Concours National Commun (CNC), Maroc

Session 2022, filière PSI

Informatique

Durée : 2 heures

Ce que traite le sujet

- Exercice : approximation du nombre pi
- Partie I : base de données et langage SQL
- Partie II : calcul numérique, méthode de Brent
- Partie III : problème, traitement des images numériques

Sujet reproduit sans retouche à partir de la page suivante.

Résumé des parties, chapitres de cours liés et autres années :
prepahelp.com/annales/cnc/psi/2022/info

Prepa Help ne revendique aucun droit sur le sujet, propriété de l'organisateur du concours.

ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ
ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵣⵔⴼⵜ ⴰⵏⴰⵎⵓⵔ
ⴰⵙⵓⵙⵙ ⴰⵎⴰⵡⴰⵍ ⴰⵏⴰⵎⵓⵔ



المملكة المغربية
وزارة التعليم العالي
والبحث العلمي والابتكار

ROYAUME DU MAROC
Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de l'Innovation

المدرسة الوطنية العليا للمعادن بالرباط
ⵜⵉⵎⵙⵙⵓⵔⵉⵏ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵣⵔⴼⵜ ⵏ ⵍⵎⵄⴰⴷⵏ ⴰⵔⵔⴱⴰⵟ
ECOLE NATIONALE SUPERIEURE DES MINES DE RABAT

وزارة الانتقال الطاقي
Ministère de la Transition Énergétique
والتنمية المستدامة
et du Développement Durable
Département de la Transition Énergétique قطاع الانتقال الطاقي



MINES-RABAT

CNC 2022

Concours National Commun

d'Admission dans les Établissements de Formation d'Ingénieurs et Établissements Assimilés

Épreuve d'**INFORMATIQUE**

Filière : **PSI**

Durée **2 heures**

Cette épreuve comporte **12 pages au format A4**, en plus de la page de garde

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- ✓ Cette épreuve est composée d'un exercice et de trois parties tous indépendants ;
- ✓ Toutes les instructions et les fonctions demandées seront écrites en Python ;
- ✓ Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- ✓ Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈

Important **: Le candidat doit impérativement commencer par traiter toutes les questions de l'exercice ci-dessous, et écrire les réponses dans les premières pages du cahier de réponses.**

Exercice : (4 points)

Approximation du nombre π

La constante **Pi** (**π**) est l'un des nombres les plus importants et les plus fascinants dans le monde des mathématiques. C'est un nombre irrationnel, son écriture est faite d'une infinité de décimales, sans aucune périodicité. On n'arrive qu'à déterminer une écriture décimale approchée de sa valeur réelle. Pour cela, plusieurs méthodes pratiques et efficaces ont été employées.

La formule suivante a été publiée par le mathématicien indien **Nilakantha Somayaji** au 15e siècle, elle donne **11** décimales correctes de **π** :

$$\pi \approx 3 + \sum_{k=1}^{n} \frac{4(-1)^{k+1}}{2k(2k+1)(2k+2)}$$

Q1- Écrire la fonction ***puiss (k)*** qui reçoit en paramètre un entier strictement positif **k**, et qui retourne la valeur de $(-1)^k$, en utilisant la parité de **k** : Si **k** est pair, alors la fonction retourne **1**. Et si **k** est impair, alors la fonction retourne **-1**.

Q2- Déterminer la complexité de la fonction ***puiss (k)***, avec justification.

Q3- Écrire la fonction ***terme (k)*** qui reçoit en paramètre un entier strictement positif **k**, et qui retourne la valeur du terme suivant :

$$\frac{4(-1)^{k+1}}{2k(2k+1)(2k+2)}$$

Q4- Écrire la fonction ***liste_termes (n)*** qui reçoit en paramètre un entier strictement positif **n**. La fonction retourne une nouvelle liste ***L*** qui contient les valeurs de ***terme(k)*** pour ***k=1, 2, 3, …, n***

Q5- Écrire la fonction ***somme (L)*** qui reçoit en paramètre une liste **L** de nombres réels, et qui retourne la valeur de la somme des éléments de **L**, sans utiliser la fonction prédéfinie sum().

