Exercices : Exercices de complexité et récursivité

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

1. Elaborer un programme itératif permettant de vérifier qu'un tableau X est trié ou non
2. Estimer sa complexité au pirecas et en moyenne, en prenant comme la probabilié que le tableau n'est pas triée à partir de l'indice k est $\frac{1}{n+1}$ et qu'il est triée avec une probabilité en $\frac{1}{n+1}$

Exercice 2 Pour convertir unnombre 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 de donner la liste des nombre binaire.
2. Donner sa complexité

Exercice 3 Soit la suite (u_n) définie par : $\begin{cases} u_0 = u_1 = 1 \\ u_{n+2} = 3u_{n+1} + 1 \end{cases}$

1. Ecrire un programme récursif permettant de calculer le nième terme de la suite.
2. Estimer sa complexité.
3. Modifer le programme pour avoir une complexité linéaire.

Exercice 4 Etudiez le nombre d'additions réalisées par l'algorithme suivant dans le meilleur cas, le pire cas, puis dans le cas moyen en supposant que les tests ont une probabilité de $\frac{1}{2}$ d'être vrai

```
s=0
for i in range(n):
    if T[i]>a:
        s=s+T[i]
```

Exercice 5 La constante e peut être calculée par la limite de la somme suivante, qui s'appuie sur la factorielle $n! = n.(n-1).(n-2)\dots 2.1$ (avec $0! = 1$) :

$$e = \lim_{n\to+\infty} \sum_{i=0}^{n} \frac{1}{i!}$$

1. Ecrire une fonction nommée `factorielle` qui prend en paramètre n et renvoie $n!$.
   En donner sa complexité.
2. Ecrire un programme en python qui demande un entier n à l'utilisateur et affiche $\sum_{i=0}^{n} \frac{1}{i!}$ en utilisant la fonction `factorielle`.
   Donner la complexité de votre programme.

3. Ecrire le même programme sans la fonction `factorielle` de complexité $O(n)$.

Exercice 6 Ecrire des programmes récursifs calculant le maximum d'une liste :

1. En considérant que le maximum est le plus grand entre le dernier terme et le maximum des (n-1) premiers termes. Estimer sa complexité.
2. En considérant que le maximum est le plus grand entre les maximums des deux moitiés du tableau. Estimer sa complexité.

Exercice 7 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.)

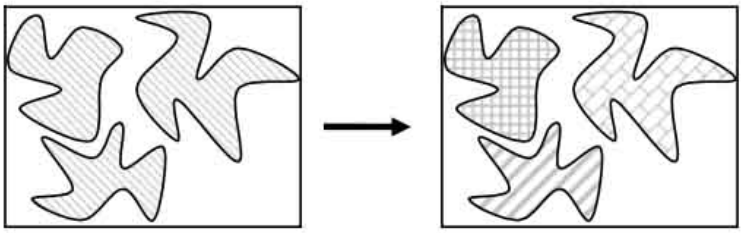

Image binaire Image étiquetée

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.

Exercice 8 Que font les programme suivant :

```
def fct1(tab, n):
    if n == 1:
        return tab[0]
    else:
        x = fcttt(tab, n-1)

    if x > tab[n-1]:
        return x
    else:
        return tab[n-1]

def fct2(a, b):
    if b==0:
        return 1

    if b%2 ==0:
        return fct2(a*a,b//2)

    return fct2(a*a,b//2)*a
```

```
def fct3(n):
    if n==0:
        return 0
    else:
        return 1+fct3(n//2)

def fct4(n):
    if n==0:
        return
    fct4(n//2)
    print(n % 2, end=" ")
```