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 suppose que l'on dispose d'une fonction qui donne correctement la longueur d'un tableau comme la fonction suivante :

```
1  longueur := proc(a :: array)
2    return(nops(convert(a,list)));
3   end;
```

Question 1

```
1  gagnant_tour1 := proc(vote :: array)
2  local n, compt, i, j, candidat, majorite;
3  n := longueur(vote);
4  majorite := n/2;
5  for i from 0 to n-1 do
6    candidat := vote[i];
7    compt := 0;
8    for j from 0 to n-1 do
9    # On comptabilise les votes (dépouillement)
10     if vote[j] = candidat then compt := compt+1 fi;
11   od;
12   if compt > majorite then return(candidat) fi;
13 od;
14 return(-1)
15 end;
```

L'algorithme proposé est en temps quadratique. Pour chaque candidat $vote[i]$, on compte le nombre de voix obtenues par $vote[i]$ (boucle lignes 8-11). Si ce nombre dépasse la majorité, on retourne la valeur de $vote[i]$ (ligne 12). Si personne n'est éligible au premier tour, on retourne la valeur -1 (ligne 14).

(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 8 à 11). 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  gagnant_tour1_bis := proc( vote :: array)
2  local n, i, majorite, compt, candidat;
3  n := longueur(vote); majorite := n/2;
4  compt := 1; candidat := vote[0];
5  for i from 1 to n-1 do
6     if vote[i] = candidat
7           then compt := compt+1;
8           else compt := 1; candidat := vote[i]
9     fi;
10    if compt > majorite then return(candidat) fi;
11 od;
12 return(-1);
13 end;
```

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  gagnant_tour1_bis := proc( vote:: array)
2  local n, i, majorite, compt, c;
3  n := longueur(vote);
4  majorite := floor(n/2)+1;
5  for i from 0 to floor(n/2)-1 do
6     if vote[i] = vote[i+majorite] then return(vote[i]) fi;
7  od;
8  return(-1);
9  end;
```

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  gagnant_tour2 := proc(vote :: array)
2  local n, i, j, candidat, max, compt;
3  n := longueur(vote);
4  max := 0;
5  # on déterminer le meilleur score en voix
6  for i from 0 to n-1 do
7    compt := 0;
8    candidat := vote[i];
9    for j from 0 to n-1 do
10       if vote[j]=candidat then compt := compt+1 fi;
11   od;
12   if compt > max then max := compt fi;
13 od;
14 # on imprime les candidats qui ont obtenu le meilleur score
15 for i from 0 to n-1 do
16   compt := 0;
17   candidat := vote[i];
18   for j from i to n-1 do
19       if vote[j]=candidat then compt := compt+1 fi;
20   od;
21   if compt = max then print(vote[i]) fi;
22 od;
23 end;
```

Dans une première étape, on détermine le meilleur score de tous les candidats (lignes 5-13), 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 5 à 10), puis on affiche les noms des candidats qui ont eu ce maximum de voix (boucle des lignes 11 à 13).

```
1  gagnant_tour2_bis := proc(a :: array)
2  local n, i, compt, maxi;
3  n := longueur(a); compt := 1; maxi := 1;
4  for i from 1 to n-1 do
5    if a[i]=a[i-1]
6       then compt:=compt+1; if compt>maxi then maxi := compt fi
7       else compt:=1
8    fi
9  od;
10 for i from 1 to n-maxi-1 do
11   if a[i]=a[i+maxi-1]then print(a[i]) fi
12 od;
13 end;
```

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  gagnant_tour1_ter := proc(a :: array)
2  local n, i, elu, nb_voix, nb_votants, dummy;
3  n := longueur(a); elu := -1; nb_voix := 0; nb_votants := 0;
4  for i from 0 to n-2 by 2 do
5     if a[i] <> a[i+1]  then dummy:=0
6   elif a[i] = elu then nb_voix := nb_voix + 2;
7                        nb_votants := nb_votants + 2;
8   elif elu = -1 then elu := a[i]; nb_voix := 2; nb_votants := 2;
9   elif nb_votants + 2 >= 2*nb_voix
10                then elu := -1; nb_votants := 0; nb_voix := 0;
11   else nb_voix := nb_voix; nb_votants := nb_votants + 2 fi;
12 od;
13 # Cas où n impair et elu=-1 en fin de boucle
14 if (n mod 2 = 1) and (elu = -1) then elu := a[n-1] fi;
15 # On compte le nombre de voix du seul candidat qui a peut-être la majorité
16 nb_voix := 0;
17 for i from 0 to n-1 do
18    if a[i] = elu then nb_voix := nb_voix+1; fi;
19 od;
20 if nb_voix > n/2 then return(elu)
21                  else return(-1)
22 fi;
23 end;
```

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.
ligne 5 : 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.
ligne 6-7 : l'élu augmente le nombre de ses voix,
ligne 8 : 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]].
ligne 9-10 : 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.
ligne 11 : l'élément `elu` reste majoritaire et on met à jour son nombre de voix et le nombre de votants considérés.
ligne 14 : 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.
ligne 17-19 : on comptabilise le nombre de voix de l'élu potentiel.
ligne 20-21 : on publie les résultats.