TP 07

Algorithmes récursifs

Activité 1 – Programmation de la factorielle n !

On rappelle la définition de la factorielle : $n! = \prod_{i=1}^{n} i$ et $n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{si } n \geq 1 \end{cases}$.

Question 1 *Proposer une fonction* `fact_it(n:int)->int` *renvoyant la factorielle de n par une méthode itérative*

Question 2 *Proposer une fonction* `fact_rec(n:int)->int` *renvoyant la factorielle de n par une méthode récursive*

Activité 2 – Recherche dichotomique

On se donne une liste `L` de nombres de longueur `n`, triée dans l'ordre croissant, et un nombre `x0`.

Pour chercher `x0`, on va couper la liste en deux moitiés et chercher dans la moitié de tableau qui encadre `x0` et ainsi de suite...

On appelle `g` l'indice de l'élément du début de la sous-liste dans laquelle on travaille et `d` l'indice de l'élément de fin.

Au début, `g = 0` et `d = n - 1`.

On utilise la méthode suivante :

- on compare `x0` à « l'élément du milieu » `L[m]` avec `m = (g + d) // 2`;
- si `x0 = L[m]`, on a trouvé `x0`, on peut alors s'arrêter;
- si `x0` < `L[m]`, c'est qu'il faut chercher entre `L[g]` et `L[m-1]`;
- si `x0` > `L[m]`, c'est qu'il faut chercher entre `L[m+1]` et `L[d]`.

On poursuit jusqu'à ce qu'on a trouvé `x0` ou lorsque l'on a épuisé la liste `L`.

On donne l'algorithme de recherche dichotomique construit avec une méthode itérative.

```
def dichotomie(x0,L):
    g=0
    d=len(L)-1
    m=(g+d)//2
    while g<=d:
        if L[m]==x0:
            return True
        elif L[m]<x0:
            g=m+1
        else:
            d=m-1
        m=(g+d)//2
    return False
```

Question 3 *Écrire une fonction* `dichotomie_rec(x0,L)` *récursive qui renvoie* `True` *ou* `False` *selon que* `x0` *figure ou non dans* `L` *par cette méthode.*

Activité 3 – Recherche de palindrome

On appelle palindrome un mot qui peut se lire indifféremment de gauche à droite ou de droite à gauche. Ainsi par exemple *radar*, *rotor* ou kayak sont trois palindromes.

Question 4 *Écrire une fonction récursive* `palindrome(x:str) -> bool` *prenant en paramètre une chaîne de caractère* `x` *et qui retourne* `True` *si il s'agit d'un palindrome et* `False` *sinon. (Les espaces seront gérés comme de simples caractères). Pour cela on remarquera que :*

- *tout mot de longueur* ≤ 1 *est un palindrome;*

- *un mot est un palindrome si et seulement si ses premier et dernier caractères sont identiques et son sous-mot allant du 2ème caractère jusqu'à l'avant dernier est un palindrome.*

Activité 4 – Bubble bobble

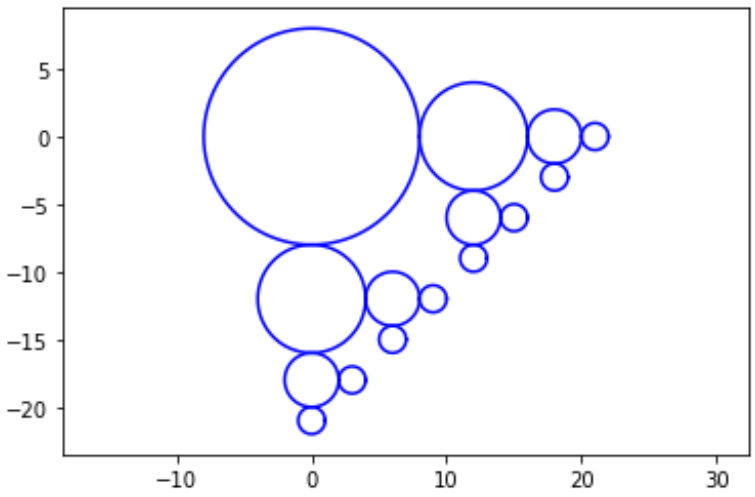
D'après ressources de Jean-Pierre Becirspahic.

On suppose disposer d'une fonction `circle(coords:list, r:float) -> None` qui trace à l'écran un cercle de centre `coords=(x,y)` et de rayon `r`.

```
import matplotlib.pyplot as plt
import numpy as np
def circle(coords:list, r:float) -> None :
    X, Y = [], []
    for t in range(101):
        X.append(coords[0]+r*np.cos(t*np.pi/50))
        Y.append(coords[1]+r*np.sin(t*np.pi/50))
    plt.plot(X, Y, 'b')
```

Dans les questions suivantes, `x` désigne l'abscisse du centre d'un cercle, `y` désigne l'ordonnée du centre d'un cercle, `r` le rayon du premier cercle et `n` le nombre de fois où le rayon sera divisé par 2.

Question 5 *Définir la fonction* `bubble1(n:int, x:float, y:float, r:float) -> None` *permettant de tracer la figure suivante. Le rayon est divisé par 2 à chaque appel.*



Question 6 *Définir la fonction* `bubble2(n:int, x:float, y:float, r:float, d:str) -> None` *permettant de tracer la figure suivante.* `d` *désigne la direction de propagation (par exemple* `n` *pour nord,* `s`*, pour sud etc...)*