ÉCOLE POLYTECHNIQUE - MP/PC - ÉPREUVE FACULTATIVE D'INFORMATIQUE 2002

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

Version de MAPLE : Classic Worksheet Maple 10.
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
La fonction ci-dessous s'exécute en temps linéaire par rapport à **n**. En effet, elle consiste à effectuer un parcours du tableau en tenant à jour deux variables contenant respectivement la plus grande et la plus petite valeur rencontrées jusqu'à présent dans le tableau, puis à retourner leur différence.

```
> amplitude:=proc(a) local maxi,mini,i;
    maxi:=a[0];
    mini:=a[0];
    for i from 1 to n-1 do
      if a[i] > maxi then maxi:=a[i] fi;
      if a[i] < mini then mini:=a[i] fi
    od;
    maxi-mini
  end;
```

Question 2
Si a contient dans cet ordre les n = 2 valeurs 2002 et 2001, alors le gain est nul tandis que l'amplitude vaut 1.
L'amplitude représente la plus grande des valeurs absolues du gain maximal et de la perte maximale.

Question 3
Après initialisation du gain à 0 (valeur minimale, atteinte dans la définition pour **i** = **j**), on utilise deux boucles impératives imbriquées.
La première (indexée par **i**) contient au tour de boucle n° **i** une boucle (indexée par **j**) de longueur **n-i** , chacun des tours de cette dernière étant constitué d'un nombre borné d'instructions élémentaires (encadrées par une instruction conditionnelle). Il existe donc un entier A tel que le nombre total d'instructions élémentaires de la fonction **gain** est majoré par Somme(0<=i<n,(n-i)*A)) = n*(n+1)*A/2=O(n^2). La fonction **gain** s'exécute donc bien en temps quadratique par rapport à **n**.

```
> gain:=proc(a) local g,i,j;
    g:=0; #valeur minimale possible du gain
    for i from 0 to n-1 do
      for j from i to n-1 do if a[j]-a[i] > g then g:=a[j]-a[i] fi
  od
    od;
    g
  end;
```

Question 4

On reprend le programme de la question 3, en ajoutant deux variables locales **achat** et **vente** contenant respectivement les dates d'achat **i** et de vente **j** correspondant au gain courant **g**. Pour minimiser **j-i** comme demandé, on autorise la mise à jour des variables **g**, **achat** et **vente** à gain **g** constant dès lors que la différence **vente-achat** diminue.

```
> gain_print:=proc(a) local g,achat,vente,i,j;
    g:=0; achat:=0; vente:=0; #initialisations
    for i from 0 to n-1 do
      for j from i to n-1 do
        if a[j]-a[i] > g or a[j]-a[i] = g and j-i < vente-achat
  then
          g:=a[j]-a[i]; achat:=i; vente:=j
        fi
      od
    od;
    print(achat),print(vente);
    g
  end;
```

Question 5

Comme dans l'énoncé, appelons gainCourant(i) le nombre Max(0<=k<=i,a(i)-a(k)). On peut remarquer que le gain maximum possible recherché est égal par définition à Max(0<=k<=i<=n-1,a(i)-a(k)) = Max(0<=i<=n-1,Max(0<=k<=i,a(i)-a(k))) = Max(0<=i<=n-1,gainCourant(i)).

De plus, il est clair par récurrence que gainCourant(i**)** = Max(0<=k<=i,a(i)-a(k)) = max(a(i)-a(i)), (a(i)-a(i-1))+gainCourant(i-1)) = max(0,(a(i)-a(i-1))+gainCourant(i-1)). Il en résulte qu'au cours d'un simple parcours du tableau **a**, on peut tenir à jour à chaque tour de boucle et en temps constant la valeur de gainCourant(i) dans une variable **gainCourant**. Il suffit donc lors du même parcours de **a** de tenir à jour une variable **g** contenant la plus grande valeur de **gainCourant** rencontrée jusqu'à présent.

La fonction **gain1** consiste ainsi en une boucle comportant **n** tours à temps constant, donc elle est bien linéaire par rapport à **n**.

```
> gain1:=proc(a) local gainCourant,g,i;
    gainCourant:=0; g:=0;
    for i from 1 to n-1 do
      gainCourant:=max(0,a[i]-a[i-1]+gainCourant);
      if gainCourant > g then g:=gainCourant fi
    od;
    g
  end;
```

Question 6

La modification est analogue à celle de la question 4 concernant la condition de mise à jour des variables **ac** et **ve** contenant respectivement la date d'achat et de vente qui minimisent leur différence. En revanche, il convient de «doubler» les dates optimales **ac** et **ve** rencontrées jusqu'alors dans la boucle par des valeurs courantes **achatCourant** et **venteCourante** qui contiennent les dates optimales correspondant au gain courant **gainCourant**. La date **venteCourante** est mise à l'indice courant si l'action a monté (condition **a[i] > a[i-1]**), la date **achatCourant** est réinitialisée à l'indice courant si l'attente jusqu'à l'indice courant se traduit par une opération blanche au niveau gain (condition **gainCourant = 0**).

```
> gain1_print:=proc(a) local
  gainCourant,g,achatCourant,ac,venteCourante,ve,i;
    gainCourant:=0; g:=0;
    achatCourant:=0; venteCourante:=0; ac:=0; ve:=0;
    for i from 1 to n-1 do
      gainCourant:=max(0,a[i]-a[i-1]+gainCourant);
      if gainCourant = 0 then achatCourant:=i fi;
      if a[i] > a[i-1] then venteCourante:=i fi;
      if gainCourant > g
      or gainCourant = g and venteCourante-achatCourant < ve-ac
      then
        g:=gainCourant;
        ve:=venteCourante; ac:=achatCourant
      fi
    od;
    print(ac),print(ve);
    g
  end;
```

