Centrale Informatique optionnelle MP 2020
Corrigé

Ce corrigé est proposé par William Aufort (professeur en CPGE) ; il a été relu par Vincent Puyhaubert (professeur en CPGE) et Benjamin Monmege (enseignant-chercheur à l'université).

Ce sujet porte sur les systèmes de vote dans lesquels l'électeur classe les candidats sur son bulletin par ordre de préférence. On peut alors modéliser l'ensemble des votes par une matrice traduisant, pour chaque couple de candidats, les préférences des électeurs. Les systèmes de vote par préférence diffèrent selon le mode de désignation du ou des vainqueurs de l'élection.

- La partie I permet de se familiariser avec les notions de vote et de graphe par préférence à travers deux exemples. On y démontre également le théorème de McGarvey, stipulant que toute matrice de préférence dont les coefficients sont pairs correspond aux résultats d'un dépouillement. La preuve proposée est constructive et donne également lieu à des questions de programmation.
- La partie suivante porte sur deux modes de désignation du ou des vainqueurs d'un vote par préférence. On y étudie rapidement la méthode de Condorcet, consistant à comparer les candidats deux par deux pour déclarer vainqueur celui qui est préféré à tous les autres, s'il existe. Cette méthode ne garantit hélas pas la présence d'un vainqueur. Le reste de la partie, qui constitue le cœur du problème, est consacré à la méthode de Schulze, qui est adaptée de l'algorithme de Floyd-Warshall au programme de l'option informatique. On prouve notamment que cette méthode assure la désignation d'un vainqueur.
- Enfin, la partie III étudie un problème de décision (relativement éloigné des deux premières parties) faisant intervenir la notion de vainqueur de Schulze. On y démontre notamment que le problème étudié est au moins aussi difficile que le problème de satisfiabilité d'une formule logique. Cette dernière partie de logique ne peut être convenablement abordée qu'après une bonne compréhension des notions introduites dans la partie précédente.

L'énoncé comporte 28 questions de difficulté globalement croissante. Seul un quart des questions nécessitent de programmer en OCaml, et la plupart d'entre elles sont abordables. Certains passages sont en revanche plus délicats et nécessitent une bonne compréhension, notamment la description de la méthode de Schulze ou encore de l'algorithme étudié dans la partie III. Ce sujet permet de bien s'entraîner sur les graphes, dans leurs aspects tant algorithmiques que théoriques.

Indications

Partie I

1 On peut commencer par écrire une fonction examinant un bulletin donné et renvoyant 1 ou -1 selon le candidat classé en premier parmi i et j.

3 Pour la parité des coefficients non diagonaux, utiliser le nombre total p de bulletins dans l'urne.

5 Considérer par exemple le bulletin

$$b_{i,j,n} = [\,i, j, 0, 1, \ldots, i-1, i+1, \ldots, j-1, j+1, \ldots, n-1\,]$$

et construire le second bulletin de sorte que l'urne obtenue soit associée à $\mathrm{E}_{i,j,n}$.

6 On doit obtenir $\mathrm{M}_3 = \mathrm{M}_1 + \mathrm{M}_2$.

7 Écrire M sous la forme d'une combinaison linéaire des matrices $\mathrm{E}_{i,j,n}$, puis utiliser les questions 5 et 6.

8 Ne pas hésiter à décomposer le code en plusieurs fonctions. Commencer par exemple par écrire une fonction implémentant la construction effectuée à la question 5. La fonction `List.rev` pourra être utile à cet effet.

9 Le code précédent étant composé de plusieurs fonctions, il est nécessaire de commencer par étudier la complexité des fonctions auxiliaires. Attention à la complexité de l'opérateur `@` en OCaml.

Partie II

11 Pour décider si un sommet donné est un vainqueur de Condorcet, il suffit d'examiner une seule ligne de la matrice.

13 Considérer un chemin ρ comportant une boucle et démontrer qu'en retirant cette boucle on obtient un chemin de poids supérieur ou égal au poids de ρ.

14 Introduire deux chemins, de i à k et de k à j, de poids respectifs $\mathrm{I}[i,k]$ et $\mathrm{I}[k,j]$, puis construire à l'aide de ces chemins un chemin de i à j.

