

Travaux Pratiques Feuille de TD N° 1

Objectifs d'apprentissage :

- Concevoir des algorithmes récursifs
- Analyser de la complexité des algorithmes

Exercice 1

1. Écrire une fonction récursive qui calcule la somme $1 + 2 + \dots + n$, où n est un entier positif.
2. Écrire une fonction récursive qui calcule la somme des chiffres d'un entier positif n .
3. Écrire une fonction récursive qui affiche à l'envers un entier positif n .

Exercice 2

Dans les algorithmes suivants, k désigne une constante entière et l'instruction « *do one basic operation* » désigne n'importe quelle opération qui s'effectue en temps constant ($O(1)$). Évaluer la complexité de chacun de ces algorithmes :

1. **for** ($i = 1; i \leq n; i++$)
do one basic operation;
2. **for** ($i = 1; i \leq n; i += k$)
do one basic operation;
3. **for** ($i = n; i \geq k; i--$)
do one basic operation;
4. $f(\text{int } n) \{$
 $\text{if}(n > 1) \{$
 do one basic operation;
 $f(n / 2);$
 $f(n / 2); \}$
 $\}$
5. $f(\text{int } n) \{$
 $\text{if}(n == 1)$
 do one basic operation;
 else $\{$
 do one basic operation;
 $f(n - 1); \}$
 $\}$
6. **for** ($i = 1; i \leq n; i = 2 * i$)
do one basic operation;

```

7. for (i = n; i >= 1; i = i / 3)
    do one basic operation;
8. f(int n) {
    if(n == 1)
        do one basic operation;
    else {
        do one basic operation;
        f(n / 2); }
    }
9. for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j = j * 2)
        do one basic operation;
10. f(int n) {
    if(n > 1) {
        f(n / 2);
        f(n / 2);
        for (i = 1; i <= n; i++)
            do one basic operation; }
    }
11. for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
        do one basic operation;
12. for (i = 1; i <= n; i++)
    for (j = 1; j <= i; j++)
        do one basic operation;
13. f(int n) {
    if (n > 1) {
        f(n / 2);
        f(n / 2);
        f(n / 2);
        f(n / 2);
        for (i = 1; i <= k; i++)
            do one basic operation }
    }
14. f(int n) {
    limite = 1;
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= limite; j++)
            do one basic operation
        limite = limite * 2; }
    }
15. f(int n) {
    if (n == 0)
        do one basic operation

```

```

else {
    f(n - 1);
    f(n - 1); }
}

```

Exercice 3

Soient a un nombre réel strictement positif et n un entier.

1. Écrire une fonction récursive qui calcule a^n en utilisant la formule $a^n = a \times a^{n-1}$.
2. Évaluer la complexité de cette fonction.
3. Écrire une fonction récursive qui calcule a^n en utilisant la formule $a^n = (a^{n/2})^2$, si n est pair et $a^n = a \times (a^{(n-1)/2})^2$.
4. Évaluer la complexité de cette fonction.

Exercice 4

Considérons la fonction suivante :

```

int f(int n)
{
    if (n == 0)
        return 1;
    else
        return f(n - 1) + f(n - 1) + f(n - 1);
}

```

1. Que calcule la fonction f ?
2. Évaluer la complexité de cette fonction.
3. Écrire une fonction qui calcule $f(n)$ et qui soit en $O(n)$.
4. Écrire une fonction qui calcule $f(n)$ et qui soit en $O(\log(n))$.