ÉCOLE POLYTECHNIQUE - PSI/PT - ÉPREUVE FACULTATIVE D'INFORMATIQUE 2002

Corrigé rédigé par Alain Schauber - alain.schauber@prepas.org

NB : dans ce problème, on manipule des tableaux unidimensionnels indexés de 0 à n - 1. Pour définir un tel tableau, on ne peut pas utiliser ici la fonction vector car pour cette fonction l'indexation doit commencer à 1. On va donc se tourner vers la fonction array . Mais il faut alors préciser l'indexation à la définition.

Question 1

On initialise un compteur à 0 et on écrit une boucle impérative qui parcourt tout le tableau a en incrémentant le compteur chaque fois qu'on rencontre x.
A la fin de la boucle, la fonction retourne la valeur du compteur. On visite chaque cellule d'un tableau de longueur n une et seule fois, on a donc une fonction en temps linéaire en n.

```
> compte:=proc(x,a) local compteur,i;
  compteur:=0;
    for i from 0 to n-1
    do
      if a[i] = x then compteur := compteur+1 fi
    od;
    compteur
  end;
```

Question 2

On crée un nouveau tableau vide de longueur **n** que l'on remplit à l'aide d'une boucle : à la cellule d'indice i on affecte le nombre de fois que la cellule n° i du tableau **a** est présente dans le tableau **a**, ce qui est fourni par le résultat de **compte(a[i],a)**. Puis on retourne le tableau ainsi créé. La fonction est bien en temps quadratique en **n**, puisqu'elle met en jeu une boucle de **n** tours, chacun d'entre eux faisant appel à la fonction linéaire compte, qui est linéaire en **n**.

```
> occurences:=proc(a) local r,i;
    r:=array(0..n,[]);
    for i from 0 to n-1 do r[i]:=compte(a[i],a) od;
    eval(r) # force l'évaluation de oc au contenu de la variable
  end;
```

Question 3

On parcourt simultanément les tableaux **a** et **t** en tenant à jour une variable Tp qui contient la période à laquelle appartient la cellule n° i de **a** de la façon suivante : si la valeur de l'énergie (dont l'indice de la cellule de dernier changement est stockée dans une variable ind) reste constante alors on assigne à Tp la différence du temps lu dans la cellule n° i de **t** et du temps lu dans la cellule n° ind de **t**, sinon on réinitialise Tp à 0 et la variable ind à i. Chaque fois que la période courante Tp est augmentée, on vérifie si elle ne devient pas supérieure au max des périodes trouvées jusqu'ici, stocké dans une variable T . Après la boucle, cette fonction retourne T.
Elle est bien linéaire en **n** puisque chaque cellule de **a** est visitée une fois et une seule en exécutant à

chaque tour de boucle correspondant des instructions qui sont toutes élémentaires.

```
> maxconstant:=proc(a) local T,Tp,ind,i;
    T:=0;
    Tp:=0;
    ind:=0;
    for i from 1 to n-1
    do
      if a[i] = a[ind] then
        Tp:= t[i] - t[ind];
        if Tp > T then T := Tp fi
      else
        ind := i;
        Tp := 0
      fi
    od;
    T
  end;
```

Question 4

Le tableau occ contient dans la cellule n° i le nombre d'occurences du rayonnement de même indice dans le tableau a. Il suffit donc de chercher dans le tableau occ les indices des deux plus grandes valeurs (éventuellement égales) correspondant à des énergies différentes dans a. Cela peut se faire en parcourant le tableau occ et en tenant à jour deux variables i1 et i2 contenant respectivement ces deux indices. Pour respecter la convention de l'énoncé, i1 est initialisée à 0 (indice de la première cellule) et i2 à -1. Lors de la visite de la cellule n° i de occ, on compare d'abord la valeur de la cellule n° i de occ aux valeurs des cellules n° i1 et i2 de occ. Si occ[i] <= occ[i2], on passe à la cellule suivante. Sinon on met à jour i2 et éventuellement aussi i1. Dans les cas d'égalité d'occurences pour deux indices correspondant au max de ces occurences, on vérifie dans le tableau a si les valeurs correspondantes des énergies sont bien différentes avant de mettre à jour.
La fonction est bien linéaire en n puisqu'il s'agit là encore du parcours complet d'un tableau de longueur n.

```
> maxoccurences:=proc(a,occ) local i1,i2,i;
    i1:=0;
    i2:=-1;
    for i from 1 to n-1
    do
      if occ[i] > occ[i1] or occ[i] = occ[i1] and a[i] <> a[i1]
  then
        i2 := i1; i1 := i
      else
        if occ[i] < occ[i1] and (i2 = -1 or occ[i] > occ[i2]) then
  i2 := i fi
      fi
    od;
    print(i1), print(i2);
  end;
```

Question 5

En examinant la figure proposée en indication, on constate qu'il suffit de parcourir le tableau a tout en tenant à jour les variables b (correspondant à la fin de la zone 1), r (correspondant à la fin de la zone 2) ainsi qu'une variable s correspondant à la fin de la zone 3. b et r sont initialisées à 0 et s à n. On définit aussi une variable i contenant l'indice de la cellule courante de a.

Lors de la visite de la cellule i dans la zone 3 (avec r <= i < s), on examine à quelle zone doit appartenir la valeur correspondante : si elle doit appartenir à la zone 1 on l'échange avec le premier élément de la zone 2 (à condition que celle-ci soit non vide), et on incrémente b, r et i ; si elle doit appartenir à la zone 2 on incrémente r et i; si elle doit appartenir à la zone 4 on l'échange avec le dernier élément de la zone 3 et on décrémente s. Chaque échange dans a est accompagné en parallèle de l'échange correspondant dans t. La fonction s'arrête lorsque r = s et ne retourne rien (effet de bord).

A noter que pour simplifier l'écriture, on peut définir une sous-fonction locale échange réalisant l'échange de deux éléments d'indice donné d'un tableau.

La fonction est linéaire en n : en effet, au départ s - r vaut n, à chaque tour de boucle cette différence est décrémentée et la fonction s'arrête lorsque s - r vaut 0. La fonction comporte donc n tours de boucle, faisant tous appel uniquement à des instructions élémentaires.

```
> trier:=proc(a,t,m1,m2) local échange,b,r,s,i;
    échange:=proc(i,j,tableau) local c;
      c:=tableau[i];
      tableau[i]:=tableau[j];
      tableau[j]:=c
    end;
    b:=0;  r:=0; s:=n;  i:=0;
    while r < s
    do
      if a[i] = m1 then
        if b < r then #cas de la zone 2 non vide
          échange(b,i,a);
          échange(b,i,t)
        fi;
        b := b+1;
        r := r+1;
        i := i+1
      elif a[i] = m2 then
        r := r+1;
        i := i+1
      else
        échange(i,s-1,a);
        échange(i,s-1,t);
        s := s-1
      fi
    od;
    RETURN ()
  end;
```

Question 6

Les échanges dans t se faisant sur des critères totalement étrangers à l'ordre des dates, y compris au sein d'une même zone, il n'y a aucune raison pour que soit conservé l'ordre des dates. Voilà un contrexemple simple : si a1 = (1,2,1) et t1 = (1,2,3), alors m1 = 1 et m2 = 2.
Dans ce cas, l'appel trier(a1,t1,m1,m2) transforme a1 en (1,1,2) et t1 en (1,3,2).