Série d'exercices-2 : Récursivité

Exercice-1 : Longueur d'une chaîne de caractères

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

a- Ecrire un algorithme récursif permettant de déterminer sa longueur
b- Estimer sa complexité

Exercice-2 :

Soit un tableau X de N entiers.

a- Ecrire une fonction récursive simple permettant de déterminer le maximum du tableau
b- Estimer sa complexité

Exercice-3 : pgcd de deux nombres

Le PGCD de deux nombres a et b est égal au PGCD de b et du reste de la division de a par b.

a- Ecrire un algorithme récursif permettant de déterminer le pgcd de deux nombres.
b- Estimer sa complexité.
c- Dérécursiver l'algorithme

Exercice-4 : Vérification de l'ordre d'un tableau

Un tableau X est trié par ordre croissant si x(i) <= x(i+1) pour tout i

a- Elaborer un algorithme récursif permettant de vérifier qu'un tableau X est trié ou non
b- Estimer sa complexité

Exercice-5 :

Rendre récursive la fonction somme (en encadré-1).

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

Exercice-6 :

Considérer le programme (en encadré-2)

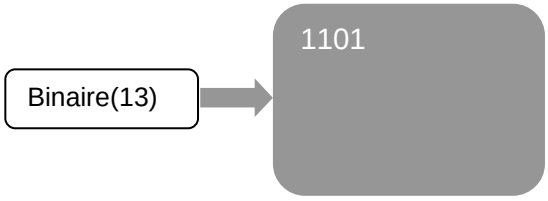
a- On exécute le programme et on tape dans l'ordre les nombres suivants : 5, 7, 13, 4, 7, 0, 5, 15, 50, 0. Quels sont les nombres affichés à l'écran.
b- Dérécursiver la fonction « traiter ».

```
void traiter()
{       int x;
        cin >> x;
        if (x != 0)
                traiter();
        cout << " , " << x;
}
void main ()
{       traiter (); }
```

Exercice-7 : 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.

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



cette complexité.

Exercice-8 : Palindrome

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

a- Ecrire une fonction récursive permettant de vérifier si un mot est ou non un palindrome. Elle doit renvoyer 1 ou 0.
b- Déterminer l'équation récurrente en fonction des appels récursifs

Exercice-9 : Multiplication récursive

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

a- Elaborer un algorithme récursif permettant de multiplier deux entiers
b- Estimer sa complexité

Exercice-10 : Division récursive

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

c- Elaborer un algorithme récursif permettant de diviser un entier a par un entier b
d- Estimer sa complexité

Exercice-11 : Suite de Fibonacci

La suite de Fibonacci est définie comme suit :

$$u_n = \begin{cases} 1 & si \quad n < 2 \\ u_{n-1} + u_{n-2} \sin on & \end{cases}$$

a- Ecrire un algorithme récursif calculant Fib(n)
b- Déterminer sa complexité
c- Ecrire un algorithme récursif qui calcule, pour n > 0, le couple (Fibonacci(n), Fibonacci(n -1)).
d- Utiliser l'algorithme précédent pour écrire un nouvel algorithme calculant Fibonacci(n).
e- Qu'elle est la complexité (en nombre d'additions) de cet algorithme ?
f- 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-12 :

Soit la suite définie par :

$$u_n = \begin{cases} 1 & si \quad n < 2 \\ 2 * u_{n-1} + u_{n-2} & \sin on \end{cases}$$

a- Ecrire un algorithme récursif permettant de calculer le n^{ème} terme de la suite.
b- Estimer sa complexité.

