Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique – Département d'Informatique
1er année Master « Sécurité des Systèmes Informatiques » SSI - 2014/2015.

---

07 Décembre 2014

Complexité et Algorithmique avancée
« Contrôle » durée 45 mn

<u>Exercice :</u> (10 pts)

Déterminer l'invariant des boucles pour chacun des algorithmes suivants et en déduire l'ordre de complexité. Justifier votre réponse

Algo_A1()

```
Début
i:=1;
Pour j := 1 à N
        Faire  i=i*j;
        Fait;
Pour j:=1 à i
        Faire <opération>;
        Fait ;
Fin.
```

Algo_A2()

```
Début
i:=1;
Tant que ( i <= N )
        Faire
        i=2*i;
        pour j:=1 à N
                Faire <opération>;
                Fait;
        Fait;
Fin.
```

Algo_A3()

```
Début
I :=1 ;
Pour j :=1 à N faire
        I :=i*j ; Fait ;
J :=1 ;
Tant que j<=i faire
J := 2*j ;
Fait ;
Fin.
```

Algo_A4 ()

```
Début
I :=2 ; j :=1 ;
Tant que i <= N
Faire I := i*i ;
fait;
Fin.
```

Algo_A5()

```
Début
I :=1 ; j :=1 ;
Tant que (i<=N)
faire
        si i<N alors i :=i+1 ;

        sinon j :=j+1 ;i :=j ;
        fsi ;
fait ;
Fin.
```

Algo_A6()

```
Début
I :=1 ; j :=1 ;
Tant que (i<=N)
Faire si i mod 2 =0 alors j :=1 ;
                        tant que j<=M
                        Faire j :=j+1 ; Fait ;
        Fsi ;
        I :=i+j ;

Fait ;
Fin.
```

Bonne Chance !

Solution :

- Algo A1 : deux boucles séquentielles

La première s'effectue en N itérations et calcule dans i le factoriel de N.

La seconde boucle s'effectue en i itérations.

Puisque i = n ! alors le nombre d'itération de la seconde boucle est égale à n !.

F1(n) = n + n ! = O(n !)

- Algo A2 : Deux boucles imbriquées

La première effectue une décomposition de n en puissance de 2

Et la seconde boucle s'effectue en N itérations.

Le nombre d'itérations au totale correspond au nombre de décomposition*N.

Le nombre de décomposition k est donné par l'expression suivante : $2^K <= N$, $k <= Log_2(n)$.

Donc, $F2(n) = n * Log_2(n) = O(nLog_2(n))$.

- Algo A3 :

La première boucle calcule n ! En n itération.

La seconde boucle effectue une décomposition en puissance de 2 de i qui est égale à n !.

Le nombre d'itération dans la seconde boucle, correspond à la puissance de 2 nécessaire pour atteindre n !. $2^k <= n!$, donc $k <= log_2(n!)$.

$F3(n) = n + log_2(n!) = n + log_2(1*2*3* \ldots *(n-1)*n) = n + log_2(\prod_{i=1}^{n} i) = n + \sum_{i=1}^{n} log2(i)$

$F3(n) <= n + n\, log_2(n) = O(nlog_2(n))$.

- Algo A4:

Ici le nombre d'itération correspond au nombre de fois que l'indice i est mis au carré pour atteindre la limite n. concrètement $(i^2)^k <= N$.

Donc $2^{2^k} \leq n \Rightarrow 2^k \leq \log_2 n \Rightarrow k \leq \log_2 \log_2 n$

Ainsi $F4(n) = O(\log_2 \log_2 n)$.

- Algo A5 :

Une seule boucle simulant deux boucles imbriquées à chaque fois que l'indice i atteint sa limite (n), il est réinitialisé avec la nouvelle valeur de j.

Ce processus peut être décrit par :

F5(n) = $\sum_{i=1}^{n} n - i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ = O(n²)

- Algo A6:

Une seule boucle linéaire car en fonction de la parité de i, l'indice est incrémenté de m.

Donc F6(n)= n = O(n).

Bonne Chance !