15 Adapter la relation de récurrence de l'algorithme de Floyd-Warshall, ainsi que les conditions pour appliquer cette relation.

17 Comparer pour les deux algorithmes le contexte d'utilisation et la complexité.

18 Pour la parité des coefficients, justifier que ceux de la matrice I apparaissent tous nécessairement dans M.

22 Dans le cas où $\mathrm{I}[i,j] \leqslant \mathrm{I}[j,k]$, utiliser le résultat de la question 14 appliqué à $\mathrm{I}[j,i]$, puis à $\mathrm{I}[i,k]$. Le cas $\mathrm{I}[i,j] > \mathrm{I}[j,k]$ se traite de façon similaire.

23 En raisonnant par l'absurde, construire une suite d'entiers naturels $(i_k)_{k\in\mathbb{N}}$ strictement inférieurs à n telle que $\mathrm{S}[i_k, i_{k+1}] < 0$ pour tout $k \in \mathbb{N}$.

Partie III

25 Les rôles de x et $\neg x$ sont symétriques. Dans le cas où le candidat b_x a été choisi, calculer le coefficient $\mathrm{S}[c, a_2]$ dans la matrice de préférence de Schulze.

26 S'il existe une variable x pour laquelle les sommets b_x et $b_{\neg x}$ ont été choisis, justifier que pour une autre variable y, ni b_y ni $b_{\neg y}$ n'ont été choisis.

27 Procéder par double implication. Pour le sens direct, il faut choisir quels sommets sélectionner en fonction de l'interprétation de la formule. Pour le calcul de I, on se limitera à la ligne et la colonne correspondant au champion c.

28 Justifier que l'algorithme de transformation de formule en élection s'exécute en temps polynomial.

I. Vote par préférence

1 Commençons par écrire une fonction `traite_bulletin` qui compare les candidats i et j sur un bulletin donné, et renvoie 1 ou -1 en fonction du candidat classé en premier parmi i et j. Pour cela, on parcourt récursivement la liste associée jusqu'à trouver pour la première fois l'entier i ou l'entier j.

```
let rec traite_bulletin i j b = match b with
  | [] -> failwith "i et j absents du bulletin"
  | x::q -> if x = i then 1
            else if x = j then -1
            else traite_bulletin i j q;;
```

> Une erreur classique consiste à utiliser directement les noms de variable `i` et `j` dans les motifs du filtrage :
>
> ```
> | i::q -> 1
> | j::q -> -1
> | x::q -> traite_bulletin i j q
> ```
>
> En effet, on ne peut utiliser dans les filtrages en OCaml que des constructeurs (comme `::` sur les listes), des constantes, mais pas des valeurs calculées.
>
> Le résultat de cette fonction est incorrect si $i = j$. Ce cas, où il n'y a pas réellement de candidats à comparer, intervient pourtant dans la construction de la matrice d'adjacence à la question 4. Cette situation sera gérée dans la fonction `duel` directement.

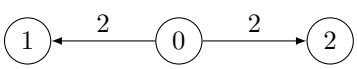
Le résultat de la comparaison des deux candidats s'obtient alors en additionnant les résultats des appels à la fonction `traite_bulletin` sur chaque bulletin de l'urne.

```
let rec duel i j u = match u with
  | _ when i = j -> 0 (* voir remarque precedente *)
  | [] -> 0
  | b::q -> (traite_bulletin i j b) + (duel i j q);;
```

2 Soit $\mathrm{M} = (m_{ij})_{0 \leqslant i,j \leqslant 2}$ la matrice d'adjacence du graphe de préférence de l'urne. Considérons par exemple les candidats 0 et 2 : un seul bulletin (le deuxième) classe 2 avant 0, tandis que les trois autres classent 0 avant 2. On obtient alors $m_{0,2} = 3-1 = 2$ et $m_{2,0} = 1 - 3 = -2$. En poursuivant de la même manière pour les autres couples de candidats, on en déduit que

$$\boxed{\mathrm{M} = \begin{pmatrix} 0 & 2 & 2 \\ -2 & 0 & 0 \\ -2 & 0 & 0 \end{pmatrix}}$$

En ne représentant que les arcs ayant un poids strictement positif, le graphe de préférence associé à cette urne est le suivant.