Exercice-13 :

Soit la suite définie par : $\begin{cases} u_n = 3u_{n-1} + 4u_{n-2} & si \quad n \geq 2 \\ u_0 = 0 \quad et \quad u_1 = 1 & \end{cases}$

a- Ecrire une fonction récursive « suite1 » permettant de déterminer le Nème terme de la suite. Estimer sa complexité.
b- Ecrire une fonction récursive « suite2 » permettant de déterminer et de retourner deux termes de la suite : le terme de rang N et le terme de rang (N-1). Pour cela,

vous pouvez utiliser l'instruction « return (a,b) » pour retourner les deux valeurs a et b.

c- En utilisant la fonction « Suite2 », écrire une fonction « Suite3 » permettant de calculer le Nème terme de la suite. Estimer la complexité de « Suite3 ».

Exercice-14 : Récursivité mutuelle

Un nombre N est pair si (N-1) est impair, et un nombre N est impair si (N-1) est pair.

a- Ecrire deux fonctions récursives mutuelles pair (N) et impair (N) permettant de savoir si un nombre N est pair et si un nombre N est impair.

b- Estimer la complexité de la fonction Pair (N) en nombre d'appels récursifs.

Exercice-15 : Récursivité mutuelle

Soient u et v les deux suites définies par : $\begin{cases} u_0 = a \\ v_0 = b \end{cases} \quad et \quad \forall n > 0 \begin{cases} u_n = u_{n-1}^2 + 2v_{n-1} \\ v_n = v_{n-1} + 3u_{n-1} \end{cases}$

a- Ecrire deux fonctions CalculerU (a,b,n) et CalculerV(a,b,n) pour calculer respectivement les deux termes Un et Vn des deux suites.

b- Calculer la complexité de U(n) en nombre d'appels récursifs

Exercice-16 : Récursivité mutuelle

Soient deux séries définies de la manière suivante:

$$\begin{cases} D(0) = 1 \\ D(i) = \sum_{k=1}^{i} \left\lfloor \frac{R(i-k)}{k} \right\rfloor \end{cases} \quad avec \quad i > 0 \qquad \begin{cases} R(0) = 1 \\ R(i) = \sum_{k=0}^{i-1} \left\lceil \frac{D(k)}{i-k} \right\rceil \end{cases} \quad avec \quad i > 0$$

le symbole $\lfloor\ \rfloor$ désigne la valeur entière au plancher et le symbole $\lceil\ \rceil$ désigne la valeur entière au plafond.

a- Comment peut-on qualifier la définition de ces deux séries ?

b- Ecrire un algorithme permettant de calculer le ième terme de chaque série.

Exercice-17 : Etiquetage de composantes connexes (1D)

Soit un tableau d'entiers contenant des valeurs 0 ou bien 1. On appelle composante connexe une suite contigue 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 3ème ait la valeur 4 et ainsi de suite. Réaliser deux fonctions :

- la première fonction n'est pas récursive et a pour rôle de chercher la position d'un 1 dans un tableau
- 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-18 : 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.).

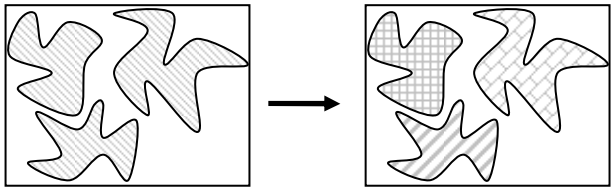


Image binaire Image étiquetée

a- 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.
b- Estimer sa complexité.
c- Ecrire un algorithme « etiqueter » permettant d'affecter une étiquette différente à chaque composante connexe.

Exercice-19 : Puissance X^n

a- Ecrire une fonction itérative permettant de calculer X^n et estimer sa complexité en nombre de multiplications
b- Ecrire une fonction récursive simple permettant de calculer X^n et estimer sa complexité en nombre de multiplications

Exercice-20 :

