DS d'informatique N°2

Quelques questions demandent d'écrire du code python, vous veillerez autant que possible à utiliser une syntaxe valide et à indenter correctement le code (utilisez des lignes verticales pour marquer les différents niveaux d'indentation)

Tournois et pronostics X–PSI/PT–2014

Cette épreuve traite des tournois à élimination directe tels qu'on en rencontre en sport. La première partie concerne leurs représentations. La deuxième partie consiste à exhiber les vainqueurs possibles d'un tournoi si l'on connaît à l'avance les résultats de tous les matchs possibles.

La complexité, ou le temps d'exécution, d'un programme P (fonction ou procédure) est le nombre d'opérations élémentaires (addition, multiplication, affectation, test, etc...) lors de l'exécution de P. Lorsque cette complexité dépend d'un paramètre n, on dira que P a une complexité en $O(f(n))$, s'il existe $K > 0$ tel que la complexité de P est au plus $K\,f(n)$, pour tout n.

De manière générale, on s'attachera à garantir une complexité aussi petite que possible dans les fonctions écrites. Le candidat y sera tout particulièrement sensible lorsque la complexité d'une fonction est demandée. Dans ce cas, il devra de plus justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Dans tout le sujet, on considère que le nombre de joueurs est fixé à n, qui est une variable globale à laquelle tous les programmes peuvent accéder, sans avoir à la définir ni à la passer en argument. On appelle vecteurs les listes à une dimension, et matrices celles à deux dimensions.

Partie I : Tournois

Dans cette partie, on s'intéresse à la représentation et à la manipulation de tournois à élimination directe entre n joueurs. Les joueurs sont représentés par les entiers de 0 à $n-1$. Un tournoi entre ces n joueurs est donc défini par une suite de $n-1$ *matchs*. Chaque match oppose deux joueurs, qui sont désignés soit nommément (dans la figure 1, le match "A" oppose le joueur 4 au joueur 1 ; le match "B" oppose le joueur 2 au joueur 0), soit en fonction du résultat d'un match antérieur (dans la figure 1, le match "C" oppose le vainqueur du match "B" au joueur 3 ; le match "D" (oppose le vainqueur du match "A" au vainqueur du match "C" ; le match "E" oppose le vainqueur du match "D" au joueur 5).

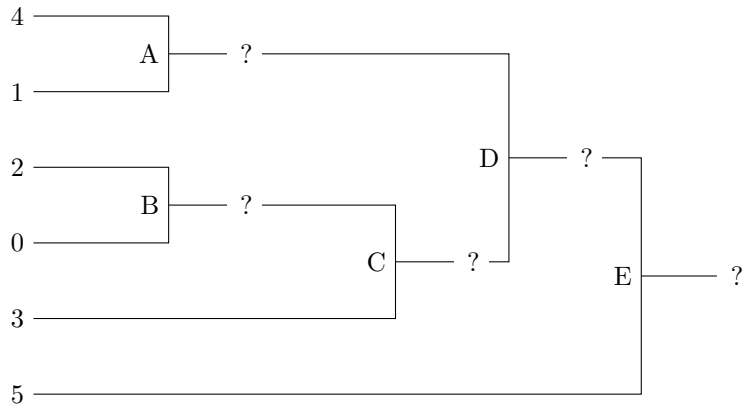

Figure 1 – Un tournoi à 6 joueurs

Afin de représenter un tournoi en machine, on va identifier les matchs avec les entiers n, $n+1$, ..., $2n-2$ plutôt qu'avec des lettres comme dans l'exemple précédent. Ainsi dans l'exemple ci-dessus le match précédemment appelé "A" sera le match 6, le match "B" sera le match 7, le match "C" sera le match 8, le match "D" sera le match 9 et le match "E" sera le match 10.

Pour représenter un match on va utiliser un tableau de taille deux dont les éléments sont les protagonistes (joueurs ou match) du match : s'il s'agit d'un joueur on donnera son indice et s'il s'agit du vainqueur d'un match précédent on donnera l'indice du match.

