TD NUMÉRO 2

RÉCURSIVITÉ

Ramzi Guetari

EXERCICE 1 : Longueur d'une chaîne de caractères

Soit une chaîne de caractères terminée par le caractère '\0'.

1. Ecrire un algorithme récursif permettant de déterminer sa longueur.
2. Estimer sa complexité.

EXERCICE 2 : Maximum dans un tableau

Soit un tableau X de N entiers.

1. Ecrire une fonction récursive simple permettant de déterminer le maximum du tableau.
2. Estimer sa complexité.

EXERCICE 3 : Vérification de l'ordre d'un tableau

Un tableau X est trié par ordre croissant si $X[i] \leq X[i+1], \ \forall i$

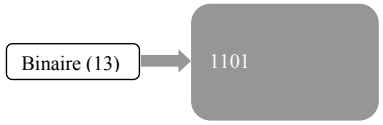
1. Elaborer un algorithme récursif permettant de vérifier qu'un tableau X est trié ou non
2. Estimer sa complexité

EXERCICE 4 : Rendre récursive la fonction somme suivante :

```
int somme (int* x, int n) {
    int i, s ;
    for (i = 0; i < n ; i++)
        s += x[i];
    return s;
}
```

EXERCICE 5 : Conversion binaire

Pour convertir un nombre entier positif N de la base décimale à la base binaire, il faut opérer par des divisions successives du nombre N par 2.
Les restes des divisions constituent la représentation binaire.



1. Ecrire une fonction récursive « Binaire » permettant d'imprimer à l'écran la représentation binaire d'un nombre N (voir exemple en face).
2. Donner la formule récurrente exprimant sa complexité en nombre de divisions. Estimer cette complexité.

EXERCICE 6 : Palindrome

Un mot est un palindrome si on peut le lire dans les deux sens de gauche à droite et de droite à gauche. Exemple : KAYAK est un palindrome.

1. Ecrire une fonction récursive permettant de vérifier si un mot est ou non un palindrome. Elle doit renvoyer 1 ou 0.
2. Estimer sa complexité en nombre d'appels récursifs en fonction de la longueur du mot.

EXERCICE 7 : Multiplication récursive

La multiplication de deux entiers peut être réalisée à l'aide d'adition seulement.

1. Elaborer un algorithme récursif permettant de multiplier deux entiers
2. Estimer sa complexité

EXERCICE 8 : Division récursive

La division de deux entiers positifs peut être réalisée à l'aide de soustraction seulement.

1. Elaborer un algorithme récursif permettant de diviser un entier a par un entier b
2. Estimer sa complexité

EXERCICE 9 : Suite de Fibonacci

La suite de Fibonacci est définie comme suit :

$$u_n = \begin{cases} 1 & \Leftrightarrow 0 \le n < 2 \\ u_{n-1} + u_{n-2} & \text{sinon} \end{cases}$$

1. Ecrire un algorithme récursif calculant Fibonacci (n)
2. Déterminer sa complexité
3. Ecrire un algorithme récursif qui calcule, pour tout n > 0, le couple (Fibonacci (n), Fibonacci (n -1)).
4. Utiliser l'algorithme précédent pour écrire un nouvel algorithme calculant Fibonacci (n).
5. Qu'elle est la complexité (en nombre d'additions) de cet algorithme ?
6. En utilisant un tableau pour y stocker les termes calculés, proposer un meilleur algorithme. Estimer sa complexité au meilleur, en moyenne et au pire.

EXERCICE 10 :

Soit la suite définie par :

$$u_n = \begin{cases} 1 & \Leftrightarrow 0 \le n < 2 \\ 2 \times u_{n-1} + u_{n-2} & \text{sinon} \end{cases}$$

1. Ecrire un algorithme récursif permettant de calculer le $n^{ème}$ terme de la suite.
2. Estimer sa complexité.

EXERCICE 11 : Etiquetage de composantes connexes (1D)

Soit un tableau d'entiers contenant des valeurs 0 ou bien 1. On appelle composante connexe une suite contigüe de nombres égaux à 1. On voudrait changer la valeur de chaque composante connexe de telle sorte que la première composante ait la valeur 2 la deuxième ait la valeur 3, la $3^{ème}$ ait la valeur 4 et ainsi de suite. Réaliser deux fonctions :

1. la première fonction n'est pas récursive et a pour rôle de chercher la position d'un 1 dans un tableau
2. la deuxième fonction est récursive. Elle reçoit la position d'un 1 dans une séquence et propage une valeur x à toutes les valeurs 1 de la composante connexe.

EXERCICE 12 : Etiquetage de composantes connexes

Soit une image binaire représentée dans une matrice à 2 dimensions. Les éléments m[i][j] sont dits pixels et sont égaux soit à 0 soit à 1. Chaque groupement de pixels égaux à 1 et connectés entre eux forment une composante connexe (Figure). L'objectif est de donner une valeur différente de 1 à chaque composante connexe (2 puis 3 puis 4 etc.).

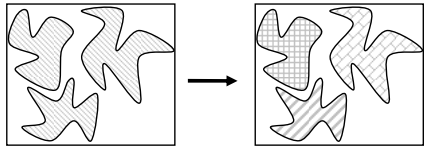


Image binaire Image étiquetée

1. Ecrire un algorithme récursif « propager » permettant de partir d'un point (i, j) situé à l'intérieur d'une composante connexe et de propager une étiquette T à tous les pixels situés à l'intérieur de la composante.
2. Estimer sa complexité.
3. Ecrire un algorithme « etiqueter » permettant d'affecter une étiquette différente à chaque composante connexe.

EXERCICE 13 : Triangle de Pascal

Le triangle de Pascal est défini comme suit : les éléments de la première colonne et ceux de la diagonale sont égaux à 1. Les autres éléments sont obtenus en additionnant l'élément du dessus à son voisin gauche. Exemple en face. Ecrire une fonction récursive permettant de générer un triangle de Pascal de taille N. Estimer sa complexité en nombre d'appels récursifs.

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
```

EXERCICE 14 :

Calculer la complexité suivante :

```
int x ;

int foo (int i, int j, int k) {
    if (k + j == i)
       return ((i - j) div k) + 1;

    x = foo (i, j + 1, k - 2);

    foo (i + 1, j + x, k - 2)
}
```