Considérer l'algorithme récursif de résolution du problème des tours de Hanoi (vu en cours). Cet algorithme est récursif multiple puisqu'il existe deux appels récursifs. On voudrait le s'en inspirer pour élaborer un algorithme non récursif pour la résolution du problème des tours de Hanoi. Pour cela, on vous demande de suivre la démarche suivante :

```
Hanoi (n, départ, intermédiaire, destination)
    If n > 0 Then
        Hanoi (n-1, départ, destination, intermédiaire)
        déplacer un disque de départ vers destination
        Hanoi (n-1, intermédiaire, départ, destination)
    Endif
End Hanoi
```

a- Exécuter le programme récursif manuellement et dresser un arbre ternaire lorsqu'on l'appelle avec Hanoi (3,1,2,3).
b- Lors de l'exécution, de quelle manière est parcouru l'arbre ?
c- Si on veut réaliser un algorithme non récursif, que doit-on faire lors de la descente de l'arbre ? et que doit-on faire lors de la remontée de l'arbre ?
d- Proposer un algorithme non récursif pour résoudre le problème des tours de Hanoi..

Exercice-21 :

Soit la suite T(n) définie par :

$$U(n) = \begin{cases} 1 \quad si \quad n = 0/1/2 \\ U(n-1) + 2*U(n-2) + U(n-3) \end{cases}$$

a- Ecrire une fonction récursive permettant de calculer la valeur *U(n)* de la suite.
b- Donner une formule récurrente exprimant sa complexité puis estimer cette complexité.
c- Proposer une fonction itérative pour calculer U(n) avec une complexité O(n)

Exercice-22 : Evaluation d'une expression en notation préfixe

Une expression arithmétique peut être présentée en notation préfixe. Ainsi l'expression suivante : « *+7**4 6 + 835 9 5 » est équivalente à : « (7 + 4*6 * (835+9))*5 ».

Ecrire une fonction récursive permettant de recevoir une chaîne de caractères représentant une expression en notation préfixe et de calculer sa valeur. On suppose que les opérateurs sont tous des opérateurs binaires (+ , - *, /) , que les valeurs sont tous entiers positifs, qu'ils peuvent être composés de plusieurs chiffres et qu'ils sont séparées par un ou plusieurs blancs.

Exercice-23 : Evaluation d'une expression postfixe

Une expression arithmétique peut être présentée en notation postfixe. Ainsi l'expression suivante : « 5 8 + 6 2 - * » est équivalente à : « (5+8)*(6-2) ».

Ecrire une fonction récursive permettant de recevoir une chaîne de caractères représentant une expression en notation postfixe et de calculer sa valeur. On suppose que les opérateurs sont tous des opérateurs binaires (+ , - *, /) , que les valeurs sont tous entiers positifs, qu'ils peuvent être composés de plusieurs chiffres et qu'ils sont séparées par un ou plusieurs blancs.

Exercice-24 : Conversion d'une expression préfixe en infixe

Une expression arithmétique peut être présentée en notation préfixée. Ainsi l'expression suivante : : « * + 5 8 – 6 2 » est équivalente à : « (5+8)*(6-2) ».

Ecrire une fonction récursive permettant de recevoir une chaîne de caractères représentant une expression en notation préfixe et de la réécrire en notation infixe. On suppose que les opérateurs sont tous des opérateurs binaires (+ , - *, /) , que les valeurs sont tous entiers positifs, qu'ils peuvent être composés de plusieurs chiffres et qu'ils sont séparées par un ou plusieurs blancs.

Exercice-25 : Conversion d'une expression postfixe en infixe

Une expression arithmétique peut être présentée en notation postfixée. Ainsi l'expression suivante : : « 5 8 + 6 2 - * » est équivalente à : « (5+8)*(6-2) ».

Ecrire une fonction récursive permettant de recevoir une chaîne de caractères représentant une expression en notation postfixe et de la réécrire en notation infixe. On suppose que les opérateurs sont tous des opérateurs binaires (+ , - *, /) , que les valeurs sont tous entiers positifs, qu'ils peuvent être composés de plusieurs chiffres et qu'ils sont séparées par un ou plusieurs blancs.

Exercice-26 : 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