Pour avoir unicité de la représentation d'un match, on adoptera systématiquement la **convention** qu'un match $[i, j]$ est tel que $i \leq j$. Ainsi le match 6 (précédemment appelé "A") et le match 9 (précédemment appelé "D") sont représentés par les tableaux $[1, 4]$ et $[6, 8]$.

Par la suite, on identifiera pour tout entier $0 \leq i < n$, le joueur i et le match $[i, i]$. Cela permet alors de représenter un tournoi par un tableau de $2n-1$ matchs de la façon suivante. Les n premiers matchs sont les matchs $[0, 0]$,...,$[n-1, n-1]$, les $(n-1)$ matchs suivants étant les matchs réels. Par exemple, le tournoi précédent pourra être représenté par le tableau suivant :

$$\big[[0,0],[1,1],[2,2],[3,3],[4,4],[5,5],[1,4],[0,2],[3,7],[6,8],[5,9]\big]$$

Bien sûr toute matrice d'entiers T, de taille $(2n-1)\times 2$ ne code pas un tournoi. En fait, un tournoi est codé par une matrice d'entiers T de taille $(2n-1)\times 2$ si et seulement si :

1. pour tout $0 \leq k < 2n-1$, $T[k][0] \leq T[k][1]$;
2. pour tout $0 \leq k < n$, $T[k][0] = T[k][1] = k$;
3. pour tout $n \leq k < 2n-1$, $T[k][0] \neq T[k][1]$;
4. pour tout $n \leq k < 2n-1$, $0 \leq T[k][0] < k$ et $0 \leq T[k][1] < k$;
5. pour tout $0 \leq k < 2n-2$, il existe un unique $k' > k$ tel que $k \in \{T[k'][0], T[k'][1]\}$

Question 1

Donner une interprétation en français des conditions (3), (4) et (5) exprimées ci-dessus pour qu'une matrice T soit un tournoi. Par exemple la condition (2) s'interprète comme "*les n premiers éléments de la représentation d'un tournoi sont les matchs triviaux identifiés aux joueurs* ". La condition (1) correspond à la convention que le premier protagoniste d'un match est, celui de plus petit indice.

Question 2

Écrire une fonction `EstVraiCondition5`(T,k) qui prend en argument une matrice d'entiers de taille $(2n-1)\times 2$ et un entier k tel que $0 \leq k < 2n-2$ et qui renvoie `True` s'il existe un unique $k' > k$ tel que $k \in \{T[k'][0], T[k'][1]\}$ et `False` sinon. Quelle est sa complexité ? *On supposera, sans avoir à le vérifier, que le matrice T est de taille $(2n-1)\times 2$.*

Question 3

Écrire une fonction `EstUnTournoi`(T) qui prend en argument une matrice d'entiers et qui renvoie `True` si T est un tournoi, et `False` sinon. Quelle est sa complexité ?
On supposera, sans avoir à le vérifier, que le matrice T est de taille $(2n-1)\times 2$.

On s'intéresse maintenant à la manière dont se déroule le tournoi, à l'aide de matrices, appelées *oracles*, qui prédisent le résultat de chaque match possible. Un oracle O est une matrice de booléens, de taille $n \times n$ qui représente les résultats de tous les matchs possibles : $O[i][j]$ contient `True` si i remporte les matchs joués entre i et j, et `False` sinon (il s'ensuit que, pour tous i et j distincts, $O[i][j]$ contient `True` si et seulement si $O[j][i]$ contient `False`). Par convention, on supposera que les cases de la diagonale contiennent `True`.

i \ j	0	1	2	3	4	5
0	True	True	False	True	True	False
1	False	True	True	True	False	True
2	True	False	True	True	False	True
3	False	False	False	True	False	False
4	False	True	True	True	True	False
5	True	False	False	True	True	True

FIGURE 2 – Exemple d'un oracle.

