Nom : ....

Prénom : ....

2023/10/20

Classe : MP/PSI

Lycée Loges

ITC : DS n° 1 - 3heures

type CCP / e3a

calculatrice interdite

Exercice I : $\cdots$ /-

Exercice II : $\cdots$ /-

Exercice III : $\cdots$ /-

Note finale : $\cdots$ **/ 20**

Les trois exercices sont indépendants.

⚠ Faites attention à la lisibilité de votre code et vos explications

Exercice I

Si A est une matrice stochastique strictement positive, on a établi dans la partie précédente la convergence de la suite $(\mu_n)_{n\in\mathbb{N}}$ associée à la matrice A. Ceci fournit un algorithme de calcul de la probabilité invariante par A. On en propose une implémentation en langage Python. On sera très attentif à la rédaction et notamment à l'indentation du code.

Un vecteur x de $\mathbb{R}^p$ sera représenté en Python par une liste de flottants. Par exemple, le vecteur $x = (1,2,3)$ de $\mathbb{R}^3$ sera représenté par la liste `[1,2,3]`. De même, une matrice A sera représentée par une liste dont les éléments sont les lignes de la matrice. Par exemple, la matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ sera représentée par la liste `[[1,2,3],[4,5,6]]`.

☐ **Q1 -** Définir sous python la matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$.

☐ **Q2 -** Donner les valeurs renvoyées lorsque l'on exécute `len(A)`, `A[1]` et `A[2][1]`.

☐ **Q3 -** Écrire une fonction `difference` qui prend en arguments deux vecteurs x et y de même taille et renvoie le vecteur $x - y$. Par exemple si $x = (5,2)$ et $y = (3,7)$, `difference(x,y)` renverra `[2,-5]`.

☐ **Q4 -** Écrire une fonction `norme` qui prend en argument un vecteur $x = (x_1, \ldots ,x_p)$ et renvoie sa norme infinie $\|x\|_\infty = \max\{|x_i| \ tq \ \ i \in [\![1;p]\!]\}$ (on pourra utiliser librement la fonction `abs` qui renvoie la valeur absolue d'un nombre, mais on s'interdit l'utilisation de la fonction `max` déjà implémentée dans Python).

☐ **Q5 -** Écrire une fonction `itere` qui prend en arguments un vecteur ligne x et une matrice carrée de même taille que x et qui renvoie le vecteur xA. Par exemple si $x = (1,1)$ et $A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$, on a $xA = (5,7)$ et donc `itere(x,A)` renverra `[5,7]`.

☐ **Q6 -** On admet que si A est une matrice strictement positive, la suite de vecteurs lignes de $\mathbb{R}^p$ associée $(\mu_n)_{n\in\mathbb{N}}$ définie par la relation : $\forall n \in \mathbb{N},\ \mu_{n+1} = \mu_n A$ converge vers un vecteur μ_∞ indépendant du choix de μ_0 vecteur stochastique.

Écrire une fonction `probaInvariante` qui prend en arguments une matrice stochastique strictement positive A de $\mathscr{M}_p(\mathbb{R})$ et un réel $\varepsilon > 0$ et qui renvoie le premier terme μ_k de la suite $(\mu_n)_{n\in\mathbb{N}}$ avec $\mu_0 = \left(\frac{1}{p}, \frac{1}{p}, \ldots, \frac{1}{p}\right)$ tel que $\|\mu_k - \mu_{k-1}\|_\infty \leqslant \varepsilon$. On ne demandera pas à l'algorithme de vérifier que la matrice passée en argument est bien stochastique et strictement positive.

Par exemple, si $A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix}$ et $\varepsilon = 10^{-6}$, `probaInvariante(A,eps)` renverra `[0.3333339691, 0.666666030]`.

Exercice II

Dans cet exercice "Algorithme de décomposition primaire d'un entier" (*Informatique pour tous*), on se propose d'écrire un algorithme pour décomposer un entier en produit de nombres premiers. Les algorithmes demandés doivent être écrits en langage `python`. On sera très attentif à la rédaction et notamment à l'indentation du code.

On définit la valuation p-adique pour p nombre premier et n entier naturel non nul.

- Si p divise n, on note $v_p(n)$ le plus grand entier k tel que p^k divise n.
- Si p ne divise pas n, on pose $v_p(n) = 0$.

L'entier $v_p(n)$ s'appelle la valuation p-adique de n.

☐ **Q1 -** Écrire une fonction booléenne `estPremier(n)` qui prend en argument un entier naturel non nul n et qui renvoie le booléen `True` si n est premier et le booléen `False` sinon.

