Master1 en informatique
Le 07/05/2018 Durée : 1 : 30

Examen du 2ème semestre en Complexité des Algorithmes

UNIVERSITE HAMMA LAKHDAR EL-OUED
FACULTE DES SCIENCES EXACTES.

Correction Type

Exercice1 :

1) O-notation
La notation grand O indique « l'ordre de grandeur » des fonctions, elle est utilisé pour mesurer l'efficacité d'un Algorithme/programme en terme de Temps (d'exécution) et Espace (mémoire).
2) Principe d'invariance
La complexité d'un algorithme est indépendante de son implementation (choix de la machine, du langage,...).
3) Réductions polynomiales
Une réduction polynomiale du langage L1 vers le langage L2 (on écrit L1 ≤ L2), s'il existe une fonction f de complexité polynomiale telle que $x \in L1$ ssi $f(x) \in L2$.
4) Classe P
C'est l'ensemble des problèmes de décision pouvant être résolus en temps polynomial, par des algorithmes déterministes.
5) Classe NP
C'est l'ensemble des problèmes de décision pouvant être résolus en temps polynomial, par des algorithmes non déterministes.
6) Classe NP-complet
Nous dirons qu'un problème est NP-complet s'il appartient à NP et s'il est aussi « difficile » que n'importe quel problème de NP.

1pt X 6

2pt X3

1) **O(f(n)) +O(g(n)) =O(max(f(n), g(n)))**
Preuve : soit f(n)> g(n) i.e dans le pire des cas O(f(n)) +O(g(n)) = O(2f(n)) = O(f(n)) (2 est un constant)
2) **3-SAT est NP-complet**
Preuve en applique Réductions polynomiales 3-SAT vers SAT (en utilisant la règle : $(a \vee b) \wedge (\bar{a} \vee c) \Leftrightarrow (b \vee c)$)
3) **Le problème du voyageur de commerce (TSP) est NP-complet**
Preuve a\ HAMILTON ≤ TSP b\ 3-SAT ≤ HAMILTON c\ SAT ≤ 3-SAT ≡ TSP est NP-complet

Exercice 2 :

$$\begin{cases} T(0) = a & \text{si } n == 0 \\ T(n) = b + T(n-1) & \text{sinon} \end{cases}$$

T(n) = T(n-1) + b = T(n-2) + 2b
= T(n-3) + 3b = ...
= a + nb ≡ O(n)

1.5pt

$$T(n) = 2 * T\left(\frac{n}{2}\right) + n$$

T(n) = 2T(n/2) + n
<= 2T(n/2) + cn
<= 4T(n/4) + (2 c n/2 + c n) = 4T(n/4) + 2cn
<= 2^k T(n/2^k) + cnk <= ...
<= $2^{\log n}$ T(1) + c n log n ≡ O(n log n)

1.5pt

Exercice 3 :

A)
```
m = n div 2;
Pour (j=1; j<=n*m; j=j++)
{
   Tant que (m>0)
   {
     m =m div 2
     Afficher ("Salam");
   }
}
```
O(3/2n * log (n/2))

2pt

B)
```
m=n div2 ;
k=(m mod 2)+ 1 ;
Tant que (m>0)
{
  Pour (j=1; j<=n*n; j=j+2)
  {
    Afficher ("Salam");
  }
  m=m-k ;
}
```
Si m ≡ 0 [2] ==> O(n^2/2 * n/2)
Si m ≡ 1 [2] ==> O(n^2/2 * n/4)

1.5pt

C)
```
Tant que (m> 0)
   {  Pour (j=1; j<=n; j++)
      { Afficher ("Salam");
        Si (n mod 3 ==0 )
           j =j+ n div 2 ;
        Sinon
           j =j + 3
      }   m=m-3 }
```
Si n ≡ 0 [3] ==> O(m/3)
Si n ≡ 1 [3] ==> O(m /3 * n/4)

1.5pt