TD 12 - Récursivité

Professeur ESSADDOUKI Mostafa (essaddouki@gmail.com)

http://www.developpement-informatique.com

28 Septembre 2017

Exercice 1

Soit une chaine de caractères :

1. Ecrire un algorithme récursif permettant de déterminer sa longueur

```
def longueur(ch):
        if not ch:
                return 0
        else:
                return 1+longueur(ch[1:])

ch="Take_It_Easy"
print(longueur(ch))
```

Exercice 2

Rendre récursive la fonction somme suivante :

```
def somme(L):
   s=0
   for val in L:
      s+=val
   return s
```

```
def somme(L):
        if not L:
                return 0
        return L[0]+somme(L[1:])
```

Exercice 3

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.

```python
def binaire(N):
        if N==0:
                return []
        return binaire(N//2)+[N%2]

print(binaire(13))
```

Exercice 4

La suite de Fibonacci est définie comme suit :

$$Fn = \begin{cases} 1 \text{ si } n <= 2 \\ F_{n-1} + F_{n-2} \text{ sinon} \end{cases}$$

1. Ecrire un programme récursif calculant $Fib(n)$

```python
def Fib(n):
        if n<=2:
                return 1
        return Fib(n-1)+Fib(n-2)

print(Fib(4))
```

Exercice 5

soit La suite définie par :

$$Un = \begin{cases} 1 \text{ si } n < 2 \\ 3U_{n-1} + F_{n-2} \text{ sinon} \end{cases}$$

1. Ecrire un programme récursif permettant de calculer le nième terme de la suite.

```python
def Suite(n):
        if n<=2:
                return 1
        return 3*Suite(n-1)+Suite(n-2)

print(Suite(4))
```

Exercice 6

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

1. 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.

```python
def Pair(N):
        if N==1:
                return False
        return Impair(N-1)

def Impair(N):
        if N==1:
                return True
        return Pair(N-1)
```

Exercice 7

Soit un tableau X de N entiers.

1. Ecrire une fonction récursive simple permettant de déterminer le maximum du tableau

```
def maximum(T):
        if len(T)==1:
                return T[0]
        m=len(T)//2
        max1=maximum(T[:m])
        max2=maximum(T[m:])
        if max1>max2:
                return max1
        return max2
```

Exercice 8

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

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

```
def Esttrier(T):
        if len(T)==0 or len(T)==1:
                return True
        if T[0]<=T[1]:
                return Esttrier(T[1:])
        return False

T=[1,2,3,4,4,5,7,8]
print(Esttrier(T))
```

Exercice 9

Un mot est un palindrome si on peut le lire dans les deux sans 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 palindrome.

```
def palindrome(ch):
        if len(ch)==1:
                return True
        if ch[0]==ch[-1]:
                return palindrome(ch[1:len(ch)-1])
        return False

ch="KAYAK"
print(palindrome(ch))
```

Exercice 10

Soit un tableau d'entiers contenant des valeurs 0 ou bien 1. On appel 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 ai la valeur 2 la deuxième ai la valeur 3, la 3ème 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 valeur 1 de la composante connexe.

```
1  def trouver(T):
2          for i in range(len(T)):
3                  if T[i]==1:     return i
4          return None
5
6  def propage(T,x,val):
7          i=trouver(T)
8          if i is None:
9                  return
10         T[i]=val
11         if (len(T)-i+1)>2:
12                 if T[i+1]==1:
13                         propage(T,x,val+1)
14                 else:
15                         propage(T,x,x)
16
17 T=[0,0,1,1,0,1,1,1,1,1,0,1,1]
18 x=2
19 propage(T,x,x)
20 print(T)
```

Exercice 11

Soit une image binaire représentés dans une matrice à 2 dimension. 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'objectifs est de donner une valeur différente de 1 à chaque composante (2 puis 3 puis 4 etc.)

1. Ecrire une fonction récursive 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. Ecrire une fonction etiqueter permettant d'affecter une étiquette différente à chaque composante connexe.

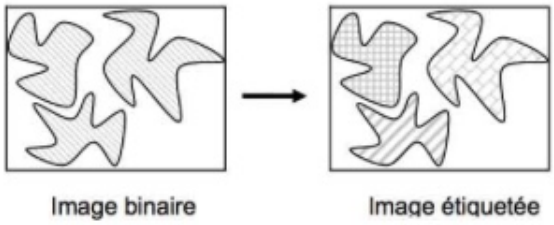

FIGURE 1 – composante connexe

```python
def propager(M,i,j,val):
        if M[i][j]==0:
                return
        M[i][j]=val
        if((i-1)>=0 and M[i-1][j]==1): #l'element en haut
                propager(M,i-1,j,val)

        if((i+1)<len(M) and M[i+1][j]==1):#l'element en bas
                propager(M,i+1,j,val)

        if((j-1)<len(M) and M[i][j-1]==1): # a gauche
                propager(M,i,j-1,val)

        if((j+1)<len(M) and M[i][j+1]==1):# a droite
                propager(M,i,j+1,val)

def etiqueter(M):
        L,C=len(M),len(M[0])
        etat=True
        val=2
        while etat==True:
                etat=False
                for i in range(L):
                        for j in range(C):
                                if(M[i][j]==1):
                                        propager(M,i,j,val)
                                        val+=1

M=[[0,0,1,0],[0,1,0,1],[1,1,1,0],[0,1,1,0]]
print(M)
etiqueter(M)
print(M)
```