Université Mohamed Khider Biskra
FSESNV
Département d'informatique

Module : CO (M2-IVA)
Date: 14/01/2020
Lieu: Salle 7

EXAMEN

1 Exercice (05 points)

Soient A un tableau composé de N entiers et a et b deux réels tel que a<b.

1. On vous demande d'écrire une fonction opérant selon le paradigme diviser pour régner et permettant de chercher, dans le tableau A, le nombre d'éléments dont la valeur est encadrée par les deux valeurs a et b.
2. Donner le type de récursivité?
3. Donner la relation de récurrence qui décrit le nombre d'appels récursifs?
4. Ecrire une fonction itérative issue de la dérécursivation de la fonction trouvée pour la question 1.

2 Exercice (04 points)

Le but de cet exercice est de tester votre compréhension de la notion de complexité dans le pire des cas, en justifiant brièvement:

1. Si je prouve que la complexité dans le pire des cas d'un algorithme est en $O(n^2)$, est-il possible qu'il soit en O(n) sur certaines données?
2. Si je prouve que la complexité dans le pire des cas d'un algorithme est en $O(n^2)$, est-il possible qu'il soit en O(n) sur toutes les données?
3. Si je prouve que la complexité dans le pire des cas d'un algorithme est en $\Theta(n^2)$, est-il possible qu'il soit en O(n) sur certaines données?
4. Si je prouve que la complexité dans le pire des cas d'un algorithme est en $\Theta(n^2)$, est-il possible qu'il soit en O(n) sur toutes les données?

3 Exercice (06 points)

1. Soient les deux programmes récursifs suivants, estimer l'ordre de grandeur de la complexité temporelle en évaluant l'ordre de grandeur du nombre d'appels récursifs effectués, en justifiant brièvement:
   (a) int A1(int n)
   if (n>2) return 1+A1(n-3);
   else return 0;
   (b) int A2(int n)
   if (n>1) return 1+A2(n-2)*A2(n-2));
   else return 0;

2. Comment calculer la complexité des algorithmes?

3. Estimer l'ordre de grandeur de la complexité temporelle de l'algorithme suivant:
Algorithme: f1(n)
Données: un entier n 0
Résultat: ?
r,i,j: entier;
début
r $\leftarrow$ 0;
pour i allant de 1 à n faire
pour j allant de i+1 à n faire
$r \leftarrow r + 1$;
$retourner(r)$;
$fin.$

4 Exercice (05 points)

1. Evaluer l'ordre de grandeur de la fonction suiviante :

$$\text{f(n)} = \begin{cases} \frac{2^n}{3}, \text{si n est un carré parfait;} \\ \frac{n^2}{2}, \text{si n est pair mais pas un carré parfait;} \\ 4n, \text{ si n est impair mais pas un carré parfait.} \end{cases}$$

2. Considérons les équations de récurrence :

$$\begin{cases} T(1) = 1 \\ n \geq 2, T(n) = T(\frac{n}{2}) + H(n) \\ \\ H(1) = 1 \\ n \geq 2, H(n) = H(n-1) + n \end{cases}$$

(a) Evaluer l'ordre de grandeur de T(n) de deux manières différentes.

3. Soit le code suivant :
a $\leftarrow n; j \leftarrow 0$;
tant que a $\neq 0 faire$
si a est impair alors a $\leftarrow a - 1$;
$finsi$;
$a \leftarrow \frac{a}{2}; j \leftarrow j + 2$;
$fintq$;

(a) Exprimer j puis a en fonction de n et déterminer la complexité de l'algorithme.

Bon courage