Q6- Écrire la fonction ***pi_nilakantha (n)*** qui reçoit en paramètre un entier strictement positif **n**, et qui retourne la valeur approchée de **π**, en utilisant la formule de Nilakantha.

Partie I : Base de données et langage SQL

Internationaux de tennis

On considère une base de données qui permet la gestion des joueurs de tennis, ayant gagné au moins un tournoi international de tennis simple. Cette base de données est composée de **3** tables : **'Joueur'**, **'Gagne'** et **'Tournoi'**.

Joueur	
id	*(entier)*
nomJr	*(texte)*
genre	*(texte)*
nationalité	*(texte)*

Gagne	
idJr	*(entier)*
codTr	*(entier)*
année	*(entier)*
prime	*(réel)*

Tournoi	
code	*(entier)*
nomTr	*(texte)*
catégorie	*(texte)*
pays	*(texte)*

a- Structure de la table 'Joueur'

La table **'Joueur'** contient des informations sur les joueurs ayant gagné au moins un tournois international de tennis simple : Le champ **id** est la clé primaire, ce champ contient un entier unique pour identifier chaque joueur. Le champ **nomJr** contient le nom complet de chaque joueur. Le champ **genre** contient **'F'** pour féminin, ou **'M'** pour masculin. Le champ **nationalité** contient le pays d'origine de chaque joueur.

Exemples :

id	nomJr	genre	nationalité
2154	Roger Federer	M	Suisse
3432	Serena Williams	F	États Unis
2691	Novak Djokovic	M	Serbie
2374	Rafael Nadal	M	Espagne
...	...	...	...

b- Structure de la table 'Tournoi'

La table **'Tournoi'** contient des informations sur les tournois internationaux de tennis simples. Le champ **code** est la clé primaire, ce champ contient un entier unique pour chaque tournoi. Les champs **nomTr** et **catégorie** contiennent le nom et la catégorie de chaque tournoi. Le champ **pays** contient le pays dans lequel se déroule chaque tournoi.

Exemples :

code	nomTr	catégorie	pays
23	Roland-Garros	Grand Chelem	France
174	Doha	ATP 250	Qatar
10	Open d'Australie	Grand Chelem	Australie
59	Sydney International	ATP 250	Australie
85	Rotterdam	ATP 500	Pays-Bas
96	Masters d'Indiana Wells	Masters 1000	États Unis
...	...	...	...

c- Structure de la table 'Gagne'

La table **'Gagne'** contient des informations sur les joueurs et les tournois gagnés par chaque joueur. Les champs **idJr** et **codTr** sont deux clés étrangères qui font respectivement référence aux clés primaires **id** et **code**. Le champ **année** contient l'année de chaque tournoi. Chaque tournoi est organisé une seule fois chaque année. Le champ **prime** contient le montant reçu par le joueur gagnant de chaque tournoi, et il est exprimé en million de dollars. La prime d'un tournoi peut changer au cours des années.

Exemples :

idJr	codTr	année	prime
2374	23	2019	2.30
2374	23	2020	1.60
2374	23	2021	1.40
3432	85	2018	2.50
2691	10	2020	4.12
2691	10	2021	2.75
...	...	...	...

Q.1 – Rédiger une requête SQL qui supprime, dans la table '**Tournoi**', tous les tournois des deux catégories '**WTA Premier Mandatory**' et '**WTA Premier 5**'.

Q.2 – Rédiger une requête SQL, qui permet d'obtenir les différentes catégories des tournois, triées dans l'ordre alphabétique.

Q.3 – Rédiger en algèbre relationnelle, une requête qui permet d'obtenir les pays qui organisent des tournois de tennis dans la catégorie '**ATP 500**' et dans la catégorie '**Master 1000**'.

Q.4 – Écrire la requête **Q.3** en langage SQL.

Q.5 – Rédiger une requête SQL, qui permet d'obtenir les codes des tournois, la plus grande prime, la plus petite prime et la moyenne des primes de chaque tournoi, triés dans l'ordre décroissant de la moyenne des primes.