On pourra utiliser le critère suivant : un entier $n \geqslant 2$ qui n'est divisible par aucun entier $d \geqslant 2$ tel que $d^2 \leqslant n$, est premier.

☐ **Q2 -** En déduire une fonction `liste_premiers(n)` qui prend en argument un entier naturel non nul n et renvoie la liste des nombres premiers inférieurs ou égaux à n.

☐ **Q3 -** Pour calculer la valuation 2-adique de 40, on peut utiliser la méthode suivante :

- 40 est divisible par 2 et le quotient vaut 20
- 20 est divisible par 2 et le quotient vaut 10
- 10 est divisible par 2 et le quotient vaut 5
- 5 n'est pas divisible par 2.

La valuation 2-adique de 40 vaut donc 3.

Écrire une fonction `valuation_p_adique(n, p)` **non récursive** qui implémente cet algorithme. Elle prend en arguments un entier naturel n non nul et un nombre premier p et renvoie la valuation p-adique de n.
Par exemple, puisque $40 = 2^3 \times 5$,

- `valuation_p_adique(40, 2)` renvoie 3,
- `valuation_p_adique(40, 5)` renvoie 1 et
- `valuation_p_adique(40, 7)` renvoie 0.

☐ **Q4 -** Écrire une deuxième fonction cette fois-ci **récursive**, `val(n, p)` qui renvoie la valuation p-adique de n.

☐ **Q5 -** En déduire une `fonction decomposition_facteurs_premiers(n)` qui calcule la décomposition en facteurs premiers d'un entier $n \geqslant 2$.

Cette fonction doit renvoyer la liste des couples $(p, vp(n))$ pour tous les nombres premiers p qui divisent n. Par exemple, `decomposition_facteurs_premiers(40)` renvoie la liste `[[2, 3],[5, 1]]`.

Exercice III

Partie A : Recherche de zéro d'une fonction

☐ **Q1 -** Soient a et b deux réels , $f\ :\ [a;b] \longrightarrow \mathbb{R}$ une fonction continue telle que $f(a)f(b) < 0$.

a) Justifier que f s'annule sur $[a;b]$.

b) Écrire en *python* une fonction `recherche_dicho` prenant en arguments une fonction `f`, deux flottants `a` et `b` tels que $f(a)f(b) < 0$ et une precision `eps` et qui renvoie un couple de réels encadrant un zéro de f à une précision `eps` près.

☐ **Q2 -** Soit f une fonction continue de $[0;1]$ dans $[0;1]$

a) Montrer que f admet un point fixe (c'est à dire une réel c de $[0;1]$ tel que $f(c) = c$).

b) Ecrire en *python* une fonction `recherche_ptfixe` qui prend en argument une fonction `f` que l'on suppose continue de $[0;1]$ dans $[0;1]$, une precision `eps` et qui renvoie un couple de réels encadrant un point fixe de f à une précision `eps` près. On pourra utiliser la fonction `recherche_dicho`.

Partie B : Recherche dans une liste

☐ **Q1 -** On propose l'algorithme suivant :

```python
def recherche_dicho(L,g,d,x):
    # L est une liste, telle que L[g,d+1] est triee
    if x>L[d]:
        return d+1
    else:
        a=g
        b=d
        while a!=b:
            c=(a+b)//2      # ligne 9
            if x<=L[c]:
                b=c
            else:
                a=c+1
    return a
```

a) On prend $L = [2{,}4{,}5{,}7{,}7{,}8{,}10]$. Que renvoient les instructions suivantes ?

```
>>> recherche_dicho(L,1,5,6)
>>> recherche_dicho(L,0,5,1)
```

On donnera les valeurs prises successivement par les variables `a` et `b` à chaque passage à la ligne 9.

b) Détailler clairement ce que fait le programme `recherche_dicho`.

c) Déterminer en le justifiant la complexité du programme, mesurée en nombre de comparaisons. On utilisera, si besoin est, la notation $\mathcal{O}$, et on pourra exprimer cette complexité en fonction d'un ou plusieurs paramètres parmi `len(L)`, `g`, `d` et `x`.

☐ **Q2 -** Proposer un algorithme `tri_dicho` de tri par insertion utilisant la fonction `recherche_dicho` pour trouver la position à laquelle insérer l'élément.

☐ **Q3 -** Estimer le nombre d'affectations de `tri_dicho` ainsi que le nombre de comparaisons effectuées par l'algorithme `tri_dicho`. Comparer avec le tri par insertion classique.