Algorithmes et structures de données : TD 5 Corrigé

Temps d'un algorithme T(n) - Notation Grand-O

Exercice 5.1 *Temps d'un algorithme T(n)*

Pour chacun des fonctions $T_i(n)$ suivant, déterminer sa complexité asymptotique dans la notation Grand-O. Exemple : $T_0(n) = 3n \in O(n)$.

1. $T_1(n) = 6n^3 + 10n^2 + 5n + 2 \in O(n^3)$
2. $T_2(n) = 3\log_2 n + 4 \in O(\log n)$
3. $T_3(n) = 2^n + 6n^2 + 7n \in O(2^n) \in O(a^n)$
4. $T_4(n) = 7k + 2 \in O(1)$
5. $T_4(n) = 4\log_2 n + n \in O(n)$
6. $T_5(n) = 2\log_{10} k + kn^2 \in O(n^2)$

Exercice 5.2 *Temps d'un algorithme T(n)*

Considérer les deux algorithmes A1 et A2 avec leurs temps d'exécution $T_1(n) = 9n^2$ et $T_2(n) = 100n + 96$ respectives.

1. Déterminer la complexité asymptotique des deux algorithmes dans la notation Grand-O. Quel algorithme a la meilleure complexité asymptotique?

- $T_1(n) = 9n^2 \in O(n^2)$
- $T_2(n) = 100n + 96 \in O(n)$

C'est l'algorithme A2 qui a la meilleure complexité asymptotique.

2. Montrer que vos solutions sont correctes en spécifiant un c et un n_0 par algorithme afin que la relation suivante soit satisfaite :
$O(f) = \{g | \exists c > 0 : \exists n_0 > 0 : \forall n \geq n_0 : g(n) \leq cf(n)\}$

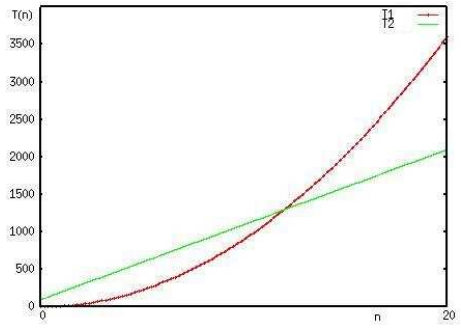
A1 p.ex. $c = 10$ et $n_0 = 1$ pour $f(n) = O(n^2)$ et $g(n) = T_1(n) = 9n^2$ car $\forall n \geq 1 : T_1(n) \leq 10n^2$

A2 p.ex. $c = 101$ et $n_0 = 100$ pour $f(n) = O(n)$ et $g(n) = T_2(n) = 100n + 96$ car $\forall n \geq 100 : T_2(n) \leq 101n$

Remarque : Bien sur, il y a d'autres choix possibles pour c et n_0

3. Calculer les temps maximales d'exécution des deux algorithmes $T_i(n)$ pour $n = 1, n = 3, n = 5, n = 10, n = 14$.

n	T_1	T_2
1	9	196
3	81	396
5	225	596
10	900	1096
14	1764	1496



4. Ebaucher les graphes des deux fonctions T_i dans un même système de coordonné (abscisse n, ordonné $T_i(n)$).
5. Pour quelles longueur de donnée n quel algorithme est le plus efficace ? (Calculer l'intersection ...)

Il faut poser :

$$T_1(n) = T_2(n) <==> 9n^2 - 100n - 96 = 0$$

Equation quadratique, la seule solution positive est n = 12. Donc :

- $0 \leq n < 12$: L'algorithme A1 est plus efficace
- $n \geq 12$: L'algorithme A2 est plus efficace

6. Quelle est la complexité asymptotique de l'algorithme suivant ? Quelle règle avez vous appliqué ?

```
début
appeler A1      {Ici l'algorithme 1 est exécuté. }
appeler A2      {Ici l'algorithme 2 est exécuté. }
fin
```

Il faut appliquer la règle des sommes :

$$T_1(n) + T_2(n) \in O(max(n^2, n)) = O(n^2)$$

La feuille continue au verso ..

