Épreuve d'informatique de l'X - 2005 - PSI

Dépouillement d'élections

Remarque : Compte tenu de l'énoncé, il est probable qu'il faille réaliser les fonctions sans utiliser de tableau auxiliaire qui aurait une taille équivalente à celle du tableau a. Ne disposant pas du rapport de concours, il est difficile de se faire une idée sur les contraintes que voulait imposer l'auteur du sujet.

Préliminaires : On crée une fonction `longueur` égale à la fonction python `len`.

```
1  longueur = len
```

Question 1

```
1  def gagnant_tour1(vote) :
2      n = longueur(vote)
3      majorite = n/2.
4      for i in range(n) :
5          candidat = vote[i]
6          compt = 0
7          for j in range(n):
8             # On comptabilise les votes (dépouillement)
9              if vote[j] == candidat :
10                 compt = compt+1
11         if compt > majorite :
12             return(candidat)
13     return(-1)
```

L'algorithme proposé est en temps quadratique. Pour chaque candidat $vote[i]$, on compte le nombre de voix obtenues par $vote[i]$ (boucle lignes 7-10). Si ce nombre dépasse la majorité, on retourne la valeur de $vote[i]$ (lignes 11-12). Si personne n'est éligible au premier tour, on retourne la valeur -1 (ligne 13).

(PSI*)

Question 2 Tant que le vote du $i^{ème}$ électeur est égal à celui de son prédécesseur, on incrémente `compt` le compteur de bulletins de vote sinon on réinitialise le compteur `compt` et le candidat (lignes 6 à 10). À chaque étape, on vérifie si le candidat étudié a atteint la majorité. Si c'est le cas, on retourne son nom, sinon on continue. Si on quitte la boucle, c'est qu'aucun candidat n'a la majorité, on retourne -1 (ligne 14).

```
1  def gagnant_tour1_bis( vote ):
2      n = longueur(vote)
3      majorite = n/2.
4      compt = 1
5          candidat = vote[0]
6      for i in range(1,n):
7          if vote[i] == candidat:
8              compt = compt+1
9          else :
10             compt = 1
11             candidat = vote[i]
12         if compt > majorite :
13             return(candidat)
14     return(-1)
```

On peut aussi remarquer que quand le tableau a est rangé en ordre croissant, on a : (x majoritaire ssi $\exists i$ tel que $x = a[i]$ et $a[i] = a[i] + n/2$) ce qui nous donne la solution simple suivante qui ne nécessite pas de compter le nombre de votes :

```
1  def gagnant_tour1_bis( vote) :
2      n = longueur(vote)
3      majorite = floor(n/2)+1
4      for i in range(floor(n/2)):
5          if vote[i] == vote[i+majorite] :
6              return(vote[i])
7      return(-1)
```

Question 3 : le tri se fait en $O(n\ln(n))$ suivi du dépouillement qui se fait en $O(n)$ qui est négligeable devant le $O(n\ln(n))$, l'ordre de grandeur est donc en $O(n\ln(n))$.

Question 4 :

```
1  def gagnant_tour2(vote):
2      n = longueur(vote)
3      max = 0
4      # on déterminer le meilleur score en voix
5      for i in range(n) :
6          compt = 0
7          candidat = vote[i]
8          for j in range(n) :
9              if vote[j] == candidat :
10                 compt = compt+1
11         if compt > max :
12             max = compt
13     # on imprime les candidats qui ont obtenu le meilleur score
14     for i in range(n) :
15         compt = 0
16         candidat = vote[i]
17         for j in range(i,n) :
18             if vote[j] == candidat :
19                 compt = compt+1
20         if compt == max :
21             print(vote[i])
```

Dans une première étape, on détermine le meilleur score de tous les candidats (lignes 5-12), puis on reparcourt le tableau pour imprimer les candidats qui ont obtenus ce meilleur score. Ces deux étapes ce font en temps quadratique.

Question 5 : On commence par déterminer le nombre maximum de voix qu'a reçu un candidat (boucle des lignes 3 à 9), puis on affiche les noms des candidats qui ont eu ce maximum de voix (boucle des lignes 10 à 12).

```
1  def gagnant_tour2_bis(vote) :
2      n = longueur(vote); compt = 1; maxi = 1;
3      for i in range(n) :
4          if vote[i] == vote[i-1] :
5              compt = compt+1
6              if compt > maxi :
7                  maxi = compt
8          else :
9              compt=1
10     for i in range(1, n-maxi):
11         if vote[i] == vote[i+maxi-1]:
12             print(vote[i])
```

Question 6 : le tri se fait en $O(n\ln(n))$ suivi du dépouillement qui se fait en $O(n)$ qui est négligeable devant le $O(n\ln(n))$, l'ordre de grandeur est donc en $O(n\ln(n))$.