3 Soit U une urne, et soit $M = (m_{i,j})_{0\leqslant i,j\leqslant n-1}$ la matrice d'adjacence de son graphe de préférence, où n désigne le nombre de candidats. Pour tout couple (i,j) de candidats, notons $d_{i,j}$ le nombre de bulletins qui placent i avant j. Pour tout candidat i, on a par convention $m_{i,i} = 0$. De plus, pour tout couple (i,j) de candidats distincts, on obtient par définition de la matrice M

$$m_{i,j} = d_{i,j} - d_{j,i} = -(d_{j,i} - d_{i,j}) = -m_{j,i}$$

ce qui prouve que

> La matrice d'adjacence d'un graphe de préférence est antisymétrique.

Par ailleurs, si p désigne le nombre de bulletins dans l'urne, on a $d_{j,i} = p - d_{i,j}$ pour tout couple (i,j) de candidats distincts. Dès lors, on obtient avec la même hypothèse $m_{i,j} = d_{i,j} - (p - d_{i,j}) = 2d_{i,j} - p$. L'entier $2d_{i,j}$ étant pair, la parité du coefficient $m_{i,j}$ est donc indépendante de i et j. Autrement dit,

> Les coefficients non diagonaux d'une matrice d'adjacence d'un graphe de préférence ont tous la même parité.

> On a montré plus précisément que cette parité correspond à la parité du nombre p de bulletins dans l'urne.

4 Commençons par créer une matrice nulle, puis utilisons la fonction `duel` de la question 1 pour calculer ses coefficients. On obtient la fonction suivante.

```
let depouillement n u =
  let mat = Array.make_matrix n n 0 in
  for i = 0 to (n-1) do
    for j = 0 to (n-1) do
      mat.(i).(j) <- duel i j u
    done
  done;
  mat;;
```

5 Supposons que $i < j$ sans perte de généralité. Définissons les deux listes ordonnées

$$b_{i,j,n} = [\,i, j, 0, 1, \ldots, i-1, i+1, \ldots, j-1, j+1, \ldots, n-1\,]$$

et

$$b'_{i,j,n} = [n-1, \ldots, j+1, j-1, \ldots, i+1, i-1, \ldots, 1, 0, i, j]$$

Tous les entiers entre 0 et $n-1$ sont classés et ce sont les seuls. Dès lors, les deux listes précédentes définissent deux bulletins, et on définit $U_{i,j,n} = \left[b_{i,j,n}, b'_{i,j,n}\right]$ l'urne contenant ces deux bulletins. On pose enfin $M = \mathrm{Mat}\,(U_{i,j,n})$ la matrice d'adjacence du graphe de préférence associé à $U_{i,j,n}$.

Démontrons à présent que $M = E_{i,j,n}$. Ces deux matrices étant antisymétriques, il suffit de montrer que leurs coefficients d'indices (k,ℓ) vérifiant $k < \ell$ sont égaux deux à deux. Si $k = i$ et $\ell = j$, alors les deux bulletins placent i avant j, donc $m_{i,j} = 2$ et $m_{j,i} = -2$, correspondant aux seuls coefficients non nuls de la matrice $E_{i,j,n}$. Il reste à montrer que tous les autres coefficients sont nuls. Il y a trois cas :

- Si $k \in \{i,j\}$ et $\ell \notin \{i,j\}$, alors $b_{i,j,n}$ place k avant ℓ et $b'_{i,j,n}$ place ℓ avant k, donc $m_{k,\ell} = 0$.
- De même, si $\ell \in \{i,j\}$ et $k \notin \{i,j\}$, $b_{i,j,n}$ place ℓ avant k et $b'_{i,j,n}$ place k avant ℓ, d'où $m_{k,\ell} = 0$.
- Si $k \notin \{i,j\}$ et $\ell \notin \{i,j\}$, $b_{i,j,n}$ place k avant ℓ (les candidats différents de i et j sont placés dans l'ordre croissant) et $b'_{i,j,n}$ place ℓ avant k (les candidats différents de i et j sont placés dans l'ordre décroissant), soit $m_{k,\ell} = 0$.