Q.6 – Rédiger une requête SQL, qui permet d'obtenir les noms et genres des joueurs, le compte des tournois gagnés par chaque joueur et la somme totale des primes reçue par chaque joueur. La somme totale des primes reçue par chaque joueur doit être supérieure à **75** millions de dollars.

Q.7 – Les tournois de la catégorie **Grand-Chelem** sont 4 : **'Open d'Australie'**, **'Roland-Garros'**, **'Wimbledon'** et **'US Open'**.

Rédiger une requête SQL, qui permet d'obtenir les années, les noms, les genres et les nationalités des joueurs qui ont gagné les **4** tournois du **Grand-Chelem** la même année, triés dans l'ordre décroissant des années.

Partie II : Calcul numérique

Recherche du zéro d'une fonction

Méthode 'Brent'

Dans cette partie, on suppose que les modules **numpy** et **matplotlib.pyplot** sont importés :

```
import numpy as np
import matplotlib.pyplot as plt
```

Calculer les zéros d'une fonction réelle f (c'est-à-dire les racines de l'équation $f(x) = 0$) est un problème que l'on rencontre très souvent en calcul scientifique. En général, cette tâche ne peut être effectuée en un nombre fini d'opérations. Par exemple, il n'existait pas de formule explicite donnant les racines d'un polynôme quelconque de très grand degré. La situation est bien sûr encore plus complexe quand f n'est pas un polynôme.

Pour résoudre ce problème, on utilise donc des méthodes itératives ou récursives : partant d'une ou plusieurs valeurs initiales, on construit une suite de valeurs x_n qui converge vers un zéro de la fonction f considérée.

Un problème simple et concret qui donne lieu à une équation non linéaire est celui de l'**équation d'état d'un gaz** : Nous voulons déterminer le volume ***V*** occupé par un gaz dont la température est ***T*** et dont la pression est ***p***. L'équation d'état, (liant ***p***, ***V*** et ***T***) est donnée par :

$$\left(\mathrm{p} + \frac{\mathrm{aN}^2}{\mathrm{V}^2}\right) * (\mathrm{V} - \mathrm{Nb}) = \mathrm{kNT}$$

Avec :

- ***a*** et ***b*** sont deux coefficients qui dépendent du gaz considéré ;
- ***N*** est le nombre de molécules contenues dans le volume ***V*** ;
- ***p*** est la pression ;
- ***T*** est la température ;
- ***k*** est la constante de Boltzmann.

Nous devons donc résoudre une équation non linéaire dont la solution est ***V***. Pour cela, on doit calculer les zéros de la fonction g définit par :

$$g(v) = \left(p + \frac{aN^2}{v^2}\right) * (v - N.b) - k.N.T$$

On suppose que les valeurs des paramètres sont les suivantes, (*pour le dioxyde de carbone CO_2*) :

a=0.40 , b=42.7 * 10^{-6} , N=10^3 , T=300 , p=3.5*10^7 et k=1.38*10^{-23}

Q.1- Définir la fonction $g(v)$ en langage Python.

Q.2- Écrire le programme python qui permet de tracer la représentation graphique, ci-dessous (Figure 1), de la fonction g. Le nombre de points générés dans la courbe est **500**.

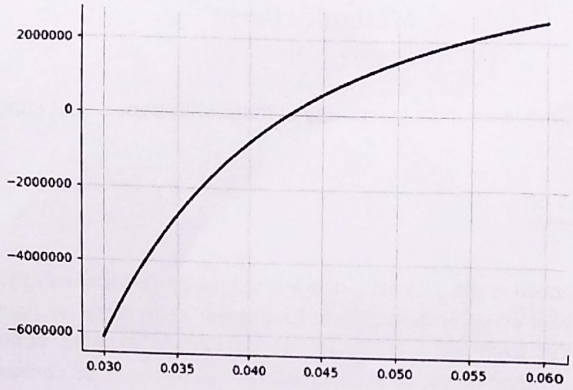


Figure 1

À partir de cette représentation graphique, on voit bien que la fonction g admet un zéro dans l'intervalle : **[0.040 , 0.045]**.

