TD2 : Complexité

Sauf indication contraire, nous supposerons dans ce TD que les opérations fondamentales suivantes nécessitent chacune un cycle machine pour être exécutée: affectation, branchement conditionnel, opération arithmétique, opération logique et appel de fonction.

Exercice 1

On donne la fonction suivante qui permet de mettre en majuscule la première lettre d'une chaîne de caractères :

```
void Majuscule(char *Text)
{
    if (Text[0]>='a' && Text[0]<='z')
        Text[0]-=32;
}
```

On s'intéresse dans cet exercice à la complexité en terme d'affectations.

a. Si la chaîne passée en paramètre est "Bonjour", compter le nombre d'opérations fondamentales.

b. Si la chaîne passée en paramètre est "bonjour", compter le nombre d'opérations fondamentales.

c. Si la chaîne passée en paramètre est "bonjour et bienvenue à Angers", compter le nombre d'opérations fondamentales.

d. Calculer la complexité de la fonction dans le pire cas $T_{max}(n)$.

e. Si l'on suppose qu'il y a une chance sur deux pour que le premier caractère de la chaîne soit une majuscule, calculer la complexité moyenne $T_{moy}(n)$.

f. En déduire l'ordre de grandeur de la complexité de la fonction .

Exercice 2

On donne la fonction suivante qui permet de retourner l'indice de l'élément X dans une liste ordonnée Tab de Nb éléments:

```
int IsItem (int *Tab, int Nb, int X)
{
    for (int i=0;i<Nb;i++)
    {
        if (Tab[i]==X)
            return i;
    }
    return -1;
}
```

On s'intéresse dans cet exercice à la complexité en terme de branchements conditionnels.

a. On suppose que Tab={1,2,3,4,5}, que renvoie l'appel de la fonction suivant:

```
IsItem (Tab,5,3)
```

b. On suppose que Tab={1,2,3,4,5}, que renvoie l'appel de la fonction suivant:

```
IsItem (Tab,5,7)
```

c. Dans les exemples des questions a. et b. calculer le nombre d'opérations fondamentales.

d. Quel est la complexité dans le pire des cas ?

e. Si le tableau contient 1024 données, compter le nombre de branchements conditionnels.

f. En déduire l'ordre de grandeur de la complexité de la fonction

Exercice 3

On donne la fonction suivante qui permet de retourner l'indice de l'élément X dans une liste ordonnée Tab de Nb éléments:

```
int IsItemDicho (int *Tab, int Debut, int Fin, int X)
{
    int Milieu;

    if(Debut >= Fin)            // l'intervalle est vide
        return -1;              // X n'est pas dans le tableau

    Milieu = (Debut+Fin)/2;     // Calcul du centre de l'interval

    if(Tab[Milieu] == X)        // On a trouvé l'élément
        return Milieu;

    if(Tab[Milieu] > X)         // L'élément est dans la partie supérieure
        return IsItemDicho(Tab, Debut, Milieu, X);
    else                        // L'élément est dans la partie inférieure
        return IsItemDicho(Tab, Milieu+1, Fin, X);
}
```

On s'intéresse dans cet exercice à la complexité en terme de branchements conditionnels.

a. On suppose que Tab={1,2,3,4,5,6,7,8,9,10}, calculer le nombre d'opérations fondamentales lors de l'appel suivant:

```
IsItemDicho (Tab,0,10,8)
```

b. Même question avec l'appel suivant:

```
IsItemDicho (Tab,0,10,12)
```

c. Calculer la complexité dans le pire des cas en fonction de x, en posant $n=2^x$.

e. Si le tableau contient 1024 données, compter le nombre de branchements conditionnels.

f. En déduire l'ordre de grandeur de la complexité de la fonction

Exercice 4

a. Compter le nombre de multiplications nécessaires pour multiplier deux matrices carrées 2x2.

b. Compter le nombre de multiplications nécessaires pour multiplier deux matrices carrées 3x3.

c. Compter le nombre de multiplications nécessaires pour multiplier deux matrices carrées nxn.

d. En déduire l'ordre de grandeur du produit de deux matrices.

e. Refaire l'analyse en prenant comme opération fondamentale l'addition.

Exercice 5

a. Dans le tableau suivant, calculer le nombre d'opérations nécessaires pour exécuter l'algorithme en fonction de sa complexité et de la taille des données à traiter.

Complexité	**n=10**	**n=100**	**n=1000**
$O(1)$			
$O(\log_2(n))$			
$O(n)$			
$O(n.log_2(n))$			
$O(n^2)$			
$O(n^3)$			
$O(2^n)$			

b. Dans le tableau suivant, calculer le temps nécessaire en supposant que l'on utilise un ordinateur capable de réaliser 10^9 opérations par seconde.

Complexité	**n=10**	**n=100**	**n=1000**
$O(1)$			
$O(\log_2(n))$			
$O(n)$			
$O(n.log_2(n))$			
$O(n^2)$			
$O(n^3)$			
$O(2^n)$			

c. Calculer la taille du plus grand problème que l'on peut traiter dans le temps imparti.

On rappelle la relation suivante : $\log_n(x^p) = p.log_n(x)$

Complexité	**1 seconde**	**1 heure**	**1 an**
$O(1)$			
$O(\log_2(n))$			
$O(n)$			
$O(n^2)$			
$O(n^3)$			
$O(2^n)$			

Exercice 6

Donnez la complexité des programmes suivants (Ordre de grandeur O et complexité exacte Θ). On précise la somme des n premiers entiers peut s'écrire: $1+2+3+4+5+...+n=\frac{n(n+1)}{2}$

a.
```
for (j=0;j<n;j++)
    x+=3;
```

b.
```
for (i=0;i<n;i++)
    for (j=1;j<n;j++)
        x+=3;
```

c.
```
for (i=0;i<n;i++)
    for (j=0;j<i;j++)
        x+=3;
```

d.
```
for (i=5;i<n-5;i++)
    for (j=i-5;j<i+5;j++)
        x+=3;
```

e.
```
for (i=0;i<n;i++)
    for (j=1;j<n;j++)
        for (k=1;k<n;k++)
            x+=3;
```

f.
```
for (i=n;i>1;i=i/2)
    x+=3;
```

g.
```
for (i=n;i>1;i=i/2)
    for (j=1;j<n;j++)
        x+=3;
```

Exercice 7

a. Etudier le nombre de multiplications réalisées avec le programme suivant dans le meilleur cas, dans le pire cas et dans le cas moyen en supposant que les tests ont une probablité de 0,5 d'être vrai.

```
for (i=0;i<n;i++)
    if (T[i]>a)
        x*=3;
```

b. Même question avec le programme suivant:

```
if (a>b)
    for (i=0;i<n;i++)
        x*=3;
else
    x*=2;
```