UNIVERSITE IBN TOFAIL
Faculté des sciences
Département d'Informatique
Kenitra

Année : 2019/2020
Filières : SMI
Semestre : 3

Algorithmique II

Examen de rattrapage

Corrigé

Exercice 1 : (Sur 8 points)

La procédure Tri_Simple décrite ci-dessous est probablement la méthode de tri, la plus simple à comprendre et à programmer. Elle est donnée par :
Soit T[1..N] un tableau d'entiers à trier par ordre croissant.
On parcourt le tableau T du début jusqu'à rencontrer deux éléments successifs qui ne sont pas dans le bon ordre. On permute alors ces deux éléments et on revient au début du tableau pour refaire le même travail. On continue ainsi, jusqu'à ce qu'on atteigne la fin du tableau T, ce dernier sera alors trié par ordre croissant.

A) Ecrire en pseudo-code la procédure Tri_Simple(T : Entier[1..N]).
B) Donner les étapes d'application de la procédure Tri_Simple au tableau T= [5, 2, 3, 6, 8, 7].
C) Calculer la complexité temporelle t(N) de la procédure Tri_Simple lorsque le tableau T est déjà trié par ordre croissant.

Corrigé :

A) Soit T[1..N] le tableau d'entiers à trier.

```
Procédure Tri_Simple(T : Entier[1..N])
Var     j, x : Entier
Début
        j←1
        Tant que (j<N) Faire
                Si (T[j]>T[j+1]) alors
                        x←T[j]
                        T[j]←T[j+1]
                        T[j+1]←x
                        j←1
                Sinon
                        j←j+1
                Fin Si
        Fin Tant que
Fin
```

B) Application de la procédure Tri_Simple au tableau T= [5, 2, 3, 6, 8, 7].
j=1, comme T[1]>T[2] on permute T[1] avec T[2] pour avoir :
T= [2, 5, 3, 6, 8, 7]
j=1, T[1]<T[2] ; pas de changement
j=2, comme T[2]>T[3] on permute T[2] avec T[3] pour avoir :
T= [2, 3, 5, 6, 8, 7]
j=1, T[1]<T[2] ; pas de changement

j=2, T[2]<T[3] ; pas de changement
j=3, T[3]<T[4] ; pas de changement
j=4, T[4]<T[5] ; pas de changement
j=5, comme T[5]>T[6] on permute T[5] avec T[6] pour avoir :
T= [2, 3, 5, 6, 7, 8]
j=1, T[1]<T[2] ; pas de changement
j=2, T[2]<T[3] ; pas de changement
j=3, T[3]<T[4] ; pas de changement
j=4, T[4]<T[5] ; pas de changement
j=5, T[5]<T[6] ; pas de changement
j=6, sortie de la boucle "Tant que"

C) Calcul de la complexité temporelle t(N) dans le cas où le tableau T est trié par ordre croissant

Dans ce cas, il y'a (n-1) passage dans la boucle "Tant que". D'où la complexité temporelle est :
t(N) = taffec + (N-1)*[2*tcomp + taffect + tadd] + tcomp
= C0 + C1*N, où C0 et C1 sont des constantes
Donc t(N) = θ(N)

Exercice 2 : (Sur 6 points)

On considère un tableau A de n entiers naturels, trié par ordre croissant, et x un entier naturel.

1. En utilisant un tableau intermédiaire A1 de n entiers, écrire la procédure : *Chercher(A :Entier[1..n], x : entier),* dont la complexité temporelle en pires des cas est θ(n), qui affiche deux éléments de A dont leur somme est égale à x; s'ils existent ; sinon elle affiche un message indiquant qu'il n'y a pas de couple d'éléments de A dont la somme est égale à x.
2. Montrer qu'effectivement la complexité temporelle dans les pires des cas de la procédure *Chercher(A, x)* est égale à θ(n).