- La **méthode de la dichotomie** (ou de la bissection) est un algorithme de recherche d'un zéro d'une fonction. Cette méthode consiste à répéter de partager un intervalle en deux parties, et sélectionner le sous-intervalle dans lequel existe le zéro de la fonction.

- La **méthode de la sécante** est un algorithme de recherche d'un zéro d'une fonction. Cette méthode est basée sur la relation de récurrence suivante :

  Étant donnés deux réels différents a et b, qui représentent deux approximations initiales du zéro d'une fonction f. On obtient la prochaine approximation d à l'aide de la formule suivante :

$$d = b - f(b)\frac{b-a}{f(b)-f(a)}$$

- La **méthode d'interpolation quadratique inverse** est un algorithme de recherche d'un zéro d'une fonction. L'idée de cette méthode et d'utiliser une interpolation quadratique afin d'approcher la fonction inverse. Cette méthode est basée sur la relation de récurrence suivante :
  Étant donnés trois réels a, b et c, qui représentent trois approximations initiales du zéro d'une fonction f. On obtient la prochaine approximation d à l'aide de la formule suivante :

$$d = \frac{a\,f(b)f(c)}{(f(a)-f(b))(f(a)-f(c))} + \frac{b\,f(a)f(c)}{(f(b)-f(a))(f(b)-f(c))} + \frac{c\,f(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}$$

- La **méthode de Brent** est un algorithme de recherche d'un zéro d'une fonction, due à *Theodorus Dekker (1969)* et à *Richard Brent (1973)*. Cette méthode combine les **3** méthodes précédentes. À chaque itération, elle décide laquelle de ces trois méthodes est susceptible d'approcher au mieux le zéro de la fonction, et effectue une itération en utilisant cette méthode. L'idée principale est d'utiliser la méthode de la sécante ou d'interpolation quadratique inverse parce qu'elles convergent vite, et de revenir à la méthode de dichotomie si besoin est. Cela donne une méthode robuste et rapide, très populaire et très appréciée.

Algorithme de la méthode de Brent :

Entrée : f est une fonction continue ayant un zéro dans un intervalle $[a\,;b]$, telle que $f(a)$ et $f(b)$ ont des signes opposés, et **eps** est la précision désirée.

$k \leftarrow Vrai$

$c \leftarrow a$

Tant que $f(b) \neq 0$ *et* $|b - a| > eps$ *(condition de convergence)*

Si $|f(a)| < |f(b)|$ Alors

échanger a et b

Fin si

Si $f(a), f(b)$ *et* $f(c)$ sont différents Alors

$d \leftarrow$ la valeur de la formule de l'interpolation quadratique inverse, appliquée pour a, b et c

Sinon

$d \leftarrow$ la valeur de la formule de la sécante, appliquée pour a et b

Fin si

Si (d n'est pas compris entre $(3a + b)/4$ et b) ou (**k** est **Vrai** et $|d - b| \geq |b - c|/2$) ou (**k** est **Faux** et $|d - b| \geq |c - p|/2$) Alors

$d \leftarrow$ la valeur du milieu de a et b

$k \leftarrow Vrai$

Sinon

$k \leftarrow Faux$

Fin si

$p \leftarrow c$

$c \leftarrow b$

Si $f(a)$ *et* $f(d)$ sont de même signe Alors

$a \leftarrow d$

Sinon

$b \leftarrow d$

Fin si

Fin tant que

Sortie : retourner b

Q.3- Écrire la fonction ***valeur*** (f, x, y, z), qui reçoit en paramètres une fonction f, et trois réels x, y *et* z, tels que $f(x), f(y)$ *et* $f(z)$ sont différents. La fonction retourne la valeur de l'expression suivante :

$$\frac{x \,.\, f(y) \,.\, f(z)}{\big(f(x) - f(y)\big) \,.\, \big(f(x) - f(z)\big)}$$

Q.4- Écrire la fonction ***brent_racine*** (f, a, b, eps), qui reçoit en paramètres une fonction continue f ayant un zéro dans l'intervalle $[a\,;b]$, et telle que $f(a)$ et $f(b)$ ont des signes opposés, et un réel **eps** qui représente la précision choisie. En utilisant la méthode de Brent, la fonction retourne la valeur du zéro de la fonction f.

