TP 11
Fonctions Récursives – Sujet

Application directe

Question 1 Implémenter la fonction `mult_it(n:int, p:int) -> int` qui permet de calculer $n \times p$ par une méthode **itérative**. L'opération $*$ est strictement interdite.

Question 2 Implémenter la fonction `mult_rec(n:int, p:int) -> int` qui permet de calculer $n \times p$ par une méthode **récursive**. L'opération $*$ est strictement interdite.

Valeur approchée de Pi

Voici deux manières d'approcher le nombre π :

- le produit de Wallis :

$$\pi = 2 \prod_{i=1}^{n} \frac{4i^2}{4i^2 - 1} = 2 \times \frac{4}{3} \times \frac{16}{15} \times \frac{36}{35} \times ... \tag{0.1}$$

- la série de Gregory :

$$\pi = 4 \sum_{i=0}^{n} \frac{(-1)^i}{2i + 1} = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ...) \tag{0.2}$$

Question 3 Implémenter la fonction `Wallis_it(n:int) -> float` qui permet de calculer π par une méthode **itérative**. Tester pour **n=10** (on trouve environ 3.0677).

Question 4 Reprendre la question précédente avec une méthode **récursive**. Tester pour **n=10**.

Question 5 Implémenter la fonction `Gregory_it(n:int) -> float` qui permet de calculer π par une méthode **itérative**. Tester pour **n=10** (on trouve environ 3.232).

Question 6 Reprendre la question précédente avec une méthode **récursive**.Tester pour **n=10.**

Palindrome...

On souhaite réaliser une fonction `miroir(m:str)` dont le but est de retourner le «miroir» d'une chaîne de caractères. Par exemple le résultat de `miroir("miroir")` serait *"riorim"*.

Question 7 Programmer la fonction `miroir_rec(m:str)` permettant de répondre au problème de manière récursive. *On rappelle qu'il est possible de réaliser des opérations de slicing avec des chaînes de caractères. Ainsi si* `ch='abcdef'`, *et* `var=ch[1:3]` *alors* `var` *contient* `'bc'`.

Question 8 Que renvoie la fonction si la chaîne de caractère est "Eh ! ça va la vache" ?

Une expression qui se lit de manière identique dans les deux sens est appelée **"palindrome"**. C'est le cas de : "radar", "esope reste et se repose", "kayak" ...

Question 9 Ecrire une fonction récursive nommée `est_palindrome_rec(m:str)->bool` qui reçoit comme argument une chaîne de caractères contenant le mot (ou la phrase). Cette fonction doit tester si les deux caractères opposés sont identiques, puis appeler récursivement la fonction sur la partie de la chaîne qui ne contient pas ces deux caractères. La chaîne ne contiendra pas d'espaces.

Suite de Fibonacci

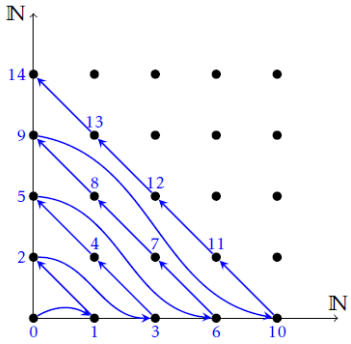
On définit la suite de Fibonacci de la façon suivante : $\forall n \in \mathbb{N}, \begin{cases} u_0 = 0, u_1 = 1 \\ u_{n+2} = u_n + u_{n+1} \end{cases}$.

Question 10 Définir la fonction `fibonacci_it(n:int)->int` permettant de calculer u_n par une méthode itérative.

Question 11 Définir la fonction `fibonacci_rec(n:int)->int` permettant de calculer u_n par une méthode récursive « intuitive».

Question 12 Observer comment passer du couple (u_n, u_{n+1}) au couple (u_{n+1}, u_{n+2}). En déduire une autre méthode récursive pour calculer le nième terme de la suite de Fibonacci.

Problème



On démontre que sur l'ensemble $\mathbb{N} \times \mathbb{N}$ est dénombrable en numérotant chaque couple $(x, y) \in \mathbb{N}^2$ suivant le procédé suggéré par la figure ci-contre.

Question 13 Écrire à la main le passage d'un point `n` au suivant `n+1` en fonction de `x` et `y`. Faire de même, dans le sens inverse avec `x` et `y` connus.

Question 14 Écrire une fonction récursive `point(x:int,y:int)->int` qui renvoie le numéro du point de coordonnées (x, y).

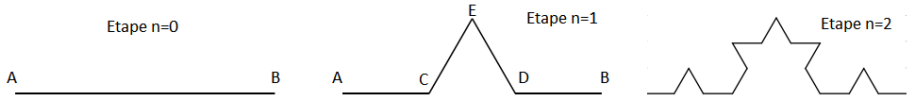
Question 15 Écrire la fonction réciproque `coordonnees(n:int)->tuple`, là encore de façon récursive.

Flocon de Von Koch – Exercice corrigé

Dans cet exercice, vous utiliserez des tableaux **numpy** pour représenter les points. C'est plus pratique que les listes python pour faire les calculs vectoriels.

- Si a et b représentent respectivement les points (x, y) et (x', y') alors $a + b$ représente le point $(x + x', y + y')$.
- Si r est un réel et a représente le point de coordonnées (x, y) alors $r * a$ représente le point (rx, ry).
- Si a et b sont des tableaux **numpy** alors $dot(a, b)$ représente le produit matriciel $a \times b$ (si ce produit est possible). La fonction **dot** est une fonction **numpy**.

Le mathématicien suédois Von Koch a défini la courbe du même nom dont voici les premières itérations.



Question 16 Ecrire une fonction `rotation` d'argument un réel `alpha` qui renvoie le tableau **numpy** correspondant à la matrice de rotation d'angle `alpha`.

Question 17 Pour l'étape $n = 1$, exprimer les points C et D en fonction de A et B.

Question 18 En utilisant une matrice de rotation, exprimer E en fonction de C et D.

Question 19 En déduire une fonction récursive `koch` d'arguments les points `A` et `B` et un entier `n`. Cette fonction tracera la courbe de Von Koch pour l'itération n à partir des points A et B.

Question 20 Ecrire une fonction `flocon` d'arguments les points A et B et un entier n. Cette fonction tracera le flocon de Von Koch pour l'itération n à partir des points A et B.

```
import numpy as np
import matplotlib.pyplot as plt
def rotation(angle):
    return np.array([[np.cos(angle),-np.sin(angle)],[np.sin(angle),np.cos(
    angle)]])
def koch(a, b, n):
    R = rotation(np.pi/3)
    if n == 0:
        plt.plot([a[0], b[0]], [a[1], b[1]],'b')
    else:
        c=np.array([(b[0]-a[0])/3+a[0],(b[1]-a[1])/3+a[1]])
        d=np.array([2*(b[0]-a[0])/3+a[0],2*(b[1]-a[1])/3+a[1]])
        vecteur=d-c
        e=np.dot(rotation(np.pi/3),vecteur)+c
        koch(a, c, n - 1)
        koch(c, e, n - 1)
        koch(e, d, n - 1)
        koch(d, b, n - 1)
def flocon(a,b,n):
    koch(a,b,n)
    vecteur=b-a
    c=np.dot(rotation(-2*np.pi/3),vecteur)+b
    koch(b,c,n)
    koch(c,a,n)
n = 5
a = np.array([0,0])
b = np.array([1,0])
flocon(a, b, n)
plt.axis('equal')
plt.show()
```