algorithmique de la force

1. Proposer un algorithme qui calcule la force entre chaque paire de joueurs, étant donné le graphe de tournoi réduit. Prouver sa correction, donner sa complexité et justifier.
2. Proposer un algorithme qui calcule les vainqueurs par force. Prouver sa correction, donner sa complexité et justifier.
3. Proposer un algorithme, le plus efficace possible, qui calcule un ordre total $\prec$ tel que, si A est strictement plus fort que B, alors $A \prec B$. Prouver sa correction, donner sa complexité et justifier.