Série 1 : Algorithmique avancée, Outils mathématiques et ordre de complexité

Exercice 1

Ecrire six algorithmes différents pour déterminer si un nombre entier est premier ou composé. Evaluer la complexité pour chacun des algorithmes proposés.

Exercice 2

Considérer 7 algorithmes A0, A1, …, A6 conçus pour résoudre un même problème de taille n. La complexité en temps de chacun de ces algorithmes est exprimée dans la table ci-dessous.

Algorithme	Complexité temporelle	Temps		
		N = 10	N = 100	N = 1000
A0	$O(1)$			
A1	$O(\log_2 n)$			
A2	$O(\sqrt{n})$			
A3	$O(n)$			
A4	$O(n^2)$			
A5	$O(n^3)$			
A6	$O(2^n)$			

1) Calculer les ordres de grandeurs suivantes en secondes :
   a) 1 heure
   b) 1 jour
   c) 1 semaine
   d) 1 mois
   e) 1 année
   f) 1 siècle
   g) 1 millénaire

1) En supposant qu'une unité de taille s'exécute en une milliseconde, calculer le temps nécessaire pour traiter des tailles respectivement égales à 10, 100 et 1000.
2) Répéter la question 2 avec une unité de temps égale à une microseconde.
3) Que peut-on en conclure ?
4) En vous basant sur ce que vous pouvez trouver sur internet, donnez des exemples algorithmes ou de problèmes qui ont une complexité du même ordre que les algorithmes A_0, …, A_6.

Exercice 3

Supposons qu'on ait écrit une procédure pour additionner ***m*** matrices carrées de ***nxn***. Si l'addition de deux matrices carrées nécessite un temps d'exécution de $O(n^2)$ quelle sera la complexité de cette procédure en fonction de ***m*** et ***n*** ?

Exercice 4

Supposons que deux algorithmes résolvent le même problème l'un s'exécute en $T_1(n)=400n$ et l'autre en $T_2(n)=n^2$. Quelles sont les complexités de ces deux algorithmes ? Pour quelles valeurs de ***n*** doit-on préférer l'algorithme de complexité plus élevée ?

Exercice 5 Classer dans l'ordre croissant les complexités suivantes :

$O(n^2)$, $O(3^n)$, $O(2^n)$, $O(n^2 \log n)$, $O(1)$, $O(n \log n)$, $O(n^3)$, $O(n!)$, $O(\log n)$, $O(n)$.

Exercice 6 Quelles sont les complexités de :

- $T_1(n)= 3n\log n + \log n$
- $T_2(n)= 2^n + n^3+25$

- $T_3(n,k)=k+n$ où $k \leq n$

Classer les dans l'ordre croissant.

Exercice 7

Soit *f(n)* et *g(n)* deux fonctions positives asymptotique. En s'aidant de la définition de base de la notation Θ, prouver que :

$max(f(n),g(n))=\Theta(f(n)+g(n))$

Exercice 8

Montrer que pour deux constantes réelles *a* et *b* quelconques, avec $b>0$:

$(n+a)^b=\Theta(n^b)$

Exercice 9

Peut-on écrire : $2^{n+1}=O(2^n)$?

$2^{2n}=O(2^n)$?

Exercice 10 Montrer que les affirmations suivantes sont correctes :

(a) $5n^2-6n=\Theta(n^2)$

(b) $n !=O(n^n)$

(c) $2n^2+n \log n=\Theta(n^2)$

(d) $\sum_{i=0}^{n} i^2 =\Theta(n^3)$

(e) $\sum_{i=0}^{n} i^3 =\Theta(n^4)$

(f) $n^{2^n} +6*2^n=\Theta(n^{2^n})$

Exercice 11

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.
```