Question 4

Écrire une fonction `EstUnOracle`(O) qui prend en argument une matrice de booléens et qui renvoie `True` si O est un oracle, et `False` sinon. Quelle est sa complexité ?
On supposera que la matrice O est de taille $n \times n$.

La donnée d'un tournoi et d'un oracle détermine la manière dont se déroule le tournoi. Dans notre exemple, on obtient le résultat décrit dans la figure 3.

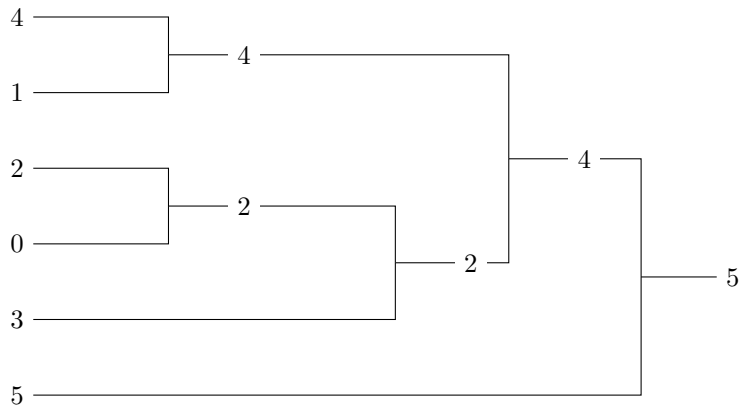

FIGURE 3 – Déroulement du tournoi de la figure 1 en fonction de l'oracle de la figure 2

Question 5

Écrire une fonction `Vainqueur`(T, O) qui prend en argument un tournoi et un oracle, et qui renvoie le vainqueur du tournoi T selon l'oracle O. Quelle est sa complexité ?
On supposera que T est un tournoi et que O est un oracle.

Partie II. Vainqueurs Potentiels

Nous avons vu dans la Partie I qu'un tournoi T et un oracle O définissaient un vainqueur. Étant donné un oracle O, le vainqueur d'un tournoi peut varier en fonction du tournoi T considéré. Nous cherchons dans cette partie à identifier tous les vainqueurs possibles lorsque T varie, mais que O est fixe. On appellera ces vainqueurs les *vainqueurs potentiels* de O.

Question 6

Donner un oracle qui n'a qu'un seul vainqueur potentiel. Donner un oracle qui a plusieurs vainqueurs potentiels.

Pour calculer efficacement tous les vainqueurs potentiels de O nous allons partir d'un premier vainqueur, pour en déduire d'autres, jusqu'à ce qu'on les ait tous trouvés.

Question 7

Pour déterminer un vainqueur possible pour un oracle donné, il suffit de trouver le vainqueur d'un tournoi choisi arbitrairement. Écrire une fonction `UnVainqueur`(O) qui prend en argument un oracle O et qui renvoie un vainqueur potentiel de O. Quelle est sa complexité ? *On supposera que O est un oracle.*

On peut démontrer les deux propriétés suivantes :

— Si i est un vainqueur potentiel de O, et si O prédit que j remporte le match entre i et j, alors j est un vainqueur potentiel de O.

— Si un ensemble X non-vide de joueurs est tel que O prédit que chaque joueur de X gagne tous ses matchs contre les joueurs hors de X, alors tous les vainqueurs potentiels de O sont dans X.

Question 8

En utilisant les questions 7 et les deux propriétés précédentes, écrire la fonction `vainqueursPotentiels`(O) qui prend en argument un oracle O et qui renvoie un vecteur v de taille n tel que $v[i] =$ `True` si i est un vainqueur potentiel de O, et $v[i] =$ `False` sinon. Quelle est sa complexité ? *On supposera que O est un oracle.*

Indication : On pourra initialiser une pile avec un premier vainqueur potentiel puis tant que la pile n'est pas vide effectuer les opérations suivantes :

— Dépiler le premier joueur de la pile.

— Parmi tous les joueurs qui ne sont pas dans la liste des vainqueurs potentiels, empiler ceux qui gagnent un match contre lui et les ajouter à la liste des vainqueurs potentiels.

