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 - 2011/2012.

03 décembre 2012

Complexité et Algorithmique avancée
« Contrôle » durée 1h

Exercice 1 : (5 pt)

Soit T un tableau d'entier de dimension N.

Écrire un algorithme qui, étant donné un tableau T en entrées, supprime les doublons, ce qui signifie éliminer toutes les occurrences qui se répètent dans le tableau en ne laissant qu'un seul exemplaire à chaque fois, comme le montre l'exemple suivant :

1	3	4	3	1	1	4	0	3	→	1	3	4	0

1. Déterminer le pire cas et en déduire la complexité de l'algorithme proposé ? Justifier votre réponse.
2. Y a-t-il un moyen de réduire la complexité de l'algorithme proposé ? si oui comment ?

Solution

```
Algorithme doublons

Début

Pour i :=1 à N-1 faire

        Pour j :=i+1 à N faire si T[j] = T[i] alors /*effectuer un décallage*/

                Pour k :=j à N-1 faire T[k] :=T[k+1] ; fait ;

                N := N-1 ;

        Fsi ;

Fait ;

Fait ;

Fin.
```

Le pire cas :

Pour déterminer le pire cas, analysons d'abord l'algorithme proposé :

Chaque élément du tableau, doit être vérifié avec tous les éléments restants du tableau afin de détection d'éventuel doublon, si un doublon est trouvé, tous les éléments qui le suivent dans le tableau, et ainsi de suite, jusqu'à ce qu'on au parcourus tout le tableau.

Bonne Chance !

Dans cet algorithme, deux actions font l'objet des boucles :

- La vérification des doublons
- Les décalages.

Pour déterminer le pire cas, nous devons considérer ces deux actions :

- Par rapport à la vérification des doublons, le pire cas correspond à un tableau sans doublons, aucun élément à supprimer on effectue uniquement des vérifications inutiles. Chaque élément du tableau est vérifier avec les (N-i) éléments restants
- Par rapport aux décalages, qu'il y'ai autant de doublons que d'éléments utile dans le tableau, ce qui signifie, chaque élément possède un doublon et que les éléments utile se trouvent d'un côté du tableau, et les doublons de l'autre.

En déduire la complexité :

- Par rapport à la vérification et son pire cas : le nombre d'itération correspond à $\sum_{i=1}^{n-1} i$, ce qui implique une complexité de l'ordre de $O(N^2)$.
- Par rapport aux décalages, le nombre d'itération ici correspond à :
    - N/2 : actions de décalages qui correspond à N/2 doublon, ce qui signifie que ces doublons vont être supprimés un par un jusqu'à ce qu'il ne reste dans le tableau que les N/2 élément utile, ce qui signifie qu'à chaque itération l'algorithme va supprimer un doublon et décaler le reste du tableau d'une case ce qui correspond à la somme suivante : N/2 (le nombre de déplacement pour trouver le premier doublon) + N/2 -1 (le nombre d'élément à décaler) + (N/2-1+ N/2-2) +(N/2-2 + N/2-3) + … + 1, qui se traduit par la formule de sommation suivante :

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

Existe-il un moyen de réduire la complexité ?

- Possible, en utilisant un tableau supplémentaire. Cette proposition permet de réduire la complexité dans le cas ou des doublons existent puisque la vérification ne va plus se faire avec tout le tableau mais avec uniquement les éléments déjà insérés dans le tableau auxiliaire, mais cette proposition ne réduit pas la complexité pour le cas où tous les éléments sont distincts.

Bonne Chance !

Exercice 2 : (3 pt)

Considérons l'algorithme suivant :

```
Algorithme Algo
    Pour i :=1 à N faire
        Algo1(N) ;
        Algo2(N) ;
    Fait ;
    Algo3(N) ;
Fin.
```

Sachant que Algo1 est de complexité de l'ordre de O(√n), Algo2 est de l'ordre de O(√n log n) et Algo3 est de l'ordre de O(N), Quelle est la complexité de l'algorithme générale (Algo) ? Justifier votre réponse.

Complexité de Algo1 = O(√n)

Complexité de Algo2 = O(√n log n)

Complexité de Algo3 = O(N)

Solution

En tenant compte de l'algorithme ci-avant, la complexité de l'algorithme globale est :

Complexité de Algo = N*(O(√n) + O(√n log n)) + O(N)

= N*√n + N*√n log n + N

= N*√n log n + N*√n + N

Pour déterminer d'ordre de complexité il faut trouver la fonction la plus dominante ici

Log n > 1 pour n>10 => √n log n > √n => N*√n log n > N*√n

D'ici on peut déduire que la complexité de Algo = O(N*√n log n).

Exercice 3 : (2 pt)

Donner l'ordre de grandeur des expressions suivantes

a. $\frac{n^2+3n+1}{n+1} = \frac{n^2}{n+1} + \frac{3n}{n+1} + \frac{1}{n+1} = O\left(\frac{n^2}{n+1}\right) = O(n)$

b. $\frac{n\log(n)+n^2+\log(n)^2}{n+1} = \frac{n\log(n)}{n+1} + \frac{n^2}{n+1} + \frac{\log(n)^2}{n+1} = O\left(\frac{n^2}{n+1}\right) = O(n)$

Bonne Chance !