UNIVERSITE IBN TOFAIL
Faculté des sciences
Département d'Informatique
Kenitra

Année : 2022/2023
Filières : SMI
Semestre : 3

Algorithmique II

Examen final

Corrigé

Exercice 1 : (Sur 5 points)

Soit T[1..n] un tableau de n entiers quelconques. Ecrire en pseudo-code la procédure Doublon(T: Entier[1..n]) permettant d'afficher les doublons du tableau T (c'est-à-dire les éléments qui se répètent), ceci avec leurs nombres d'occurrences. Par exemple, si T= [5,2,4,5,2,1,2,4,2], la procédure affichera (5,2), (4,2) et (2,4). Ce qui veut dire que les éléments 5 et 4 se trouvent deux fois dans T et l'élément 2 se trouve quatre fois dans T.

Corrigé

```
Procedure Doublon(T : Entier[1..n])
Var     i, j : Entier
        Occurrence : Entier[1..n]  //Pour sauvegarder les nombres d'occurrence des éléments de T
Début
        //Initialisation du tableau Occurrence
        Pour (i←1 à n-1) Faire
                Occurrence[i]←0
        Fin Pour
        Pour (i←1 à n-1) Faire
                Si (Occurrence[i]=0) Alors  //qui veut dire qu'on rencontre T[i] pour la 1ère fois
                        Occurrence[i]←1
                        Pour (j←i+1 à n) Faire
                                Si (T[j] = T[i]) Alors
                                        Occurrence[i]←Occurrence[i]+1
                                        Occurrence[j]← -1        //T[j] est déjà traité
                                Fin Si
                        Fin Pour
                Fin Si
        Fin Pour
        //Affichage des résultats
        Pour (i←1 à n-1) Faire
                Si (Occurrence[i] > 1) Alors
                        Ecrire("(",T[i],", ",Occurrence[i],")  ")
                Fin Si
        Fin Pour
Fin
```

Exercice 2 : (Sur 8 points)

On considère la procédure "Mystere" donnée par :

Procédure Mystere (A : Entier[1..n])

```
Var     i, j, x : Entier
        B : Booléen
Début
        Pour (i←1 à n-1) Faire
                B ← Vrai
                Pour (j←1 à n-i) Faire
                        Si (A[j] < A[j+1]) Alors
                                x ← A[j]
                                A[j] ← A[j+1]
                                A[j+1] ← x
                                B ← Faux
                        Fin Si
                Fin Pour
                Si (B=Vrai) Alors
                        Retourner
                Fin Si
        Fin Pour
Fin
```

1. Appliquez la procédure "Mystere" au tableau d'entiers A=[2, 1, 5, 4, 3]. Dressez un tableau, comme ci-dessous, indiquant les contenus de i, j, B et A après chaque passage dans la boucle interne ˚Pour˚.

Contenu de i	Contenu de j	Contenu de B	Contenu du tableau A
……………………	…………………	…………….	…………………………….

2. Expliquez brièvement l'effet de cette procédure sur le tableau A en entrée.
3. Donnez en fonction de la taille n du tableau A, les complexités temporelles de cette procédure, dans le meilleur et dans le pire des cas.

Corrigé

1. Application de la procédure Mystere au tableau A=[2, 1, 5, 4, 3]

Contenu de i	Contenu de j	Contenu de B	Contenu du tableau A
1	1	Vrai	[2, 1, 5, 4, 3]
1	2	Faux	[2, 5, 1, 4, 3]
1	3	Faux	[2, 5, 4, 1, 3]
1	4	Faux	[2, 5, 4, 3, 1]
2	1	Faux	[5, 2, 4, 3, 1]
2	2	Faux	[5, 4, 2, 3, 1]
2	3	Faux	[5, 4, 3, 2, 1]
3	1	Vrai	[5, 4, 3, 2, 1]

2. Effet de la procédure Mystere sur le tableau A :
   On remarque que le tableau obtenu ci-dessus est trié par ordre décroissant.
   A chaque passage dans la boucle externe "Pour" le minimum du sous tableau T[1..n-i] est déplacé à la fin de ce sous tableau ; c'est l'opération inverse de l'algorithme de tri à bulles.
   Ainsi, l'application de la procédure Mystere au tableau A permet de le trier dans l'ordre décroissant.
3. Complexités temporelles de la procédure Mystere :

- Dans le pire des cas, on passe dans la boucle externe "Pour" (n-1) fois et pour chaque $i^{ème}$ passage dans cette boucle (avec $1 \le i \le n-1$) on passe (n-i) fois dans la boucle interne "Pour". D'où dans le pire des cas la complexité temporelle de cette procédure est :

  t1(n) = $\sum_{i=1}^{n-1}(taffect + \sum_{j=1}^{n-i}(tcomp + 4taffect) + tcomp))$

  t1(n)= $\sum_{i=1}^{n-1}(taffect + (n-i)(tcomp + 4taffect) + tcomp))$

  t1(n)= $\sum_{i=1}^{n-1}(taffect + tcomp) + \sum_{i=1}^{n-1}((n-i)(tcomp + 4taffect))$

  t1(n) = (n-1)(taffect $+tcomp$) + $\frac{n(n-1)}{2}(tcomp + 4taffect)$

  t1(n) = $\theta(n^2)$
- Dans le meilleur des cas, on passe une seule fois dans la boucle externe "Pour" et (n-1) fois dans la boucle interne "Pour". D'où dans le meilleur des cas la complexité temporelle de cette procédure est :

  t2(n) = $taffect + \sum_{j=1}^{n-1}(tcomp + 4taffect) + tcomp$

  t2(n) = $taffect + tcomp + (n-1)(tcomp + 4taffect)$

  t2(n) = $\theta(n)$

Exercice 3 : (Sur 7 points)

On considère la fonction récursive f donnée par :

```
Fonction f ( a : Réel , b : Entier ) : Réel
Début
        Si (b=1) Alors
                Retourner  a
        Sinon
                Retourner (a + f( a , b−1)
        Fin Si
Fin
```

1. Déterminer en fonction de a et b la valeur de f(a, b) pour un réel a et un entier b quelconques, justifier votre réponse en effectuant un raisonnement par récurrence sur b..
2. En déduire une fonction itérative équivalente à la fonction f.
3. Déterminer la complexité temporelle de la fonction f.

Corrigé

1. f(a,1)=a
   f(a,2)=a+f(a,1)=a+a=2*a
   f(a,3)=a+f(a,2)=a+2*a=3*a
   On montre par récurrence sur b que : f(a, b)= b* a, pour $b \in IN^*$ et $a \in IR$.
2. Une fonction itérative équivalente à f est donc donnée par :

   ```
   Fonction f-itérative( a : Reel, b : Entier) : Reel
   Début
       Retourner b*a
   Fin
   ```

3. Il est clair que la complexité temporelle de f(a, b) dépend uniquement de b.
   Soit donc t(b) la complexité temporelle de f(a,b), on a :
   t(b) = tcomp + tretour, si b=1
   et t(b) = tcomp + tretour + taddi + t(b-1), si b>1
   D'où la relation de récurrence :
   t(b) = C0, si b=1

t(b) = C1 + t(b-1), si b>1 avec C0 et C1 deux constantes réelles.

Ainsi, pour b>1 on a :

t(b)   = C1 + t(b-1)
t(b-1)   = C1 + t(b-2)
t(b-2)   = C1 + t(b-3)
----
t(2)   = C1 + t(1)

En effectuant la somme terme à terme on obtient après élimination des termes identiques des deux côtés :

t(b) = (b-1)C1 + C0

D'où finalement la complexité temporelle de f(a, b) est θ(b)