Partie III. Pronostics

Dans cette partie, les oracles sont remplacés par des *pronostics*, qui donnent des chances (ou probabilités) de victoire plutôt que des certitudes. L'objectif de cette partie est de calculer la probabilité qu'a chaque joueur de remporter le tournoi.

La première difficulté réside dans le fait que l'issue d'un match ne peut plus être déterminée avec certitude, et que donc plusieurs joueurs peuvent potentiellement participer à un même match. Il faut donc pouvoir calculer, pour chaque match, l'ensemble des joueurs susceptibles d'y participer.

Question 9

Écrire une fonction `PremierJoueurPossible`(T, k) qui prend en argument un tournoi T et un entier k tel que $0 \leqslant k < 2n-1$, et qui renvoie un vecteur de booléens L de taille n tel que $L[i] =$ `True` si le joueur i peut être le premier joueur du match d'indice k du tournoi T, et $L[i] =$ `False` sinon. *On supposera que T est un tournoi et que k est un entier compris entre 0 et $2n-2$.*

Indication : On pourra à nouveau utiliser une pile pour déterminer tous les joueurs qui peuvent être le premier joueur d'un match donné. Les premiers joueurs possibles d'un match sont tous les joueurs qui peuvent accéder au match précédent. On peut donc empiler le numéro dont le gagnant est le premier joueur du match k et tant que la pile n'est pas vide faire :

— On dépile un match m de la pile

— Si m est un match trivial ($m < n$) ajouter le joueur m à la liste des joueurs possibles

— Sinon empiler `T[m][0]` et `T[m][1]` les deux matchs qui amènent au match m.

Nous considérons maintenant, que pour chaque match possible, les chances de victoire de chaque joueur sont données par des pronostics. Un *pronostic* P est une matrice de réels dans $[0, 1]$, de taille $n \times n$, et telle que $P[i][j]$ est la probabilité que i remporte un match contre j (par exemple $P[i][j] = 0.75$ s'interprète comme "le joueur i a 3 chances sur 4 de battre le joueur j"). Il s'ensuit que $P[i][j] + P[j][i] = 1$, pour tout i et j.

Le but de la fin de cette partie est de calculer, à tournoi T et pronostic P donnés, les chances de remporter le tournoi pour chaque joueur.

L'approche proposée consiste à maintenir un vecteur (de taille n) V de probabilités représentant les chances de chaque joueur d'être encore présent à un moment donné du tournoi : avant le match n (le premier vrai match), le vecteur V ne contiendra que des 1 ; après le dernier, il devra contenir les chances de victoire de chaque joueur.

Si V est le vecteur de probabilités courantes avant le match k, alors le vecteur V' de probabilités après le match est défini comme suit :

— Si i est un premier joueur possible : $V'[i] = V[i] \times \sum_{j \text{ est un second joueur possible}} (V[j] \times P[i][j])$

— Si i est un second joueur possible : $V'[i] = V[i] \times \sum_{j \text{ est un premier joueur possible}} (V[j] \times P[i][j])$

— Sinon : $V'[i] = V[i]$.

Question 10

En utilisant les fonctions `PremierJoueurPossible` et une fonction analogue `SecondJoueurPossible` (que l'on s'abstiendra d'écrire), écrire une fonction `MiseAJour`(T, P, k, V) qui prend en argument un tournoi T, un pronostic P, un match $k \geqslant n$, et un vecteur de probabilités V et qui met à jour le vecteur de probabilités V après le match k du tournoi T.

On supposera que T est un tournoi, P est un pronostic, k est un entier compris entre n et $2n-2$, et V est un vecteur de réels de longueur n.

Question 11

Écrire une fonction `ChancesDeVictoire`(T, P) qui prend en argument un tournoi T et un pronostic P et qui renvoie un vecteur de réels V tel que est la probabilité que i remporte le tournoi T selon le pronostic P.

On supposera, sans avoir à le vérifier, que T est un tournoi et que P est un pronostic.

* *
*