Université Bordeaux 1 LST 2003/2004

Initiation à l'algorithmique

Corrigé du Contrôle Continu du 31 mars 2004

Exercice .1

```
fonction  Mystère(ref T: tableau[0..NMAX-1] d'entiers,
                     val N: entier,
                     ref X,Y: entiers): vide;
  var i: entier;
  début
       X = T[0];
       Y = T[0];
       pour i allant de 1 jusqu'à N-1 faire
           si T[i] < X alors X = T[i];
           si T[i] > Y alors Y = T[i];
       finpour;
  fin
finFonction
```

1. On considère le tableau `T` contenant la suite $5, 9, 7, 3, -12, 15, 0$ et les variables `n=7`, `a=4`, `b=3`.
   Quel est l'effet de l'appel `Mystère(T,n,a,b)` ? Justifiez votre réponse.

   Attention : les paramètres `a` et `b` sont passés par référence. Toute modification de `X` est donc une modification de `a` et toute modification de `Y` est une modification de `b`.

   Trace de l'exécution :

```
N=7  X=4  Y=3
X=5  Y=5
i=1
          i <= N-1 ?? oui
             T[i] < X ?? non
             T[i] > Y ?? oui
                Y=9
i=2
          i <= N-1 ?? oui
             T[i] < X ?? non
             T[i] > Y ?? non
i=3
          i <= N-1 ?? oui
             T[i] < X ?? oui
                X=3
             T[i] > Y ?? non
i=4
          i <= N-1 ?? oui
```

```
                T[i] < X ?? oui
                   X=-12
                T[i] > Y ?? non
i=5
             i <= N-1 ?? oui
                T[i] < X ?? non
                T[i] > Y ?? oui
                   Y=15
i=6
             i <= N-1 ?? oui
                T[i] < X ?? non
                T[i] > Y ?? non
i=7
             i <= N-1 ?? non
```

L'appel de `Mystère(T,n,a,b)` a comme effet de modifier le contenu des deux variables `a` et `b` passées comme paramètres par réference. À la fin de l'exécution `a` contiendra -12 (le minimum des valeurs de la suite) et `b` contiendra 15 (le maximum des valeurs de la suite).

2. En général, que fait la fonction `Mystère` ?
   La fonction `Mystère` parcourt le tableau `T` en cherchant, pour chaque élément, s'il est le minimum et s'il est le maximum des éléments rencontrés jusque-là. Les deux valeurs, minimum et maximum se trouveront à la fin de l'exécution dans les deux derniers paramètres de la fonction.
3. Indiquez le nombre de comparaisons requises par cet algorithme.
   $2(N-1)$ comparaisons, puisque la boucle `pour` est effectuée $N-1$ fois et il y a deux comparaisons à l'interieur de celle-ci.
4. Quel est le nombre maximal d'affectations effectuées ? (Justifiez brièvement la réponse).
   Au plus il y a $N+1$ affectations : en effet on a toujours `X`$\leq$`Y`, donc on ne peut pas avoir à la fois `T[i]`$<$`X` et `T[i]`$>$`Y`, et il ne peut y avoir qu'une affectation pour chacun des $N-1$ passages dans la boucle `pour`. A noter que cette borne $N+1$ peut être atteinte (par exemple, si le tableau est en ordre croissant, `Y` sera affecté N fois, `X` une fois).
5. Indiquer l'ordre de complexité de l'algorithme.
   Il s'agit d'un parcours de tableau : chaque élément du tableau est testé deux fois. La complexité est donc $O(N)$.

Exercice .2 On considère deux tableaux A et B contenant respectivement `n` et `m` éléments. On dit que A est inclus dans B si tous les éléments de A figurent au moins une fois dans B.

1) Soit `T` un tableau dont le contenu n'est pas trié et soit la fonction :

`cherche (ref T:tableau[0..NMAX-1] d'élément; val n:entier; val e:élément):booléen;` qui retourne `vrai` si l'élément `e` est dans le tableau `T` et retourne `faux` sinon.