Q.5- Écrire le programme python qui permet d'afficher la valeur du zéro de la fonction g (voir Figure1), en utilisant la méthode de Brent, sachant que la valeur de la précision choisie est : **eps = 10^{-12}**

Partie III : Problème

Traitement des images numériques

Une image numérique désigne, dans son sens le plus général, toute image qui a été acquise, traitée et sauvegardée sous une forme codée et représentable par des nombres (valeurs numériques).

Par traitement d'images, on désigne l'ensemble des opérations informatiques et mathématiques sur les images numériques, qui transforment une image en une autre image.

Une image numérique en **niveaux de gris** est représentée par une matrice à deux dimensions. Chaque élément de la matrice est appelé : **pixel**. Chaque pixel contient un nombre entier compris entre **0** et **255**. Le nombre **0** correspond au noir, le nombre **255** correspond au blanc, et chacun des autres nombres **1**, **2**, **3**,..., **253**, **254** correspond à un niveau de gris entre le noir et le blanc. Ainsi l'image est codée sur **256** niveaux de gris.

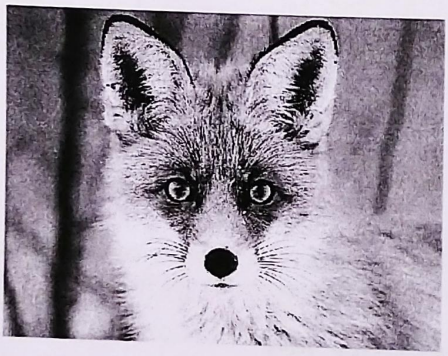


Figure 1

Par exemple, l'image ci-dessus (figure 1) est représentée par la matrice à **500** lignes et **690** colonnes, suivante :

$$\begin{bmatrix} 132 & 131 & 132 & \dots & 143 & 146 & 145 \\ 129 & 132 & 131 & \dots & 146 & 144 & 144 \\ 133 & 130 & 128 & \dots & 144 & 145 & 145 \\ \vdots & & & \vdots & & & \vdots \\ \vdots & & & \vdots & & & \vdots \\ \vdots & & & \vdots & & & \vdots \\ 162 & 161 & 164 & \dots & 190 & 190 & 187 \\ 163 & 160 & 161 & \dots & 192 & 189 & 189 \\ 165 & 160 & 161 & \dots & 189 & 187 & 187 \end{bmatrix}$$

NB : Pour plus de clarté, on utilisera l'image de la figure 1, dans tous les exemples de cette partie.

1- Taille d'une image en niveaux de gris :

Q.1.a- Quel est le nombre minimum de bits nécessaire pour coder, en binaire, un nombre entier sans signe compris entre **0** et **255** ?

Q.1.b- Déduire, en **Kilo-Octet**, la taille mémoire nécessaire pour stocker l'image de la figure 1.

Représentation d'une matrice :

Pour représenter une matrice de **n** lignes et **p** colonnes, on utilise une liste contenant **n** listes toutes de même longueur **p**. Par exemple, la matrice de l'image de la figure 1 est représentée par la liste **M**, composée de **500** listes, de taille **690** chacune :

```
M = [ [132, 131, 132,   ...   143, 146, 145],
      [129, 132, 131,   ...   146, 144, 144],
      [133, 130, 128,   ...   144, 145, 145],
       ...              ...
      [165, 160, 161,   ...   189, 187, 187] ]
```

M[i][j] est la valeur du pixel $M_{i,j}$ qui se trouve à la $i^{ème}$ ligne et la $j^{ème}$ colonne dans l'image.

Par exemple, la valeur du pixel **M[2][1]** est **130**, et qui représente un niveau de gris.

M[i] est la liste qui représente la ligne d'indice ***i*** dans l'image.

