Devoir maison n°1

À rendre le lundi 18/09

⋆ ⋆ ⋆

Système de vote

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et en Computer Modern à chasse fixe du point de vue informatique (par exemple `n`).

Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n,m))$ où n et m sont les tailles des arguments de la fonction, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement.

Dans ce problème, les graphes ont un ensemble de sommets de la forme $\{0,1,\ldots,n-1\}$ et sont orientés, complets et pondérés par des entiers : pour tout couple (i,j) de sommets distincts, il existe un arc de i à j de poids $m_{i,j} \in \mathbb{Z}$. Le graphe est représenté par sa matrice d'adjacence $M = (m_{i,j})_{0 \leqslant i,j < n}$, avec la convention que $m_{i,i} = 0$ pour tout sommet i (il n'y a pas d'arc d'un sommet vers lui-même).

Vous indiquerez en début de copie la partie (1 ou 2) dont vous souhaitez la correction.

1 Vote par préférence

Cette partie est à traiter en langage OCaml.

Nous considérons une élection, à laquelle se présentent n candidats. Chaque électeur inscrit sur son bulletin l'ensemble des candidats (tous les candidats sont classés), par ordre de préférence. L'ensemble des bulletins est rassemblé dans une urne. Le nombre de votants est noté p, l'urne contient donc p bulletins.

Les trois types de données suivants sont utilisés pour représenter un vote :

- `type candidat = int` : chaque candidat est désigné par un numéro de 0 à $n-1$;
- `type bulletin = candidat list` : un bulletin de vote est une liste (ordonnée) de candidats ;
- `type urne = bulletin list` : une urne est un ensemble de bulletins de vote.

Par exemple, s'il y a trois candidats et qu'un électeur préfère le candidat 2, puis le candidat 0 et enfin considère que le candidat 1 est le moins souhaitable, son bulletin de vote sera `[2; 0; 1]`.

Une fois le vote effectué, on compare les résultats de deux candidats particuliers i et j en comptant le nombre de bulletins qui classent le candidat i avant le candidat j. On exprime le résultat de la comparaison entre les candidats i et j en calculant la différence entre le nombre de bulletins de vote qui placent i avant j et le nombre de ceux qui placent j avant i.

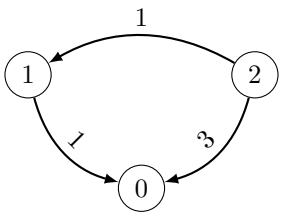
1.1 Premier exemple

On considère une élection avec trois candidats et trois votants : $n = 3$, $p = 3$. Les bulletins sont `[2; 0; 1]`, `[2; 1; 0]` et `[1; 2; 0]`. La comparaison entre le candidat numéro 0 et le candidat numéro 1 donne -1 car le candidat 0 est placé avant le candidat 1 et deux fois après.

Question 1 Écrire une fonction `duel (i: candidat) (j: candidat) (u: urne) : int` qui renvoie la comparaison entre le candidat i et le candidat j à partir des bulletins contenus dans l'urne u, u étant une urne quelconque.

On peut alors synthétiser le contenu d'une urne u en construisant le *graphe de préférence* des votants : ses sommets correspondent aux candidats et l'arc du sommet i vers le sommet j est pondéré par la comparaison entre le candidat i et le candidat j, c'est-à-dire la valeur de `duel i j u`.

La figure 1 donne le graphe de préférence obtenu à partir du vote de l'exemple 1, ainsi que la matrice d'adjacence M. Afin d'alléger le schéma, seuls les arcs avec un poids strictement positif sont représentés.



$$M = \begin{pmatrix} 0 & -1 & -3 \\ 1 & 0 & -1 \\ 3 & 1 & 0 \end{pmatrix}$$

FIGURE 1 – Un graphe de préférence et sa matrice d'adjacence

1.2 Deuxième exemple

On considère une élection avec trois candidats et 4 votants : $n = 3$, $p = 4$. À l'issue du vote, le contenu de l'urne est `[[0; 1; 2]; [1; 2; 0]; [0; 2; 1]; [0; 2; 1]]`.

Question 2 Tracer le graphe de préférence de l'urne (en ne dessinant que les arcs ayant un poids strictement positif) et donner sa matrice d'adjacence.

1.3 Construction du graphe de préférence

Question 3 Expliquer pourquoi la matrice d'adjacence d'un graphe de préférence est antisymétrique et pourquoi tous ses coefficients non-diagonaux ont la même parité.

Étant donné une urne U, on note $\mathrm{Mat}(U)$ la matrice d'adjacence du graphe de préférence associée à cette urne.

Question 4 Écrire une fonction `depouillement (n: int) (u: urne) : int array array` qui prend en paramètre le nombre de candidats et une urne u et renvoie la matrice $\mathrm{Mat}(u)$.

1.4 Théorème de McGarvey

Le but de cette sous-partie est de démontrer le théorème de McGarvey : « Pour toute matrice antisymétrique à coefficients pairs M, il existe une urne U telle que $M = \mathrm{Mat}(Uu)$. »

Soient i, j et n trois entiers naturels tels que $i < n$, $j < n$ et $i \neq j$. On note $E_{i,j,n}$ la matrice carrée de taille n dont tous les coefficients sont nuls sauf les coefficients d'indices (i, j) et (j, i) qui valent respectivement 2 et -2.

Question 5 Montrer qu'il existe une urne $U_{i,j,n}$ contenant deux votes telle que $\mathrm{Mat}(U_{i,j,n}) = E_{i,j,n}$.

Question 6 On considère deux urnes U_1 et U_2. Exprimer $M_3 = \mathrm{Mat}(U_1 \cup U_2)$ en fonction de $M_1 = \mathrm{Mat}(U_1)$ et $M_2 = \mathrm{Mat}(U_2)$.

