Dépouillement d'elections (d'après X PSI 2005)

Durée : 2 heures

La complexité temporelle T d'une fonction f est le nombre d'opérations élémentaires nécessaire au calcul de f. Lorsque cette complexité dépend d'un paramètre n, elle sera noté $T(n)$. On dit que la fonction f s'exécute :

- en temps linéaire s'il existe $K > 0$ tel que pour tout n, $T(n) \leqslant Kn$ (autrement dit si $T(n) = O(n)$) ;
- en temps quadratique s'il existe $K > 0$ tel que pour tout n, $T(n) \leqslant Kn^2$ (autrement dit si $T(n) = O(n^2)$).

Une élection est modélisée de la façon suivante : les votants sont numérotés de 0 à $n-1$, où n est un entier supérieur ou égal à 2, et chaque votant choisit la personne pour qui il veut voter (tous les votants sont automatiquement candidats). Ainsi, un tour de vote peut être représenté par un tableau `vote` de taille n où pour tout $k \in [\![0, n-1]\!]$, `vote[`k`]` $\in [\![0, n-1]\!]$ représente le candidat choisit par k.

Partie I. Dépouillement du premier tour

Pour être élu au premier tour, un candidat doit recueillir strictement plus que la moitié des votes.

Question 1. Rédiger une fonction `estGagnant(k, vote)` qui prend pour arguments un entier $k \in [\![0, n-1]\!]$ et le tableau `vote` et renvoie le booléen `True` si k est élu au premier tour, et `False` sinon.

Question 2.

a) En déduire une fonction `gagnantTour1(vote)` qui prend pour argument le tableau `vote` et renvoie le numéro du vainqueur au premier tour, s'il existe (et ne renvoie rien, ou plus exactement la valeur `None` dans le cas contraire).

b) Exprimer en la justifiant la complexité temporelle de `gagnantTour1` en fonction de l'entier n.

Question 3. Dans cette question uniquement, on suppose le tableau `vote` trié en ordre croissant, autrement dit vérifiant `vote[0]` $\leqslant$ `vote[1]` $\leqslant \cdots \leqslant$ `vote[`$n-1$`]`.
Rédiger une fonction `gagnantTour1Trie(vote)` qui prend pour argument le tableau `vote` et renvoie en temps linéaire le vainqueur au premier tour, s'il existe.

Question 4. En admettant que l'on sache trier un tableau de taille n avec une complexité en $O(n \log n)$, quel est l'ordre de grandeur du temps pris par le dépouillement du premier tour ?

Partie II. Dépouillement du deuxième tour

S'il n'y a pas de candidat élu au premier tour un second tour est organisé, mais cette fois c'est le candidat qui a obtenu le plus de voix qui est élu. En cas d'égalité, tous les candidats ayant obtenus ce nombre maximal de voix sont élus.

Question 5. Rédiger une fonction `nombreDeVoix(k, vote)` qui prend pour arguments un entier $k \in [\![0, n-1]\!]$ et le tableau `vote` et renvoie le nombre de votes que le candidat k a obtenu.

Question 6.

a) En déduire une fonction `gagnantsTour2(vote)` qui prend pour argument le tableau `vote` et renvoie la liste des gagnants du second tour.

b) Exprimer en la justifiant la complexité temporelle de `gagnantsTour2` en fonction de l'entier n.

Question 7. Dans cette question uniquement on suppose le tableau `vote` trié en ordre croissant.
Rédiger une fonction `gagnantsTour2Trie(vote)` qui prend pour argument le tableau `vote` et renvoie en temps linéaire la liste des gagnants du second tour.

Question 8. En admettant que l'on sache trier un tableau de taille n avec une complexité en $O(n \log n)$, quel est l'ordre de grandeur du temps pris par le dépouillement du second tour ?

Partie III. Dépouillement rapide du premier tour

On suppose de nouveau le tableau `vote` non trié.

Un multi-ensemble E est un ensemble où on autorise chaque élément à apparaître plusieurs fois.
Par exemple, $\langle 0,2,2,3,0,0,0,5,0\rangle$ est un multi-ensemble.
On dit que x est un élément majoritaire dans un multi-ensemble E de n éléments lorsque x apparaît strictement plus de $n/2$ fois dans E. Le gagnant du premier tour est donc l'élément majoritaire (s'il existe) dans le multi-ensemble $\langle$`vote[0]`,`vote[1]`,...,`vote[`$n-1$`]`$\rangle$. Dans ce dernier cas, on dira plus simplement que x est majoritaire dans le tableau `vote`.

Question 9. Démontrer que si x est majoritaire dans le tableau `vote` et qu'il n'existe pas d'élément majoritaire dans le multi-ensemble $\langle$`vote[0]`,`vote[1]`,...,`vote[`$k-1$`]`$\rangle$ alors x est majoritaire dans le multi-ensemble $\langle$`vote[`k`]`,`vote[`$k+1$`]`,...,`vote[`$n-1$`]`$\rangle$.

Question 10.

a) On considère la fonction suivante :

```
def gagnantPotentiel(vote):
    c, v = vote[0], 1
    for k in range(1, len(vote)):
        if v == 0:
            c, v = vote[k], 0
        if vote[k] == c:
            v += 1
        else:
            v -= 1
    return c
```

Démontrer que *s'il existe un élément majoritaire* dans le tableau `vote`, cet élément est renvoyé par cette fonction. (S'il n'existe pas d'élément majoritaire, l'élément renvoyé est quelconque.)

b) En déduire une fonction `gagnantTour1Rapide(vote)` qui prend pour argument le tableau `vote` et renvoie en temps linéaire le vainqueur du premier tour, s'il existe.