Question 7

La longueur **n** du tableau **a** étant une variable globale, on va s'inspirer de la fonction **gain1** de la question 5 pour écrire une fonction locale **gainpartiel** qui prend en argument le tableau **a** et un indice **k** et ajoute les gains maximaux obtenus entre les dates 0 et **k** d'une part, entre les dates **k** et **n** -1 d'autre part. Chacun de ces deux gains partiels est calculé en appliquant aux deux parties de **a** à gauche et à droite de **k** le principe de l'algorithme utilisé dans la fonction **gain1**.
Il en résulte que le temps d'exécution de **gainpartiel** est de l'ordre de T_(k+1)(**gain1**)+T_(n-k)(**gain1**)+O(1)=O(k+1)+O(n-k)+O(1)=O(n).
Il ne reste plus qu'à faire varier **k** de 0 à **n**-1 et de mettre à jour au fur et à mesure dans une variable **g** le gain cumulé maximal. La boucle utilisée pour cela comporte **n** tours, la fonction **gain2** est donc en **n***O(**n**)=O(**n**^{2}).

```
> gain2:=proc(a) local gainpartiel,g,k;
    gainpartiel:=proc(a,k) local gainCourant,g1,g2,i;
      gainCourant:=0; g1:=0;
      for i from 1 to k do #algorithme de gain1 entre les indices
  0 et k
        gainCourant:=max(0,a[i]-a[i-1]+gainCourant);
        if gainCourant > g1 then g1:=gainCourant fi
      od;
      gainCourant:=0; g2:=0;
      for i from k+1 to n-1 do #algorithme de gain1 entre les
  indices k et n-1
        gainCourant:=max(0,a[i]-a[i-1]+gainCourant);
        if gainCourant > g2 then g2:=gainCourant fi
      od;
     g1+g2 #calcul du cumul des gains partiels
    end;
    g:=0;
    for k from 0 to n-1 do g:=max(g,gainpartiel(a,k)) od;
    g
  end;
```

Question 8

Même idée que la modification de **gain1** en **gain2**: on utilise l'algorithme employé dans **gain1_print** sur les deux parties d'une partition du tableau **a** à l'aide d'un indice **k**. Ceci est mis en oeuvre dans la fonction locale **gainpartiel**, qui retourne sous forme de liste le gain obtenu et les quatre dates correspondantes.

Il ne reste donc plus qu'à faire varier **k** afin de calculer le meilleur choix. On notera que l'optimisation se fait au niveau du gain obtenu, l'énoncé ne disant pas clairement comment privilégier à gain égal les solutions correspondantes par rapport aux durées de rétention des actions. A priori, une bonne idée aurait été d'imposer une somme (j-i)+(j'-i') minimale, ce qui pourrait se faire facilement au niveau de l'instruction conditionnelle de la boucle finale (indexée par **k**).

```
> gain2_print:=proc(a) local gainpartiel,g,optimal,k,l;
    gainpartiel:=proc(a,k)
      local
  gainCourant,achatCourant,venteCourante,ac1,ve1,ac2,ve2,g1,g2,i;
      gainCourant:=0; g1:=0;
      achatCourant:=0; venteCourante:=0; ac1:=0; ve1:=0;
      for i from 1 to k do
        gainCourant:=max(0,a[i]-a[i-1]+gainCourant);
        if gainCourant = 0 then achatCourant:=i fi;
        if a[i] > a[i-1] then venteCourante:=i fi;
        if gainCourant > g1
        or gainCourant = g1 and venteCourante-achatCourant <
  ve1-ac1
        then
          g1:=gainCourant;
          ve1:=venteCourante; ac1:=achatCourant
        fi
      od;
      gainCourant:=0; g2:=0;
      achatCourant:=k; venteCourante:=k; ac2:=k; ve2:=k;
      for i from k+1 to n-1 do
        gainCourant:=max(0,a[i]-a[i-1]+gainCourant);
        if gainCourant = 0 then achatCourant:=i fi;
        if a[i] > a[i-1] then venteCourante:=i fi;
        if gainCourant > g2
        or gainCourant = g2 and venteCourante-achatCourant <
  ve2-ac2
        then
          g2:=gainCourant;
          ve2:=venteCourante; ac2:=achatCourant
        fi
      od;
      [g1+g2,ac1,ve1,ac2,ve2]
      end;
    g:=0; optimal:=0;
    for k from 0 to n-1 do
      l:=gainpartiel(a,k);
      if l[1] > g then g:=l[1]; optimal:= k fi;
    od;
    l:=gainpartiel(a,optimal);
    print(l[2]),print(l[3]),print(l[4]),print(l[5]);
    l[1]
  end;
```