Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique – Département d'Informatique
1er année Master « Sécurité des Systèmes Informatiques » SSI - 2016/2017.

04 Décembre 2016

Complexité et Algorithmique avancée
« Contrôle » durée 45 mn

Exercice 1: *(07 pts)*

Déterminer l'invariant des boucles pour chacun des algorithmes suivants et en déduire l'ordre de complexité. Justifier votre réponse

Algo_A1()
```
Début
Pour i := 1 à N
        Faire  Pour j := i+1 à N
                Faire k := N ;
                Tant que k >= i
                        Faire k = k-1 ;
                        Fait;
                Fait ;
        Fait ;
Fin.
```

Algo_A2()
```
Début
x:= N;
Tant que (x > 1)
Faire   x := √x
Fait ;
Fin.
```

Algo_A3 (N)
```
Début
Si N < B alors Retourner (-1) ;
        Sinon Si N = B
                Alors Retourner (0) ;
                Sinon
                Retourner( A3 (N – A)) ;
                Fsi ;
        Fsi ;

Fin.
```
Avec A, B deux entiers qui peuvent être positifs ou négatifs.

Solution : **(2, 2, 3 pts)**

- L'algorithme A1 comporte 3 boucles imbriquées. Deux boucles externes « Pour » avec borne inférieure, supérieure et pas clairement défini.

La boucle externe s'exécute en N itérations faisant varier l'indice i de 1 à N.

Pour chaque valeur de i la seconde boucle s'exécute en N-i en faisant varier l'indice j de i+1 à N.

Pour chaque valeur de j la boucle la plus interne fait varier un indice k de N à i.

Autrement dit,

Pour i =1, la seconde boucle devra faire N-1 itération et à chaque itération de la seconde boucle la boucle la plus interne devra itérer N-i+1 fois.

Donc, pour chaque itération de la boucle externe les deux boucles internes coutent (N-i)*(N-i+1).

Donc le nombre d'itérations totale = $\sum_{i=1}^{N}(\mathrm{N}-\mathrm{i}) * (\mathrm{N}-\mathrm{i}+1) = \sum_{i=1}^{N}(N-i)^2 + N - i$

$= \sum_{i=1}^{N}(N-i)^2 + \sum_{i=1}^{N} N - i$

$$= \sum_{i=1}^{N-1} i^2 + \sum_{i=1}^{N-1} i = O(N^3) + O(N^2)$$

Ainsi l'algorithme A1 est de l'ordre de $O(N^3)$.

- Le nombre d'itération de l'Algorithme A2 correspond au nombre de fois que la racine est calculé pour atteindre en partie entière 1.

Donc en réalité la valeur de X à l'issu de la dernière itération doit être comprise entre 2 et 1 pour que la partie entière soit égale à 1 et que la terminaison de la boucle soit garantie

Autrement dit :

$$1 < (((N^{\frac{1}{2}})^{\frac{1}{2}}) \ldots\ldots)^{\frac{1}{2}} < 2 \Rightarrow 1 < N^{\frac{1}{2^k}} < 2 \Rightarrow \log 1 < \log N^{\frac{1}{2^k}} < \log 2$$

$$\Rightarrow 0 < \frac{1}{2^k} \log N < \log 2$$

$$\Rightarrow 0 < \frac{1}{2^k} < \frac{\log 2}{\log N}$$

$$\Rightarrow K < \frac{1}{log2} \log\log N - loglog2$$

$$\Rightarrow La\ complexit\acute{e}\ est\ en\ O(loglog\ N).$$

La limite peut également être calculée autrement comme suit :

$$1 < N^{\frac{1}{2^k}} < 2 \Rightarrow 1 < N < 2^{2^k} \Rightarrow K < \log\log(N)$$

- L'algorithme A3 est une fonction récursive qui dépend de N. Le comportement des appels récursifs est influencé par la relation qui peut exister entre A et B.

L'équation de récurrences associée est :

$$\begin{cases} \mathrm{T(N)} = \mathrm{T(N-A)} + \mathrm{C}; \mathrm{N} > \mathrm{B} \\ \mathrm{T(B)} = 1 \\ T(N < B) = 1 \end{cases}$$

Après résolution de cette équation de récurrence par substitution :

T(N) = T(N - (k*A)) + k*C. K étant le nombre d'itérations

Pour garantir la terminaison du processus : N - k*A <= B

Plusieurs cas possible en fonction de A et B.

1. A et B > 0 ⇒ k <= N/A
2. A et B < 0, cas impossible sinon la condition d'arrêt ne sera jamais atteinte
3. A > 0 et B < 0, k <= N/A + |B|/A
4. A <0 et B >0, si N <B alors k=1 sinon cas impossible puisque la condition d'arrêt ne sera jamais atteinte.

Puisque A et B sont constant et en vue des cas possibles, l'algorithme A3 est en O(N).

<u>Exercice 2:</u> (08 pts)

Soit en entrée un tableau binaire. On s'intéresse au problème de compression de données qui réduit un tableau en entrée de dimension N vers un tableau de dimension M comme suit :

1 1 0 0 0 0 1 0 0 1	1 2 4 1 2 1

Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique – Département d'Informatique
1er année Master « Sécurité des Systèmes Informatiques » SSI - 2016/2017.

0 1 0 0 0 1 1 1 0 1		0 1 1 3 3 1 1

1. Exprimer M (la taille du tableau après compression) en fonction de N (taille du tableau en entrée) **(1 pt)**

M = K + 1, sachant que K représente le nombre de séquence possible.

1<K<N ; 1 lorsque le tableau représente une seule séquence (que des 0 ou bien que des 1), et N lorsque le tableau contient N séquence de langueur 1 chacune (Alternance de 0 et de 1).

Ainsi 2 <= M <= N+1.

2. Proposer un algorithme de compression. (4 pts)

```
Compression (Entrée :T1 ; Sortie T2) ;
Début
Int i, j,C,nb ;
i = 1 ; j = 2 ;
T2[1] = T1[1] ; //Premier bit
C = T1[1] ; nb=0 ;
Tantque (i <=N)
Faire
        Tant que (T1[i] = C et i <=N)
                Faire nb = nb +1 ; i = i+1 ;fait ;
        T2[j]=nb ; j = j+1 ; C = T1[i] ; nb=0 ;
Fait ;
Fin.
```

a. Déterminer le pire cas et en déduire l'ordre de complexité correspondant **(1 pt)**

Au pire cas, le nombre de séquence est égal à N $\Rightarrow O(N)$

b. Déterminer le meilleur cas et en déduire l'ordre de complexité correspondant **(1 pt)**

Au meilleur cas, le nombre de séquence est égal à 1 $\Rightarrow O(N)$

3. En déduire les complexités au meilleur cas et au pire cas de l'algorithme de décompression. **(1 pt)**

Idem que l'algorithme de compression, la complexité est linéaire par rapport à la taille du tableau binaire.