Par exemple, **M[1]** est la liste d'indice **1** dans **M** : **[129, 132, 131, ... , 146, 144, 144]**

2- Inverse d'une image :

Pour inverser les niveaux de gris d'une image, on remplace la valeur de chaque pixel $M_{i,j}$ par la valeur de la différence $255 - M_{i,j}$. Ainsi, les régions sombres deviennent claires, le noir devient blanc, et vice versa.

Exemple :



Q.2- Écrire la fonction ***inverse (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris. La fonction inverse les niveaux de gris de l'image ***M***.

3- Luminosité d'une image :

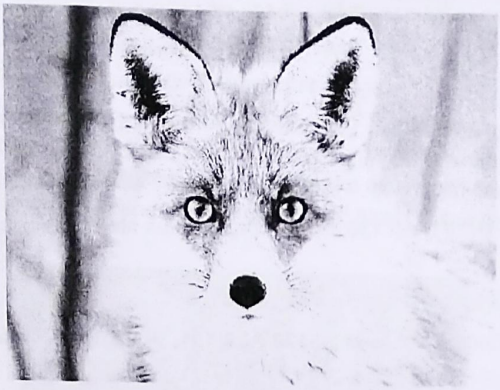
Pour augmenter (ou diminuer) la luminosité d'une image, il suffit d'augmenter (ou diminuer) la valeur de tous les pixels de l'image, en ajoutant un nombre entier ***v*** positif (ou négatif) à chaque pixel. Attention toutefois à ce que les valeurs des pixels $M_{i,j}$ restent bien comprises entre **0** et **255** :

- Si $M_{i,j} + v > 255$ alors, $M_{i,j} \leftarrow 255$;
- Si $M_{i,j} + v < 0$ alors, $M_{i,j} \leftarrow 0$.

Q.3- Écrire la fonction ***luminosite (M, v)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris, et un entier **v** (positif ou négatif). La fonction ajoute le nombre **v** à chaque pixel de l'image ***M***.

Exemple :

Après l'appel de la fonction **luminosite (M, 70)**, on obtient la matrice qui représente l'image suivante :



4- Seuillage :

Le *seuillage* est un traitement qui permet de sélectionner les informations significatives dans une image en niveaux de gris. Ce traitement nécessite le réglage d'un seuil ***S*** : Si la valeur d'un pixel $M_{i,j}$ est inférieure au seuil ***S***, alors la valeur de ce pixel est remplacée par **0**, sinon, la valeur de ce pixel est remplacée par **255**. Ainsi, on obtient une image en noir et blanc, sans niveaux de gris.

Exemple :



Q.4.a- Écrire la fonction ***moyenne (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris. La fonction retourne la valeur de la moyenne des valeurs des pixels de ***M*** : la somme des valeurs de tous les pixels dans l'image, divisée par le compte des pixels dans l'image.

Exemple : La fonction **moyenne (M)** retourne la valeur **225.69**

Q.4.b- Écrire la fonction ***seuillage (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris. Dans l'image ***M***, la fonction effectue le traitement du seuillage, en utilisant pour seuil : la moyenne de ***M***.

5- Histogramme d'une image :

L'**histogramme** représente la distribution des intensités de l'image. C'est un outil fondamental du traitement d'images, avec de très nombreuses applications. Les histogrammes sont aussi très utilisés en photographie et pour la retouche d'images.

Dans une image en niveaux de gris, les valeurs des pixels sont comprises entre **0** et **255**. L'histogramme de cette image associe à chaque niveau de gris ***k*** le nombre de pixels ayant la même valeur **k**.

Q.5.a- Écrire la fonction ***occurrences (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris. La fonction retourne une liste ***C*** de taille **256** : pour chaque niveau de gris ***k***, l'élément ***C[k]*** contient le compte des pixels de l'image ***M***, ayant la même valeur ***k***.