Écrire cette fonction.

```
fonction cherche (ref T:tableau[0..NMAX-1] d'élément,
                  val n:entier,
                  val e:élément):booléen;
   var i: entier;
   début
     i = 0;
     tantque i < n faire
       si T[i] == e alors
          retourner (vrai);
       i = i+1;
     fintantque;
     retourner (faux);
   fin
finFonction
```

Écrire une fonction `est_inclus` qui, sans trier les tableaux, dit si A est inclus dans B. On pourra utiliser la fonction `cherche` définie à la question précédente (même si on n'a pas su répondre à la question).
Quelle est la complexité de votre algorithme en nombre de comparaisons entre éléments des deux tableaux ?

```
fonction est_inclus (ref A:tableau[0..NMAX-1] d'élément,
                    ref B:tableau[0..NMAX-1] d'élément,
                    val n,m:entier):booléen;
   var i: entier;
   début
     i = 0;
     tantque i < n faire
       si non cherche(B,m,A[i]) alors
         retourner(faux);
       i = i+1;
     fintantque;
     retourner (vrai);
   fin
finFonction
```

Pour chaque élément du tableau `A` on parcourt le tableau `B` : il y a n éléments dans `A` et m dans `B` et la complexité est donc en $O(m * n)$.

On suppose que l'un des deux tableaux est trié, et que la recherche est une recherche dichotomique. Lequel des deux tableaux doit-on trier pour améliorer la complexité ? Quelle est alors la compléxité obtenue ?

Il faut trier `B`, le tableau dans lequel on effectue la recherche. Si chaque élément de `A` est cherché dans `B` en utilisant la recherche dichotomique, la recherche d'un élément se fera en au plus $\lg(m)$ pas. Pour les n éléments de `A` le test d'inclusion pourra donc être effectué en $O(n * \lg(m))$.

Exercice .3

On considère la fonction suivante :

```
fonction Mystère_bis(ref T: tableau[0..NMAX-1] d'entiers,
                     val N: entier): vide;
  var i,j,p: entier;
  début
        p = T[N-1];
        i = 0;
        j = N-2;
        tantque i<j faire
          tanque T[i] < p faire
            i = i+1;
          fintantque;
          tanque T[j] >= p faire
            j = j-1;
          fintantque;
          si i < j alors
            échange(T[i],T[j]);
          finsi;
        fintantque;
        échange(T[i],T[N-1]);
  fin
finFonction
```

Soit `T` le tableau contenant la suite $9, 7, 3, 5, 12, 7, 1, 6$ et `n=8`.

Décrire l'évolution du contenu du tableau `T` et des variables `i`, `j` et `p` lors de l'exécution de la fonction.

Trace de l'exécution :

```
p=6   i=0   j=6
                i <j ?? oui
                  T[0] < 6 ?? non
                  T[6] >= 6 ?? non
                  i < j ?? oui
                    échange(T[0],T[6]) donne: 1, 7, 3, 5, 12, 7, 9, 6
p=6   i=0   j=6
                i < j ?? oui
                  T[0] < 6 ?? oui
p=6   i=1   j=6
                  T[1] < 6 ?? non
                  T[6] >= 6 ?? oui
p=6   i=1   j=5
                  T[5] >= 6 ?? oui
p=6   i=1   j=4
                  T[4] >= 6 ?? oui
p=6   i=1   j=3
                  T[3] >= 6 ?? non
                  i < j ?? oui
                    échange(T[1],T[3]) donne: 1, 5, 3, 7, 12, 7, 9, 6
p=6   i=1   j=3
                i < j ?? oui
                  T[1] < 6 ?? oui
```

```
p=6   i=2   j=3
                    T[2] < 6 ?? oui
p=6   i=3   j=3
                    T[3] < 6 ?? non
                    T[3] >= 6 ?? oui
p=6   i=3   j=2
                    T[2] >= 6 ?? non
                    i <j ?? non
                  i <j ?? non (fin de la boucle tantque externe)
échange(T[3],T[7]) donne: 1, 5, 3, 6, 12, 7, 9, 7
```

À la fin de l'exécution de la fonction la suite dans le tableau à été "séparée" par rapport au dernier lélément : tous les éléments inférieurs à `p` (valeur initiale de `T[n-1]`) se retrouvent à sa gauche, tous les éléments supérieurs ou égaux à `p` se retrouvent à sa droite.