Question 7 Démontrer le théorème de McGarvey.

Question 8 Écrire une fonction `mcgarvey (m: int array array) : urne` qui prend en argument une matrice antisymétrique à coefficients tous pairs et renvoie l'urne associée.

Question 9 Estimer la complexité de la fonction `mcgarvey` en fonction de n, la taille de la matrice, et de q, le maximum des coefficients de la matrice.

2 Recherche du vainqueur

Cette partie est à traiter en langage C.

L'objectif de cette partie est de déterminer le vainqueur, ou les vainqueurs *ex æquo*, d'un vote par préférence.

2.1 Manipulation de matrices

On représente une matrice d'adjacence M d'un graphe par un objet `M` de type `int**` tel que `M[i][j]` correspond au coefficient $M_{i,j}$.

Question 10 Écrire une fonction `int** copie_matrice(int** M, int n)` qui prend en argument une matrice et sa taille et renvoie une copie de cette matrice, sans liaison de données.

Question 11 Écrire une fonction `void liberer_matrice(int** M, int n)` qui libère toute la mémoire occupée par une matrice.

2.2 Vainqueur de Condorcet

On appelle *vainqueur de Condorcet* tout sommet tel que, dans le graphe de préférence, les arcs sortants de ce sommet ont tous un poids positif ou nul. Ainsi, dans le premier exemple de la partie 1, le candidat 2 est un vainqueur de Condorcet.

Question 12 Écrire une fonction `bool* condorcet(int** M, int n)` qui prend en argument une matrice d'adjacence M d'un graphe de préférence et sa taille n et renvoie un tableau de booléens `vainqueur` de taille n tel que pour tout candidat i, `vainqueur[i]` vaut `true` si et seulement si i est un vainqueur de Condorcet.

Question 13 En se plaçant dans le cas $n = 3$ et $p = 3$, construire une urne pour laquelle il n'existe pas de vainqueur de Condorcet. Tracer le graphe de préférence correspondant.

2.3 Graphe intermédiaire de Schultze

À la fin du XXe siècle, Markus Schulze imagina une méthode permettant de déterminer un vainqueur à l'issue de n'importe quel vote par préférence.

On appelle *poids d'un chemin* du graphe de préférence, le **minimum** des poids des arcs constituant ce chemin. Ainsi, dans le premier exemple de la partie 1, le poids du chemin $2 \to 0 \to 1$ est -1.

Le *graphe intermédiaire de Schulze* est un graphe orienté pondéré complet dont les sommets sont les candidats et dont l'arc du sommet i vers le sommet j (distinct de i) est pondéré par le **maximum** des poids de tous les chemins allant de i à j dans le graphe de préférence. Sa matrice d'adjacence est notée I.

Question 14 Pour i et j candidats distincts, montrer que $I[i, j]$ est aussi le maximum des poids des chemins sans boucle de i à j, c'est-à-dire des chemins $i = i_0 \to i_1 \to \ldots \to i_k = j$ avec $i_0, \ldots, i_k$ deux à deux distincts.

Question 15 Montrer que $\min(I[i, j], I[j, k]) \leqslant I[i, k]$ pour tout triplet (i, j, k) de sommets distincts.

Question 16 Écrire une fonction `int** intermediaire(int** M, int n)`, adaptée de l'algorithme de Floyd-Warshall, qui prend en argument une matrice d'adjacence M d'un graphe de préférence et sa taille n et renvoie la matrice d'adjacence du graphe intermédiaire de Schulze correspondant.

Question 17 Déterminer les complexités temporelle et spatiale de la fonction `intermediaire` en fonction de n.

Question 18 Serait-il pertinent d'adapter l'algorithme de Dijkstra au lieu de l'algorithme de Floyd-Warshall ? Justifier.

2.4 Graphe de préférence de Schulze

Le *graphe de préférence de Schulze* est défini à partir du graphe intermédiaire de Schulze. Si I est la matrice d'adjacence du graphe intermédiaire de Schulze, alors la matrice d'adjacence S du graphe de préférence de Schulze est définie par $S[i,j] = I[i,j] - I[j,i]$ pour tous entiers naturels i et j strictement inférieurs à n.

Question 19 Montrer que la matrice d'adjacence d'un graphe de préférence de Schulze est antisymétrique et que tous ses coefficients sont pairs.

Question 20 Écrire une fonction `int** graphe_schulze(int** I, int n)` qui prend en argument un graphe intermédiaire de Schulze et sa taille et renvoie le graphe de préférence de Schulze correspondant.

2.5 Vainqueur de Schulze

Un *vainqueur de Schulze* est un vainqueur de Condorcet dans le graphe de préférence de Schulze.

Question 21 Écrire une fonction `bool* schulze(int** M, int n)` qui prend en argument une matrice d'adjacence M d'un graphe de préférence et sa taille n et renvoie un tableau de booléens `vainqueur` de taille n tel que pour tout candidat i, `vainqueur[i]` vaut `true` si et seulement si i est un vainqueur de Schulze.

Question 22 Déterminer la complexité temporelle de la fonction `schulze` en fonction du nombre de candidats n.

À partir d'un graphe de préférence de Schulze représenté par la matrice S, on définit la relation R_S entre les candidats comme suit : iR_Sj si et seulement si $S[i,j] > 0$.

Question 23 Montrer que la relation R_S est transitive.

Indication : si I désigne la matrice d'adjacence du graphe intermédiaire, on pourra distinguer les cas $I[i,j] \leqslant I[j,k]$ et $I[i,j] > I[j,k]$.

Question 24 Montrer que quelle que soit l'urne non vide considérée, il existe toujours au moins un vainqueur de Schulze.

⋆ ⋆ ⋆