TP10 : graphes et systèmes de votes

AP3 Algorithmique et programmation — L2 mathématiques

15 décembre 2016

1 Graphes et cycles

Exercice 1 (Détection d'un cycle dans un graphe orienté)**.**
On représente un graphe orienté $G = (S, A)$ par sa matrice d'adjacence. Autrement dit, si $S = \{0, 1, \dots, n-1\}$, le graphe G est donné par la liste de listes M telle que $M[i][j] = 1$ si $(i, j) \in A$ et $M[i][j] = 0$ sinon.

(a) Ecrire une fonction prenant en entrée un graphe orienté G et renvoyant `True` si G est acyclique et `False` sinon.

(b) Donner la complexité de cette procédure.

2 Application aux systèmes de votes

On s'intéresse dans cet exercice à des problématiques d'algorithmique élémentaires pour l'agrégation de préférences en particulier ici les méthodes de décision liées aux systèmes de vote.

Exercice 2 (Lecture et mise en forme des votes)**.**
On considère un vote où les candidats sont $C_0, C_1, \dots, C_{n-1}$ et les votants $V_1, \dots, V_\ell$. Chaque votant exprime ses préférences sous la forme d'un classement ordonné de la liste de tous les candidats. Les listes de préférences sont stockées dans un fichier où la ligne i contient le vote de V_i. La liste de préférences d'un votant V_i correspond à une permutation π de $\{0, 1, \dots, n-1\}$ dont la signification est la suivante : $C_{\pi(0)}$ est le candidat préféré de V_i, puis c'est $C_{\pi(1)}$, etc. Dans ce cas, la ligne i du fichier contiendra cette permutation π écrite de la façon suivante :

$$\pi(0)\ \pi(1)\ \dots\ \pi(n-1)$$

Noter que l'espace est le séparateur des éléments de chaque liste de préférences dans le fichier.

Écrire une fonction Python `VoteFtoL(monFichier)` prenant en entrée le nom d'un fichier `monFichier` dans lequel est stocké l'ensemble des votes et renvoyant une liste double L dont l'élément $L[i]$ est la liste $[\pi(0), \pi(1), \dots, \pi(n-1)]$ de préférences du votant V_i.

Par exemple, pour la liste de votes suivante :

```
0 1 2 4 3
1 0 3 4 2
```

```
2 0 4 1 3
4 0 1 2 3
2 0 3 4 1
```

la liste renvoyée sera :

```
[[0, 1, 2, 4, 3], [1, 0, 3, 4, 2], [2, 0, 4, 1, 3], [4, 0, 1, 2,
    3], [2, 0, 3, 4, 1]]
```

Exercice 3 (Résultats des duels entre candidats)**.**

(a) On définit la fonction $f : \{0, 1, \dots, n-1\} \times \{0, 1, \dots, n-1\} \to \mathbb{N}$ de la façon suivante : étant donnés $i, j \in \{0, 1, ..., n-1\}$ tels que $i \neq j$, la fonction $f(i, j)$ est égale au nombre d'électeurs qui préfèrent le candidat C_i au candidat C_j (on pose par exemple $f(i, i) = 0$). Écrire une fonction `duelCandidat(L)` prenant en entrée la liste L définie à la question précédente et renvoyant une liste de listes `comp` telle que `comp[i][j]`$= f(i, j)$ pour tout $i \neq j$.

(b) Écrire une fonction `MaxDuelScore(comp)` prenant en entrée une liste de listes de la forme précédente et renvoyant une liste M telle que pour tout $i \neq j$, M contient $[i, j, f(i, j)]$ si $f(i, j) > f(j, i)$ et $[j, i, f(j, i)]$ sinon. En d'autres termes, on garde pour tout duel entre deux candidats C_i et C_j le score du vainqueur.

(c) Modifier un des algorithmes de tris écrit dans une séance précédente pour obtenir une fonction prenant en entrée une liste de triplets d'entiers $[i, j, k]$ et renvoyant cette liste triée par ordre décroissant suivant la dernière coordonnée k.

Exercice 4 (La méthode de Tideman)**.**
Cette méthode, vue en cours, établit un processus de désignation de vainqueur dans le cas où un cycle de Condorcet existe. La méthode procède en trois étapes :

- A partir de l'expression des votes, on calcule la liste M des triplets $[i, j, f(i, j)]$ où C_i gagne son duel contre C_j avec le score $f(i, j)$;
- On trie la liste M par ordre décroissant selon la troisième coordonnée ;
- On construit le graphe orienté G dont l'ensemble des sommets est $\{0, 1, \dots, n-1\}$ et l'ensemble des arcs A est défini de la façon suivante. On considère successivement chaque élément $[i, j, f(i, j)]$ de la liste M (dans l'ordre) et à chaque étape, on insère l'arc (i, j) dans A si $A \cup \{(i, j)\}$ est sans cycle.

L'idée dans la dernière phase de l'algorithme est de construire un graphe de préférences sans cycle en donnant priorité aux duels remportés avec le plus d'écart. Le gagnant de l'élection est l'unique sommet du graphe sans arête entrante. Notez qu'une liste complète de préférences peut être obtenue en éliminant les uns après les autres les sommets sans arête entrante dans le graphe obtenu.

Implémenter la méthode des paires ordonnées. Vous utiliserez la procédure de détection de cycles dans un graphe orienté écrite dans l'exercice 1.