UNIVERSITE IBN TOFAIL
Faculté des sciences
Département d'Informatique
Kenitra

Année : 2020/2021
Filières : SMI
Semestre : 3

Algorithmique II

Examen final

Corrigé

Exercice 1 : (Sur 12 points)

Deux entiers sont dits amiables si chacun d'eux est égal à la somme des diviseurs de l'autre (par exemple 220 et 284. En effet, les diviseurs de 220 sont : 1, 2, 4, 5, 10 , 11, 20, 22, 44, 55 et 110. Leur somme est bien égale à 284 ; les diviseurs de 284 sont : 1, 2, 4, 71, 142, leur somme est bien égale à 220)

1. Ecrire une fonction *Amiable(p, q : Entier)* qui reçoit deux entiers p et q, retourne Vrai si les deux entiers sont amiables ; Faux sinon.
2. Ecrire un algorithme *RechercheAmiable* qui lit un entier N ; détermine et affiche toutes les paires de nombres amiables inférieurs à N, en utilisant la fonction *Amiable* développée en 1.

Corrigé :

1. La fonction suivante Amiable(p, q :Entier) permet de déterminer si les deux entiers p et q sont amiables. Elle retourne Vrai si c'est le cas, sinon elle retourne Faux

```
Fonction Amiable(p, q : Entier) : Boolléen
Var     S1, S2 : Entier   //pour calculer la somme des diviseurs de p et q
        i, p1, q1 : Entier
Début
        S1←1    // somme des diviseurs de p
        S2←1    // somme des diviseurs de q
        p1←p Div 2
        q1←q Div 2
        //Calcul de la somme des diviseurs de p
        Pour ( i←2 à p1, i←i+1 ) Faire
              Si (p mod i =0)  Alors
                  S1←S1+i
              Fin Si
        Fin Pour
        //Calcul de la somme des diviseurs de q
        Pour ( i←2 à q1, i←i+1 ) Faire
              Si (q mod i =0)  Alors
                  S2←S2+i
              Fin Si
        Fin Pour
        Si (p=S2 et q=S1) Alors
                Retourner Vrai
        Sinon
                Retourner Faux
        Fin Si
Fin
```

2. L'algorithme suivant RechercheAmiable détermine et affiche tous les couples d'entiers inférieurs à un entier N donné qui sont amiables en utilisant la fonction Amiable.

```
Algorithme RechercheAmiable()
Var     i, j, N : Entier
Début
        //Saisir la valeur de N
        Ecrire("Donner la valeur de N")
        Lire(N)
        Ecrire("\nLes paires de nombres amiables inférieurs à ", N, " sont:\n")
        Pour ( i←2 à N, i←i+1 ) Faire
                Pour ( j←i+1 à N, j←j+1 ) Faire
                        Si (Amiable(i,j) = Vrai) Alors  //Affichage du coupe (i,j)
                           Ecrire("(", i, ",", j,")\t")
                        Fin Si
                Fin Pour
        Fin Pour
Fin
```

Exercice 2 : (Sur 8 points)

Soient T[0..n] un tableau de réels et x un réel. On supposer que T est trié par ordre croissant. On considère la fonction f donnée par :

```
Fonction f(T : Reel[0..n], x : Reel) : Boolleen
Var     i, j : Entier
Début
        i← 0
        j← n
        Tant que (i<=n et j>=0) Faire
                Si (T[i]+T[j]=x) Alors
                        Retourner Vrai
                Sinon
                        Si (T[i]+T[j]<x) Alors
                                i←i+1
                        Sinon
                                j←-j-1
                        Fin Si
                Fin Si
        Fin Tant que
        Retourner Faux
Fin
```

1. Quel est le but de la fonction f ?
2. Déterminer la complexité temporelle de f en fonction de n.

Corrigé :

1. ***But de la fonction f***

   La fonction f cherche à déterminer s'il existe deux éléments du tableau T dont la somme est égale à x. La fonction retourne Vrai s'il existe deux éléments du tableau T dont la somme est égale à x. S'il n'y'a pas de couple d'éléments de T dont la somme est égale à x, la fonction f retourne Faux.

2. *Complexité temporelle de la fonction f*

Soit t(n) la complexité temporelle de f on a :
t(n)=2*taffect + niter*(3*tcomp + tadd + tcomp + tadd + tcomp + taffect + tadd) + 3*tcomp + tretour, où niter est le nombre de passages dans la boucle "Tant que".

A chaque passage dans la boucle "Tant que", soit i est incrémenté ou j est décrémenté (pas les deux en même temps), et comme i est initialisé à 0 et j est initialisé à n et la sortie de la boucle est réalisée dans les pires des cas quand i>n et j<0, alors le nombre de passages dans cette boucle dans les pires des cas est 2*(n+1). D'où niter=2*(n+1).
Par suite,
t(n) = C1 + 2*C2*(n+1), où C1= 2*taffect + 3*tcomp + tretour et C2= 5*tcomp + 3*tadd + taffect qui sont des constantes.
Donc finalement t(n)=θ(n).