Corrigé

1. Description de la procédure Chercher(A : Entier[1..n], x : Entier)

```
Procédure Chercher(A : Entier[1..n], x : Entier)
Var     T1 : Entier[1..n]
        i, j : Entier
        B : booleen
Début
   //Remplissage du tableau T1
   Pour (i←1 à n) Faire
        T1[i] ← x-T[i]
   Fin pour
   //Comme T est trié par ordre croissant alors T1 est trié par ordre décroissant.
   //On parcourt les deux tableaux A et T1 en même temps à la recherche d'un élément commun.
   i ←1
   j ←n
   B ← Faux
   Tant que (B=Faux et i<= j) Faire
        Si (A[i]=T1[j]) Alors
            B ← Vrai
        Sinon
            Si (A[i]<T1[j]) Alors
                i ← i+1
```

Sinon
j ← j-1
Fin si
Fin si
Fin tant que
Si (B=Vrai) Alors
Ecrire ("Les deux éléments de A dont la somme est égale à ",x," sont : ", A[i], " et ", x-T1[j])
Sinon
Ecrire ("Il n'existe pas de couple d'éléments de A dont la somme est égale à ", x)
Fin si
Fin

2. Complexité temporelle de la procédure Chercher(A, x)

Soit t(n) la complexité temporelle dans les pires des cas de la procédure Chercher(A, x).
Le pire des cas de la procédure Chercher est lorsque i=j. Dans ce cas on a :

$$t(n) = \sum_{i=1}^{n}(taffect + taddi) + 3 * taffect + n * (5 * tcomp + taffect + taddi) + 4 * tcomp + taffichage$$

t(n)=n*(taffect+taddi) + 3*taffect + n*(5*tcomp+taffect+taddi) + 4*tcomp + taffichage
t(n)=α*n, où α est une constante
D'où t(n)= θ(n).

Exercice 3 : (Sur 6 points)

On considère la procédure *Calcul (T : Entier[0..N], deb : Entier, fin : Entier)* donnée par :

```
Procédure Calcul (T : Entier[0..N], deb : Entier, fin : Entier)
Var     i, j : Entier
Debut
        j ← deb
        Pour  (i allant de deb à fin-1) Faire
                Si (T[i] <= T[fin]) alors
                        Echanger (T[i], T[j])
                        j ← j + 1
                Fin Si
        Fin Pour
        Echanger (T[fin], T[j])
Fin
```

Où Echanger (x, y) est la procédure qui consiste à permuter les deux éléments x et y.

1. Faites tourner la procédure Calcul(T, deb, fin) sur le tableau T suivant, avec deb=0 et fin=7.
   T=[2, 9, 3, 8, 6, 10, 4, 5]
2. Quel est le but de la procédure Calcul ?

Corrigé

1. On va faire tourner la procédure Calcul sur T=[2, 9, 3, 8, 6, 10, 4, 5], avec deb=0 et fin=7.
   j=0,
   Pour i=0, et j=0, comme T[i]<= T[fin] on permute T[i] avec T[j]. Puisqu'ils sont égaux il n'y aura pas de changement. Mais j=1
   Pour j=1 et i=1, comme T[i]> T[fin], il n'y aura pas de changement.
   Pour j=1 et i=2, comme T[i]<= T[fin] on permute T[i] avec T[j]. On obtient :
   T=[2, 3, 9, 8, 6, 10, 4, 5], et j=2

Pour j=2 et i=3, comme T[i] > T[fin] il n'y aura pas de changement.
Pour j=2 et i=4, comme T[i] > T[fin] il n'y aura pas de changement.
Pour j=2 et i=5, comme T[i] > T[fin] il n'y aura pas de changement.
Pour j=2 et i=6, comme T[i] <= T[fin], on permute T[i] avec T[j]. On obtient :
T= [2, 3, 4, 8, 6, 10, 9, 5], et j=3.
Pour i=7 et j=3, sortie de la boucle "Pour", on permute T[fin] avec T[j], on obtient :
T= [2, 3, 4, 5, 6, 10, 9, 8]

2. On constate que la procédure Calcul sert à déterminer un pivot qui sépare le tableau en deux parties les éléments à gauche du pivot sont inférieurs au pivot et ceux à droite sont supérieurs au pivot.