Niveau : M2 d'Informatique
Date : 09/03/2021
Durée : 1H
Responsable de matière : Dr. S.Slatnia

Université Mohamed Khider Biskra
FSE&SNV

Examen: Complexité et Optimisation

Nom : ……
Prénom : ……..
……

Exercice 1– (04 points)

Lesquelles des assertions suivantes sont vraies.

1. $n^3 \in \Omega(n^2)$	……
2. $2^{n+1} \in O(2^n)$	……
3. $(n+1)^2 \in \theta(n^2)$	……
4. $n^2 \in \Omega(n^3)$	……
5. $T(n)=2T(\frac{n}{2})+ n^2 \Rightarrow T(n)=\Theta(n^2)$	…….
6. $T(n) = 2T([\frac{n}{4}]) + 3\log n \Rightarrow T(n)=\Theta(\sqrt{n})$	……
7. $T(n) = T(n-1) + O(\log n) \Rightarrow T(n) = O(n.\log n)$	……
8. $T(n) = 2T([\frac{n}{4}]) + 3\log n \Rightarrow T(n)=\Theta(\log n)$	……

Exercice 2– (07 points)

1. Comment calculer la complexité des algorithmes?
2. Calculer la complexité des algorithmes suivants par rapport aux opérations arithmétiques?

Algorithme E	Algorithme F
Début 1. s←0 ; 2. i←1 ; 3. tan que i≤n faire 4. pour j ← n^2 à 5 faire 5. s ←s+1; finpour ; 6. i ←i+2; fintantque; 7. return s Fin.	Début 1. Pour i de 1 à n-1 faire 2. tmp←i ; 3. pour j de i+1 à n faire 4. Si T[j]<T[tmp] alors 5. tmp←j; finpour ; 6. T[i] ↔T[tmp] ; finpour ; Fin.

3. Prouver que la complexité temporelle de l'algorithme de tri classique et l'algorithme de tri avancé est $O(n^2)$ et $O(n\log n)$ respectivement ?

Exercice 3– (05 points)

1. A quoi sert l'étude des heuristiques?
2. Le problème du voyageur de commerce, il consiste à visiter un nombre N de villes en un minimum de distance sans passer deux fois par la même ville.

Question. Modéliser ce problème à partir d'un algorithme génétique et des divers opérateurs.

Exercice 4– (04 points)

Quel est l'ordre de grandeur de ces fonctions suivantes ?

a. $f(n) = 12n+7$

b. $f(n) = a_0\ n^p + a_1\ n^{p-1} + ... + a_{p-1}\ n + a_p$

c. Soit I le nombre de passages dans la boucle d'un algorithme A. On va calculer I. A chaque fois on divise la taille de l'espace de recherche (f-place+1) en deux et on s'arrête quand il ne contient plus qu'un seul élément.

Bonne chance

Corrigé type CO

Exercice 1– (04 points)

vraies: 1,2,3,5,6,7,8

fausse: 4

Exercice 2– (07 points)

1. voir le cours
2. par rapport aux opérations arithmétiques

• l'algorithme E : w(A) = w(1) + w(2) + w(3) + w(7)

$= 0 + 0 + 0 + \sum_{i=1}^{n/2}(w(4) + w(6))$

$$= \sum_{i=1}^{\lceil \frac{n}{2} \rceil} \sum_{j=5}^{n^2} w(5) = \sum_{i=1}^{\lceil \frac{n}{2} \rceil} \sum_{j=5}^{n^2} 1$$

$$= \sum_{i=1}^{\lceil \frac{n}{2} \rceil} (n^2 - 5 + 1) = \sum_{i=1}^{\lceil \frac{n}{2} \rceil} (n^2 - 4)$$

$$= \lceil \frac{n}{2} \rceil \cdot (n^2 - 4) = O(n^3)$$

• l'algorithme F :

$$\sum_{i=1}^{n-1}\left(4 + 2 \cdot \sum_{j=i+1}^{n} 1\right) = 2 \cdot \left(\sum_{i=1}^{n}(n-i)\right) + 4 \cdot (n-1)$$

$$= 2 \cdot \left(\sum_{i=1}^{n} n\right) - 2 \cdot \left(\sum_{i=1}^{n} i\right) + 4 \cdot (n-1)$$

$$= 2n(n-1) - 2 \cdot \frac{n \cdot (n-1)}{2} + 4 \cdot (n-1) = n(n-1) + 4(n-1)$$

$$= O(n^2)$$

3.

