

Examen Semestre 2
Matière : Informatique

Classes : MP1, PC1, PT1
Date : 18 Mai 2022

Nombre de pages : 3
Durée : 2 h

Exercice 1 (7 points)

En mathématiques, la fonction "**somme des puissances k -ièmes des diviseurs**", notée σ_k , est la fonction qui à tout entier $n > 0$, associe la somme des puissances k -ièmes des diviseurs positifs de n , où k est un entier positif quelconque :

$$\sigma_k(n) = \sum_{d|n} d^k \quad \text{où } d|n \text{ signifie « } d \text{ divise } n \text{ ».}$$

Par exemple, $\sigma_0(12)$ est le nombre des diviseurs de 12 :

$$\sigma_0(12) = 1^0 + 2^0 + 3^0 + 4^0 + 6^0 + 12^0 = 1 + 1 + 1 + 1 + 1 + 1 = 6,$$

tandis que $\sigma_1(12)$ est la somme de tous les diviseurs :

$$\sigma_1(12) = 1^1 + 2^1 + 3^1 + 4^1 + 6^1 + 12^1 = 1 + 2 + 3 + 4 + 6 + 12 = 28$$

1) Ecrire une fonction **somme(n, k)** qui prend en argument les entiers n et k et retourne la somme $\sigma_k(n)$.

On se propose de créer un fichier texte comportant trois colonnes dont les valeurs sont séparées par des virgules :

- 1^{ère} colonne l'entier n (la valeur de n varie de 1 à 100).
- 2^{ème} colonne $\sigma_0(n)$
- 3^{ème} colonne $\sigma_1(n)$

Voici un extrait :

1, 1, 1

2, 2, 3

3, 2, 4

...

99, 6, 156

100, 9, 217

2) Ecrire une fonction **sigma(nom)** qui prend en argument un nom de fichier et permet de créer un fichier texte conformément à la disposition de valeurs ci-dessus.

3) Ecrire une fonction **graphe(nom)** qui prend en argument un nom de fichier et permet de tracer sur la même fenêtre les graphes $\sigma_0(n)$ et $\sigma_1(n)$ en fonction de n .

La fonction doit :

- ouvrir le fichier **nom** et parcourir toutes ses lignes,
- enregistrer dans une liste x les valeurs de n ,
- enregistrer dans une liste y les valeurs de $\sigma_0(n)$,
- enregistrer dans une liste z les valeurs de $\sigma_1(n)$,
- tracer la superposition des graphes $\sigma_0(n)$ et $\sigma_1(n)$ en appelant l'axe des abscisses n et l'axe des ordonnées σ .

Nota :

- les listes x , y et z doivent être extraites à partir du fichier et non pas calculées par la fonction *somme*
- La commande **plt.xlabel('x')** permet de nommer l'axe des abscisses x
- La commande **plt.ylabel('f(x)')** permet de nommer l'axe des ordonnées $f(x)$

Exercice 2 : Tri comptage (6 points)

Le **tri comptage**, appelé aussi **tri casier**, est un algorithme de tri par dénombrement qui s'applique sur des valeurs **entières**.

Principe de l'algorithme :

- On recherche dans l'ensemble d'éléments (stockés dans une liste L), le plus petit élément Min et le plus grand élément Max .
- On construit une liste C de taille $(Max - Min + 1)$ initialisée à 0.
- On parcourt L et pour chaque élément $L[i]$, on incrémente la case d'indice $(L[i] - Min)$ de C tel que $C[L[i] - Min]$ contient le nombre d'occurrences de $L[i]$ dans L ; c'est-à-dire :
 - $C[0]$ contient le nombre d'apparition de la valeur Min dans L ,
 - $C[1]$ contient le nombre d'apparition de la valeur $Min+1$ dans L ,
 - $C[k]$ contient le nombre d'apparition de la valeur $Min+k$ dans L ,
 - $C[Max - Min]$ contient le nombre d'apparition de la valeur Max dans L .
- On parcourt ensuite la liste C d'occurrences, et on remplit au fur et à mesure la version triée de la liste L initiale.

Exemple : Soit la liste L d'entiers à trier :

$L =$

4	0	-1	2	0	0	2	5	-1	2	0	4	2
---	---	----	---	---	---	---	---	----	---	---	---	---

Le plus petit élément $Min = -1$ et le plus grand élément $Max = 5$, la liste C de taille $Max - Min + 1 = 7$ est donc construite. Ensuite, on considère chaque élément $L[i]$ de la liste L et on incrémente la case d'indice $(L[i] - Min)$ de la liste C .

Pour cet exemple, le calcul d'occurrences des éléments de L aboutit à la liste C suivante :

$C =$

2	4	0	4	0	2	1
---	---	---	---	---	---	---

1) Ecrire une fonction **Casier(L)** qui prend en argument une liste L et retourne la liste C .

2) En parcourant ensuite la liste C , on peut remplir au fur et à mesure la version triée de la liste initiale :

D'après le tableau C on a :	Remplissage au fur et à mesure de la version triée :											
2 fois -1	-1	-1										
4 fois 0	-1	-1	0	0	0	0						
0 fois 1	-1	-1	0	0	0	0						
4 fois 2	-1	-1	0	0	0	0	2	2	2	2		
0 fois 3	-1	-1	0	0	0	0	2	2	2	2		
2 fois 4	-1	-1	0	0	0	0	2	2	2	2	4	4
1 fois 5	-1	-1	0	0	0	0	2	2	2	2	4	5

La liste triée par ordre croissant est alors :

-1	-1	0	0	0	0	2	2	2	2	4	4	5
----	----	---	---	---	---	---	---	---	---	---	---	---

Ecrire une fonction **Tri_comptage(L)** qui prend en argument une liste L et retourne une liste T composée par les éléments de L triés par ordre croissant. Cette fonction doit appeler la fonction **Casier** de la question 1.

Exercice 3 (7 points)

1) Soit A un tableau *numpy* de taille $m \times n$. On se propose de remplir un vecteur ligne *numpy* T de taille $m.n$ par les éléments de A selon le parcours des éléments de A indiqué par le schéma suivant :

Exemple : $A(4 \times 5)$

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

T :

1	2	3	4	5	10	9	8	...	19	18	17	16
---	---	---	---	---	----	---	---	-----	----	----	----	----

Ecrire une fonction **Remplir1(A)** qui prend en argument le tableau A et retourne le vecteur T .

2) Soit A un tableau *numpy* de taille $m \times n$. On se propose de remplir un vecteur ligne *numpy* T de taille $m.n$ par les éléments de A selon le parcours des éléments de A indiqué par le schéma suivant :

Exemple : $A(4 \times 5)$

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

T :

1	6	11	16	17	12	7	2	...	5	10	15	20
---	---	----	----	----	----	---	---	-----	---	----	----	----

Ecrire une fonction **Remplir2(A)** qui prend en argument le tableau A et retourne le vecteur T .

3) Matrice de Toeplitz

Une matrice de *Toeplitz* ou matrice à **diagonales constantes** est une matrice de taille $m \times n$ dont les coefficients sur une diagonale descendante de gauche à droite sont les mêmes, c'est-à-dire dont les indices vérifient la relation $a_{i,j} = a_{i+1,j+1}$ avec $0 \leq i \leq m-2$, $0 \leq j \leq n-2$. Par exemple, la matrice $A(5 \times 5)$ suivante est une matrice de *Toeplitz* :

$$A = \begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix}$$

Ecrire une fonction **Toeplitz(A)** qui prend en argument un tableau A de type *numpy* et retourne **True** si A est une matrice de *Toeplitz* et **False** sinon.