Question 7 : Si x est majoritaire dans $\langle a[0], a[1], \ldots, a[n-1]\rangle$, il apparaît strictement plus de $n/2$ fois. Si on supprime deux éléments différents y et z, on supprimera au plus une fois x. Donc le nombre de fois qu'apparaît x dans la nouvelle liste est supérieur strict à $n/2 - 1 = \dfrac{n-2}{2}$ or $(n-2)$ est le cardinal de la nouvelle liste donc x reste majoritaire dans la liste obtenue en supprimant y et z.

Question 8 : si x est majoritaire dans $a = \langle a[0], a[1], \ldots, a[n-1]\rangle$, il y apparaît strictement plus de $n/2$ fois. Si dans $\langle a[0], a[1], \ldots, a[i-1]\rangle$ il n'y a pas d'élément majoritaire, x n'y ait donc pas majoritaire donc y apparaît au plus $i/2$ fois. Donc x apparaît strictement plus que $n/2 - i/2 = \dfrac{n-i}{2}$ fois parmi les $n-i$ derniers, c'est à dire x est majoritaire dans $\langle a[i], a[i+1], \ldots, a[n-1]\rangle$.

Question 9 :

```
1  gagnant_tour1 := proc(a :: array)
2   local n, b, i;
3   n := longueur(a);
4   b := array(0..n-1);
5   # initialisation du tableau "b"
6   for i from 0 to n-1 do b[i] := 0 od;
7   # On comptabilise les votes (dépouillement)
8   for i from 0 to n-1 do b[a[i]]:= b[a[i]]+1 od;
9   for i from 0 to n-1 do
10     if b[i] > (n/2) then return(i) fi;
11  od;
12  return(-1)
13  end;
```

L'algorithme proposé est en temps linéaire, mais n'utilise aucune des propriétés mises en évidence précédemment. Je suppose que ce n'est pas la solution attendue par l'auteur du sujet, le problème venant certainement du fait que cette solution utilise un deuxième tableau.

Voici une solution qui me semble dans l'esprit du problème et qui utilise bien les résultats des questions 7 et 8.

```
1  def gagnant_tour1_ter(a) :
2      n = longueur(a)
3      elu = -1
4      nb_voix = 0
5      nb_votants = 0
6      for i in range(0,n-1,2) :
7          if a[i] <> a[i+1] :
8              dummy = 0
9          elif a[i] == elu :
10             nb_voix = nb_voix + 2
11             nb_votants = nb_votants + 2
12         elif elu == -1 :
13             elu = a[i]; nb_voix = 2; nb_votants = 2
14         elif nb_votants + 2 >= 2*nb_voix :
15             elu = -1; nb_votants = 0; nb_voix = 0
16         else :
17             nb_voix = nb_voix; nb_votants = nb_votants + 2
18     # Cas où n impair et elu=-1 en fin de boucle
19     if (n % 2 == 1) and (elu == -1) :
20         elu = a[n-1]
21     # On compte le nombre de voix du seul candidat qui a peut-être la majo
22     nb_voix = 0;
23     for i in range(n) :
24         if a[i] == elu :
25             nb_voix = nb_voix+1
26     if nb_voix > n/2:
27         return(elu)
28     else :
29         return(-1)
```

Les variables `nb_voix` et `nb_votants` comptabilisent respectivement le nombre de voix de l'élément majoritaire de notre sous-ensemble ainsi que le cardinal du sous-ensemble d'électeurs considéré.
lignes 7-8 : on est dans le cas de la question 7, on ne modifie pas l'élément majoritaire en ne prenant pas en compte deux votes distincts.
lignes 9-11 : l'élément `elu` reste majoritaire et on met à jour son nombre de voix et le nombre de votants considérés.
lignes 12-13 : on n'avait pas d'élément majoritaire, on vient d'en trouver un qui pourra l'être et qui en tout cas l'est dans le multi-ensemble [a[i],a[i+1]].
lignes 16-17 : l'élément qui était majoritaire ne l'est plus en rajoutant à notre multi-ensemble les deux votes [a[i],a[i+1]], du coup, on se retrouve dans le cas de la question 8. Si un élément est majoritaire, il l'est dans le multi-ensemble

des votes non encore traités.
lignes 19-20 : On traite le cas où le nombre de votants est impair et qu'il ne s'est pas dégagé d'élément majoritaire dans les $(n-1)$ premiers votes.
lignes 23-25 : on comptabilise le nombre de voix de l'élu potentiel.
lignes 26-29 : on publie les résultats.