a. Algorithme TRI-SELECTION(T : tableau [1..N] d'entiers, entier N)

```
Var
i, j, min : entier;
Début
    1. pour i ← 1 à N - 1 faire
    2. min ← i ;
    3. pour j ← i + 1 à N faire
    4.    si (T[j] < T[min]) alors
    5.          min ← j ;  finsi ;  finpour ;
    6. échange (T[i]; T[min]) ;
    7. finpour ;
Fin.
```

w(A)=w(1)

$= \sum_{i=1}^{n-1}(w2 + w3 + w6) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n}(w4 + w5)$

$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n}(O(1)) = \sum_{i=1}^{n-1}(n - (i+1) + 1) = \sum_{i=1}^{n-1}(n - i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i$

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

=> la complexité est $\Theta(n^2)$

- **Prouver que la complexité** de l'algorithme de tri par fusion = O(nlog n) :

b. Algorithme TRI-FUSION (T : tableau [1..N] d'entiers ; N , l, r : entiers)

```
Var
m: entier;
Dédut
   si (l < r) alors    m ← [(l + r)/2] ;  finsi ;
   TRI-FUSION(T ; l ; m) ;
   TRI-FUSION(T; m+1, r) ;
   fusion(T; l; r; m) ;
fin.
```

- La complexité de l'algorithme récursif est donnée par son équation,

Le coût de la récursivité est donné par T(m) tel que :

T(m) = α cas de base

T(m) = c.T(f(m)) + g(m) équation de récurrence

α : temps d'exécution dans le cas de base.

c : nombre de fois qu'on fait un appel récursif = 2.

g(m), le coût d'un appel récursif) => tri par fusion (le cout de l'utilisation des registres temporaire est linéaire) => g(m)=O(n) => T(m) = 2.T([n/2]) + O(n) d'après les relations de récurrences (Cours) :
$T(n) = c.T(n/d) + O(n^k)$
c=2, d=2 et k=1 => $c=d^k$, alors T(n) = O(nlog n)

Exercice 3– (05 points)

1. Voir le cours
2. Voir le cours
   - Phase de modélisation (individu, population et fitness)
   - Phase d'évolution (croisement, mutation ,reproduction et critères d'arret)

Exercice 4– (04 points)

a.

Pour c, il faut une valeur supérieure à 12. Si on prend 13 (par exemple), on cherche alors n_0 tel que pour n supérieur à n_0, on ait $12n+7 \leq 13n$, soit $n \geq 7$.

On pourrait aussi prendre $n_0 = 1$ et $c = 19$. **f(n) est en O(n).**

b.

$f(n) = a_0 n^p + a_1 n^{p-1} + ... + a_{p-1} n + a_p$

On suppose $a_0 > 0$.

Si tous les coefficients sont positifs, on peut alors choisir :

$c = a_0 + a_1 + ... + a_p$, avec $g(n) = n^p$

On a bien $f(n) <= (a_0 + a_1 + ... + a_p) n^p$

Si des coefficients sont négatifs, il suffit pour c de faire la somme des coefficients positifs. **Donc f est en O(n).**

c. "On a un seul élément à l'itération I" se traduit par $n/2^I = 1$.

Donc $n = 2^I$

$\log_2(n) = \log_2(2^I) = I$

$I = \log_2(n)$

L'algo est donc en **O(log2(n)).**