Exemple :

La fonction **occurrences (M)** retourne la liste **C** de taille **256**, suivante :

C = [0, 1, 0, 1, 0, 1, 2, 3, 4, 6, 5, 7, 4, 8, 9, 15, 10, 14, 16, 17, 25, 22, 28, 29, 33, 33, 42, 33, 46, 60, 53, 49, 52, 58, 80, 76, 75, 84, 67, 98, 87, 74, 72, 89, 89, 97, 78, 92, 93, ..., 115, 83, 61, 53, 44, 56, 48, 141, 82 **]**

Ainsi, à partir de cette liste, on déduit que :

- **C[0]** contient le nombre **0**, cela signifie qu'il y a **0** pixel ayant la couleur noire ;
- **C[1]** contient le nombre **1**, cela signifie qu'il y a **1** pixel ayant le niveau de gris **1** ;
- **C[7]** contient le nombre **3**, cela signifie qu'il y a **3** pixel ayant le niveau de gris **7** ;
- ....
- **C[255]** contient le nombre **82**, cela signifie qu'il y a **82** pixels ayant la couleur blanche.

Q.5.b- Déterminer la complexité de la fonction **occurrences (M)**. Justifier votre réponse.

Q.5.c- Écrire la fonction ***histogramme (C)*** qui reçoit en paramètre la liste ***C*** des occurrences d'une image en niveaux de gris. La fonction retourne une liste ***H*** de tuples, qui représente l'histogramme de l'image ***M*** : Chaque tuple de ***H*** est composé de deux éléments : le premier élément est la valeur d'un niveau de gris ***k*** qui existe dans l'image ***M***, le second élément est le compte des pixels de ***M*** ayant la même valeur ***k***.

Exemple :

La fonction **histogramme (C)** retourne la liste **H** suivante :

H = [(**1**, 1), (**3**, 1), (**5**, 1), (**6**, 2), (**7**, 3), (**8**, 4), (**9**, 6), (**10**, 5), (**11**, 7), (**12**, 4), (**13**, 8), (**14**, 9), (**15**, 15), (**16**, 10), (**17**, 14), (**18**, 16), (**19**, 17), (**20**, 25) , ..., (**250**, 53), (**251**, 44), (**252**, 56), (**253**, 48), (**254**, 141), (**255**, 82) **]**

6- Histogramme normalisé :

Les histogrammes sont en général normalisés : ***on divise l'occurrence de chaque niveau de gris, par le nombre total de pixels de l'image***. Les valeurs obtenues varient alors entre **0** et **1**, et peuvent être interpréter comme ***les probabilités d'occurrence de chaque niveau de gris*** dans l'image.

Q.6- Écrire la fonction ***normal (H)*** qui reçoit en paramètre la liste ***H*** qui représente l'histogramme d'une image en niveaux de gris. La fonction retourne une liste ***P*** de tuples, qui représente l'histogramme normalisé de l'image. Chaque tuple de ***P*** est composé de deux éléments : le premier élément est la valeur d'un niveau de gris ***k*** dans l'image, le second élément est l'occurrence de ***k*** divisée par le compte total de pixels dans l'image.

Exemple :

La fonction **normal (H)** retourne la liste **P** suivante :

P = [(**1**, 2.91*10^{-6}), (**3**, 2.91*10^{-6}), (**5**, 2.91*10^{-6}), (**6**, 5.82*10^{-6}), (**7**, 8.73*10^{-6}), (**8**, 1.16*10^{-5}), (**9**, 1.75*10^{-5}), (**10**, 1.45*10^{-5}), (**11**, 2.04*10^{-5}), (**12**, 1.16*10^{-5}), ..., (**253**, 1.4*10^{-4}), (**254**, 4.10*10^{-4}), (**255**, 2.39*10^{-4}) **]**

7- Entropie d'une image :

Le nombre de bits réellement nécessaires pour coder une image varie d'une image à une autre, en fonction de leur contenu informationnel. Ce nombre dépend de l'*entropie* ***E***, définie à partir de la distribution des niveaux de gris de l'image, selon la formule suivante :

$$E = \sum -p_k * log_2(p_k)$$

Avec :

- ***0 < pk < 1*** est la probabilité d'occurrence de chaque niveau de gris ***k*** dans l'image ;
- On suppose que la fonction prédéfinie ***log2*** est importée.

