جامعة هواري بومدين للعلوم و التكنولوجيا

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique

Département d'Informatique

Concours d'accès au Doctorat 3 ième Cycle Informatique 2019 – 2020

Le 26/10/2019

Matière 1 : Algorithmique avancée et complexité,
Coefficient **1**, durée **1 h 30**
(Spécialités : IA, MFA, SIGL)

Exercice 1 : (8 points)

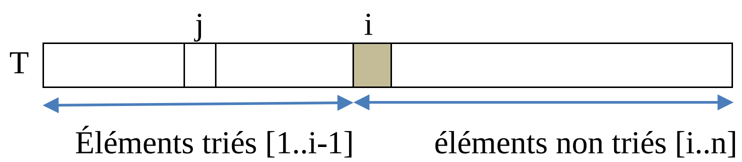
a. Donnez les définitions des notations Landau O, Θ et Ω.

b. Les affirmations ci-dessous sont-elles vraies ou fausses ? Si vous pensez qu'une affirmation est fausse, indiquez pourquoi et corrigez l'affirmation. (Reproduire, sur la copie, le tableau ci-dessous et le compléter).

Affirmation	Vraie ou Fausse?	Affirmation correcte
Si $f(n) \in O\big(g(n)\big),\ alors\ g(n) \in O\big(f(n)\big)$.		
Soit $S(n) \in O\big(f(n)\big), T(n) \in O\big(g(n)\big)$ Si $f(n) \in O\big(g(n)\big),\ alors\ S(n) + T(n) \in O\big(f(n)\big)$.		
Si $f(n) \in O\big(g(n)\big), alors\ f(n) + g(n) \in O\big(g(n)\big)$		
La meilleure borne asymptotique de $O(f(\mathrm{n})) + O(g(\mathrm{n}))$ est $O(f(\mathrm{n}) + g(\mathrm{n}))$		
Si $g = O(f(\mathrm{n}))$ et $h = O(g(\mathrm{n}))$ alors $h = O(f(\mathrm{n}))$		
$5n + 8n^2 + 100n^3 \neq O(n^4)$		
$100n + \log n = \Theta(n + (logn)^2)$		
$\sqrt{n} = O((logn)^2)$		

Exercice 2 : (12 points) Tri par insertion

Le tri par insertion d'un tableau T[1..n] de n éléments consiste à insérer chaque élément à sa place. Le premier élément constitue, à lui tout seul, une suite triée de longueur 1. On range ensuite le second élément pour constituer une suite triée de longueur 2, puis on range le troisième élément pour avoir une suite triée de longueur 3 et ainsi de suite...
Le principe du tri par insertion est donc d'insérer à la i-ème itération le i-ème élément à la bonne place. Il faut pour cela trouver où l'élément doit être inséré en le comparant aux autres, puis décaler les éléments afin de pouvoir effectuer l'insertion.



1. Ecrire l'algorithme « Tri_insertion1 » du tri par insertion tel que décrit précédemment. Donner sa complexité en nombre de comparaisons.

2. Puisque la suite d'éléments, dans laquelle on cherche le rang d'insertion, est triée (T[1..i-1]), on propose d'utiliser la recherche dichotomique.
    a. Donner l'algorithme « Tri_insertion2 ».
    b. Donner sa complexité en nombre de comparaisons.
    c. La complexité du tri a-t-elle été améliorée ?

3. Soit l'algorithme récursif suivant :

```
Procédure Tri_insertion3(E/ T : tableau[n] entier ; p : entier)
Var k : entier ;
Début
        Si (p>0) alors
                Tri_insertion3(T, p-1) ;
                k←p ;
                tantque (k>1 et T[k-1]>T[k])
                      faire
                          permuter(T[k-1], T[k]) ;
                          k←k-1 ;
                      fait ;
        fsi ;
Fin ;
```

L'appel initial se fait par Tri_insertion3(T, n).

a. Dérouler l'algorithme avec n=8 et T=[6| 3| 7| 4| 2| 8| 1| 5]
b. Donner l'équation de récurrence qui décrit le temps d'exécution de l'algorithme. Résoudre l'équation.

4. Que constatez-vous (comparer les trois algorithmes) ?

Bon courage