Exercice 5.3 *Addition de matrices*

Considérer les deux matrices quadratiques A et B de taille n :

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & ... & b_{1n} \\ b_{21} & b_{22} & ... & b_{2n} \\ ... & ... & ... & ... \\ b_{n1} & b_{n2} & ... & b_{nn} \end{pmatrix},$$

L'addition de ces deux matrices donne la matrice C quadratique de taille n:

$$C = \begin{pmatrix} c_{11} & c_{12} & ... & c_{1n} \\ c_{21} & c_{22} & ... & c_{2n} \\ ... & ... & ... & ... \\ c_{n1} & c_{n2} & ... & c_{nn} \end{pmatrix}$$

avec

$$c_{ij} = a_{ij} + b_{ij} \quad \forall i, \forall j$$

1. Définir le type des matrices quadratiques et déclarer les variables A,B, et C.

```
type t_matrix = array[1..n, 1..n] of real;

var A,B,C     :  t_matrix;
```

2. Ecrire un algorithme qui effectue l'addition des deux matrices A et B et stocke les résultats en C.

```
pour i de 1 à n           {boucle exterieur}
        pour j de 1 à n   {boucle interieur}
                C[i,j] := A[i,j] + B[i,j];
        fin pour
fin pour
```

3. Déterminer la fonction de temps maximale ("worst case") T(n) pour des matrices de taille n.

- Dans la boucle intérieur : Une affectation j:=j+1, une comparaison $j <= n$, et une affectation C[i,j] := A[i,j] + B[i,j], le tout est répété n*n fois. ==> $3n^2$
- Dans la boucle extérieur : Une affectation i:=i+1, une comparaison $i <= n$, une affectation j := 1, le tout est répété n*n fois. ==> $3n$
- uniquement une fois : Une affectation i:=1. ==> 1

$T(n) = 3n^2 + 3n + 1$

4. Déterminer la complexité Grand-O pour des matrices de taille n.
$T(n) = 3n^2 + 3n + 1 \in O(n^2)$

Exercice 5.4 *Multiplication de matrices*

La multiplication des deux matrices quadratiques de taille n donne la matrice C quadratique de taille n:

$$C = \begin{pmatrix} c_{11} & c_{12} & ... & c_{1n} \\ c_{21} & c_{22} & ... & c_{2n} \\ ... & ... & ... & ... \\ c_{n1} & c_{n2} & ... & c_{nn} \end{pmatrix}$$

avec

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad \forall i, \forall j$$

1. Ecrire un algorithme qui effectue la multiplication des deux matrices A et B et stocke les résultats en C.

```
pour i de 1 à n           {boucle i}
        pour j de 1 à n   {boucle j}
                C[i,j] := 0;
                pour k de 1 à n   {boucle k}
                        C[i,j] := C[i,j] + A[i,k] * B[k,j];
                fin pour
        fin pour
fin pour
```

2. Déterminer la fonction de temps maximale ("worst case") T(n) pour des matrices de taille n.

   - Dans la boucle k : Une affectation k:=k+1, une comparaison $k <= n$, et une affectation C[i,j] := C[i,j] + A[i,k] * B[k,j], le tout est répété n*n*n fois. ==> $3n^3$
   - Dans la boucle j : Une affectation j:=j+1, une affectation C[i,j] := 0, une comparaison $j <= n$, une affectation k := 1, le tout est répété n*n fois. ==> $4n^2$
   - Dans la boucle i : Une affectation i:=i+1, une comparaison $i <= n$, une affectation j := 1, le tout est répété n fois. ==> $3n$
   - uniquement une fois : Une affectation i:=1. ==> 1

   $T(n) = 3n^3 + 4n^2 + 3n + 1$
3. Déterminer la complexité O(n) pour des matrices de taille n.
   $T(n) = 3n^3 + 4n^2 + 3n + 1 \in O(n^3)$