Algorithmes et structures de données : TD 5

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

Exercice 5.1 *Appels des fonctions par valeur*

Considérer le programme suivant :

```
<?php
function ajouter($a)
{
    echo "paramètre a : ".$a;
    echo "<br />";
    $a = $a + 2;
    echo "paramètre a : ".$a;
    echo "<br />";
}
$b = 4;
echo "b est (avant) : ".$b;
echo "<br />";
ajouter($b);
echo "b est (après) : ".$b;
?>
```

1. Faites tourner cet algorithme dans un tableau.

b	**a** (variable locale du 1^{er} appel de la fonction ajouter)
4	
	4
	6

2. Qu'est-ce qui est affiché à l'écran ?

```
b est (avant) : 4
paramètre a : 4
paramètre a : 6
b est (après) : 4
```

Exercice 5.2 *Appels des fonctions par référence*

Considérer le programme suivant :

```
<?php
function ajouter(&$a)
{
    echo "paramètre a : ".$a;
    echo "<br />";
    $a = $a + 2;
    echo "paramètre a : ".$a;
    echo "<br />";
}
$b = 4;
echo "b est (avant) : ".$b;
echo "<br />";
ajouter($b);
echo "b est (après) : ".$b;
?>
```

1. Faites tourner cet algorithme dans un tableau.

b (et en même temps **a** du 1^{er} appel de la fonction ajouter qui est une référence à **b**)
4
6

2. Qu'est-ce qui est affiché à l'écran ?

```
b est (avant) : 4
paramètre a : 4
paramètre a : 6
b est (après) : 6
```

Exercice 5.3 *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 5.4 *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. $c = 9$ et $n_0 = 1$ pour $f(n) = O(n^2)$ et $g(n) = T_1(n) = 9n^2$ car $\forall n \geq 1 : T_1(n) \leq 9n^2$

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

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 $n = 1, n = 3, n = 5, n = 10, n = 14$.

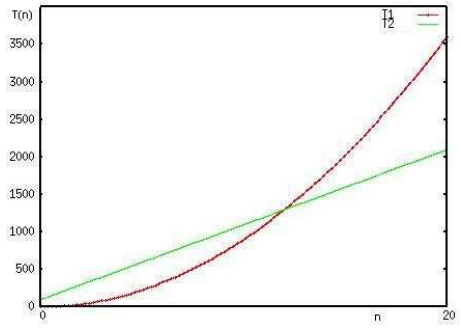
n	T_1	T_2
1	9	196
3	81	396
5	225	596
10	900	1096
14	1764	1496

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 :

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

Exercice 5.5 *Complexité asymptotique*

1. Considérer les algorithmes suivantes avec un temps d'exécution $T(n)$ pour une longueur de données n. Déterminer leur complexités asymptotiques respectives, et indiquer quel(s) règle(s) vous aviez appliqués.

Algorithme A1 $T(n) = 3n + 2$

Complexité asymptotique $O(n)$

Algorithme A2 $T(n) = 6$

Complexité asymptotique $O(1)$

Algorithme A3 $T(n) = 4n^2 + n + 2$

Complexité asymptotique $O(n^2)$

Algorithme A4

```
Exécuter A1;
Exécuter A2;
Exécuter A3;
```

Règle de somme : Complexité asymptotique $O(n^2)$

Algorithme A5

```
pour i de 1 à n faire
    Exécuter A3;
fin pour
Exécuter A1;
```

Règle de produit pour la boucle et ensuite règle de somme : Complexité asymptotique $O(n^3)$

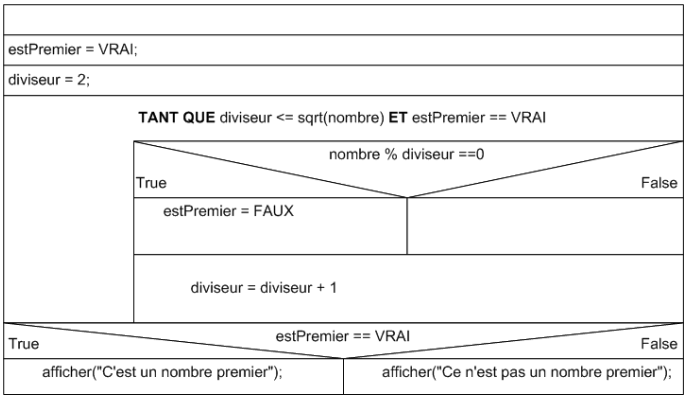
Algorithme A6

```
pour i de 1 à 5 faire
    Exécuter A1;
fin pour
```

5 fois la règle de somme : Complexité asymptotique $O(n)$

Exercice 5.6 *Fonctions*

Considérer le structogramme d'un algorithme qui affiche à l'écran si un nombre est un nombre premier :



1. Ecrire une fonction en PHP qui s'appelle `premier` et qui décide si un nombre est premier, donc qui retourne `true` si un nombre est premier, et `false` sinon. Quelle est la complexité de cette fonction ?

```
function premier($nombre)
{
    $estPremier = true;
    $diviseur = 2;
    while ($diviseur <= sqrt($nombre) && $estPremier == true)
    {
        if ($nombre % $diviseur == 0)
            $estPremier = false;
        $diviseur = $diviseur + 1;
    }
    return $estPremier;
}
```

La complexité de cet algorithme est de O(sqrt(n))

2. Ecrire un algorithme qui affiche à l'écran tous les nombre premiers compris entre `2` et `n`, en utilisant la fonction `premier`. Quelle est la complexité de cet algorithme ?

```
$i = 2;
while ($i <= $n)
{
    if (premier($i) == true)
        echo $i."  ";
    $i = $i + 1;
}
```

La complexité de cet algorithme est de $O(\sqrt{n} + \sqrt{n-1} + \sqrt{n-2} + ... + \sqrt{1}) = O(\frac{2}{3}n * \sqrt{n} + \sqrt{n}) = O(n * \sqrt{n})$