INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للإعلامية

Section	*1ING*
Matière	*Algorithmes Avancés*
Enseignants	*F. Kebair, R.Guetari*

TD1 : Correction Complexité algorithmique

Exercice 1

Calculer la complexité des fragments de code suivants :

1. Structures conditionnelles + boucles imbriquées sans dépendance des indices

```
Pour i de 1 à N faire                          nc1
    R = R + X + Y + Z ;                        nc2
    Si T[i] + K < B alors                      nc3
            Pour j de 1 à N faire              n^2c4
              R = R + T[j] ;
            Fin Pour
    Sinon
            R = R + T[i] ;                     nc5
    Fin Si
Fin Pour
```

T(n) = nc1 + nc2 +n c3 + n^2c4 + nc5, donc la complexité asymptotique est de O(n^2)

2. Structures conditionnelles + boucles imbriquées avec dépendance des indices

```
Pour i de 1 à N faire                          nc1
    R = R + X + Y + Z ;                        nc2
    Si T[i] + K < B alors                      nc3
            Pour j de 1 à i faire              c4 Σ_{i=1}^{n} i = n(n+1)/2
              R = R + T[j] ;                   c5 Σ_{i=1}^{n} i = n(n+1)/2
            Fin Pour
    Sinon
            R = R + T[i] ;                     nc6
    Fin Si
Fin Pour
```

$$c4\sum_{i=1}^{n} i = n(n+1)/2 \qquad c5\sum_{i=1}^{n} i = n(n+1)/2$$

T(n) = nc1 + nc2 + nc3 + c4 x n(n+1)/2 + c5 x n(n+1)/2 + nc6, donc la complexité asymptotique est de O(n^2)

Section	*1ING*
Matière	*Algorithmes Avancés*
Enseignants	*F. Kebair, R.Guetari*

Exercice 2

```
i  = n
S  = 0
Tant que (i > 0) faire
  j = 2*i
  Tant que (j > 1) faire
    S  = S + (j-i)* (S+1)
    j = j-1
  Fin TQ
  i  = i div 2
Fn TQ
```

La boucle interne (TQ j > 1) fait 2*i itérations.
Pour i=n la boucle interne fait (2n - 1) itérations,
Pour i=n/2, elle fait (n -1) itérations,
puis n/2 - 1,
n/4 - 1,
n/8 -1,
:
2-1.

$T(n) = 2n-1 + n-1 + n/2 - 1 + n/2^2 - 1 + n/2^3 - 1 + ... + 2-1 = 4n - \log_2 n$ donc la complexité asymptotique est de O(n).

Exercice 3

```
I = 1
Tant que (i < n) faire
  j = 1
  Tant que (j < 2*n) faire
     j = j*2
  Fin TQ
  i = i+1
Fin TQ
```

La boucle interne fait environ 2 $\log_2 n$ itérations, car j varie de 1 à 2n par progression logarithmique (j = j*2). La boucle externe fait n itérations, donc les deux boucle imbriquées font n * 2$\log_2 n$ itérations.
Les instructions, autre que les boucles, sont des affectations (o(1)), donc la complexité asymptotique de l'algorithme entier est en **O(nlog n).**

Exercice 4

1. Calculer la complexité de l'algorithme itératif de recherche dichotomique.

```
Fonction rechercheDichotomique(entier[] T, entier x) : entier
```

Section	*1ING*
Matière	*Algorithmes Avancés*
Enseignants	*F. Kebair, R.Guetari*

```
Début
    entier i, j;
    i = 0; j = T.longueur-1;
    tant que (i <= j) faire
            Si (T[(j+i)/2] = x)
                    alors retourner (j+i)/2;
            Sinon
                    Si (T[(j+i)/2] > x)
                            alors j = (j+i)/2 - 1;
                    Sinon i = (j+i)/2 + 1;
                    Fin si
            Fin si
    Fin tant que
    retourner -1;
  Fin
```

- Au pire, la longueur de la partie du tableau comprise entre i et j est d'abord n, puis n/2, puis n/4,…jusqu'à ce que $n/2^t = 1$.
- Le nombre de tours de boucles est donc un entier t tel que $n/2^t = 1$, soit $2^t = n$, soit t*log(2) = log(n) ou t = log(n)

La complexité au pire est T(n) = c *log(n), donc la complexité asymptotique est de O(log n)

Exercice 5

Procédure tri_selection(entier[] T, entier n)
Début

pour i de 1 à n – 1 faire	(n-1)c1
min ← i	(n-1)c2
pour j de i + 1 à n faire	$c3 \sum_{j=2}^{n} j$ = c3 n(n-1)/2
si t[j] < t[min], alors min ← j	$c4 \sum_{j=2}^{n} j$ = c4 n(n-1)/2
fin pour	
si min ≠ i alors échanger t[i] et t[min] (n-1)c5	(n-1)c5
fin pour	

Fin

T(n) = (n-1)c1 + (n-1)c2 + c3 n(n-1)/2 + c4 n(n-1)/2 + (n-1)c5, donc la complexité asymptotique est de O(n^2)