Cette grandeur représente le nombre moyen de bits par pixel nécessaires pour coder toute l'information présente dans l'image. Elle est utilisée dans les techniques de compression sans perte pour adapter le volume de donnée des images à leur contenu informationnel.

Q.7- Écrire la fonction ***entropie (P)*** qui reçoit en paramètre une liste ***P*** de tuples, qui représente l'histogramme normalisé d'une image en niveaux de gris. La fonction retourne la valeur de l'entropie de l'image.

Exemple :

La fonction **entropie (M)** retourne la valeur **7.14**

8- Histogramme cumulé d'une image :

À partir de l'histogramme normalisé, on définit également l'**histogramme cumulé** : à chaque niveau de gris ***k***, on fait correspondre la somme partielle des probabilités d'occurrence des niveaux de gris ***i*** tel que : ***i ≤ k***

$$k \rightarrow \sum_{i \leq k} p_i$$

Q.8- Écrire la fonction ***cumul (P)*** qui reçoit en paramètre une liste ***P*** de tuples, qui représente l'histogramme normalisé d'une image en niveaux de gris. La fonction retourne un dictionnaire ***D***, qui représente l'histogramme cumulé de l'image. Les clés du dictionnaire ***D*** sont les différents niveaux de gris présents dans l'image. Et, les valeurs du dictionnaire ***D*** sont les sommes partielles des probabilités d'occurrence correspondante à chaque niveau de gris.

Exemple :

La fonction **cumul (P)** retourne le dictionnaire **D** suivant :

{ 1: 2.91*10^{-6}, **3:** 5.82*10^{-6}, **5:** 8.73*10^{-6}, **6:** 1.45*10^{-5}, **7:** 2.33*10^{-5}, **8:** 3.49*10^{-5}, **9:** 5.24*10^{-5}, **10:** 6.69*10^{-5}, **11:** 8.73*10^{-5}, **12:** 9.89*10^{-5}, ... , **250:** 0.998920, **251:** 0.999048, **252:** 0.999211, **253:** 0.999351, **254:** 0.999761, **255:** 1.00 **}**

9- Égalisation d'histogramme :

En traitement d'images, l'**égalisation d'histogramme** est une méthode d'ajustement du contraste d'une image numérique, en utilisant l'histogramme.

L'égalisation d'histogramme permet de mieux répartir les intensités sur l'ensemble de la plage de valeurs possibles (0 à 255), en « étalant » l'histogramme. L'égalisation est intéressante pour les images dont la totalité, ou seulement une partie, est de faible contraste (l'ensemble des pixels sont d'intensité proches).

La méthode de l'égalisation d'histogramme consiste à appliquer une transformation indépendamment sur chaque pixel de l'image. Cette transformation est construite à partir de l'histogramme cumulé. ***La valeur de chaque pixel $M_{i,j}$ est remplacée par le produit suivant*** :

$$255 * \sum_{q \leq k} p_q$$

k étant la valeur du pixel $M_{i,j}$.

Q.9- Écrire la fonction ***egalisation (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris. La fonction effectue la transformation de l'égalisation d'histogramme de l'image ***M***.

Exemple :





Image avant et après égalisation d'histogramme

********** FIN **********

Prepa Help

prepahelp.com · classes préparatoires, Maroc et France

Travailler les chapitres de ce sujet

Prepa Help suit le programme de PSI chapitre par chapitre : cours détaillé, méthodes, pièges, exercices et problèmes corrigés pas à pas, tirés des concours. L'essai gratuit donne accès aux chapitres offerts de ta filière, sans carte bancaire.

Chapitres liés à ce sujet

- Architecture de l'ordinateur et représentation des nombres
- Bases de Python : expressions, types, instructions, fonctions
- Complexité des algorithmes itératifs et récursifs

Ouvrir l'essai gratuit

Scanne le code ou clique sur le bouton.

prepahelp.com/#essai

Une question sur le concours ou sur ta filière : **écrire sur WhatsApp**

Les autres années et épreuves : prepahelp.com/annales/cnc/psi