Concours National Commun (CNC), Maroc

Session 2025, filière PSI

Informatique

Durée : 2 heures

Ce que traite le sujet

- Exercice
- Partie II : apprentissage automatique, prix des téléphones mobiles (première partie, numérotée II dans le sujet)
- Partie II : réduction d'image, algorithme seam carving (seconde partie, numérotée II elle aussi)

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/2025/info

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

المملكة المغربية
وزارة التعليم العالي
والبحث العلمي والابتكار

Royaume du Maroc
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique et de l'Innovation

جامعة الحسن الثاني بالدار البيضاء
UNIVERSITE HASSAN II DE CASABLANCA

ENSEM

CONCOURS NATIONAL COMMUN

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

Session 2025

Épreuve d'Informatique

Filières : **PSI**

Durée : **2 heures**

Cette épreuve comporte **11 pages** au format A4, en plus de cette page de garde.

IMPORTANT
L'usage de la calculatrice et de tout système électronique ou informatique est interdit durant cette épreuve.

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 deux 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)

Q.1- Écrire la fonction ***somme (k)*** qui reçoit en paramètre un entier ***k*** positif, et qui retourne la valeur de la somme : ***1 + 2 + 3 + ... + k***.

Q.2- Donner la complexité de la fonction ***somme (k)***. Justifier votre réponse.

Q.3- Écrire la fonction ***som_puiss (x, n)*** qui reçoit en paramètres un réel ***x*** non nul et un entier positif ***n***. La fonction retourne la valeur de la somme :

$$x^0 + x^1 + x^2 + \ldots + x^n$$

Q.4- Écrire la fonction ***produit (L)*** qui reçoit en paramètre une liste de nombres ***L***, et qui retourne la valeur du produit des éléments de ***L***.

Q.5- Écrire la fonction ***pairs (L)*** qui reçoit en paramètre une liste d'entiers positifs ***L***, et qui retourne une nouvelle liste contenant les nombres pairs dans ***L***.

Exemple :

pairs ([15, 42, 7, 25, 42, 10]) retourne la liste ***[42, 42, 10]***

Q.6 - Écrire la fonction ***impairs (L)*** qui reçoit en paramètre une liste d'entiers positifs ***L***, et qui retourne le compte des nombres impairs dans ***L***.

Q.7- Écrire la fonction ***indices (x, L)*** qui reçoit en paramètres un nombre ***x*** et une liste de nombres ***L***. La fonction retourne une nouvelle liste qui contient les indices des occurrences de ***x*** dans ***L***.

Exemple :

indices (7, [15, 7, 42, 7, 7, 25, 7, 17]) retourne la liste ***[1, 3, 4, 6]***

Partie II :

Apprentissage automatique

Prix des téléphones mobiles

Une nouvelle entreprise de téléphonie mobile veut livrer un combat difficile aux grandes entreprises comme Apple, Samsung, etc.

Dans le but d'estimer le prix des mobiles qu'elle crée, cette nouvelle entreprise a collecté des données de diverses entreprises de téléphonie mobile. Cet ensemble de données permet d'identifier les facteurs sous-jacents qui affectent les prix des téléphones mobiles. En analysant les relations entre ces fonctionnalités et leurs gammes de prix respectives, on peut obtenir des informations précieuses sur la dynamique du marché des téléphones mobiles

Les données collectées pour chaque téléphone mobile sont composées de **15** variables (caractéristiques), dont les **14** premières sont des variables prédictives. La dernière est la variable cible :

1. ***puiss_batterie*** : Puissance de la batterie
2. ***vitesse_micropr*** : Vitesse du microprocesseur
3. ***cam_fr*** : Capacité de la caméra frontale
4. ***mem_int*** : Capacité de la mémoire interne en giga-octet
5. ***profondeur*** : Profondeur du téléphone mobile
6. ***poids*** : Poids du téléphone mobile
7. ***cœurs*** : Nombre de cœurs d'un processeur
8. ***cam_pr*** : Capacité de l'appareil photo principal
9. ***resolution_htr*** : Hauteur de résolution des pixels
10. ***resolution_lrg*** : Largeur de résolution des pixels
11. ***mémoire*** : Capacité de la mémoire vive en giga-octet
12. ***ecran_htr*** : Hauteur de l'écran du téléphone mobile
13. ***ecran_lrg*** : Largeur de l'écran du téléphone mobile
14. ***max_charge*** : La durée la plus longue d'une seule charge de batterie
15. ***prix*** : Le prix du mobile (variable cible)

Base de données et langage SQL :

Les données collectées par la nouvelle entreprise sont rangées dans une base de données composée de deux tables **'Entreprise'** et **'mobile'** dont les structures sont les suivantes :

- **Entreprise (code** (texte) **, nom** (texte), **pays** (texte) **)** ;
- L'attribut **code** est la clé primaire de la table **'Entreprises'** ;
- **Mobiles (codEntrp** (texte), **puiss_batterie** (entier), **vitesse_micropr** (réel), **cam_fr** (entier), **mem_int** (entier), **profondeur** (réel), **poids** (entier), ***cœurs*** (entier), **cam_pr** (entier), **resolution_htr** (entier), **resolution_lrg** (entier), **mémoire** (entier), **ecran_htr** (entier), **ecran_lrg** (entier), **max_charge** (entier), **prix** (réel) **)**
- Dans la table **'Mobile'**, l'attribut **codEntrp** est une clé étrangère qui fait référence à la clé primaire de la table **'Entreprise'**.

Exemples :

Entreprise		
code	*nom*	*pays*
apl	Apple	USA
sms	Samsung	Corée sud
hw	Huawei	Chine
xm	Xiaomi	Chine
...	...	...

Mobiles															
codEntrp	puiss_batterie	vitesse_micropr	cam_fr	mem_int	profondeur	poids	coeures	cam_pr	resolution_htr	resolution_lrg	mémoire	ecran_htr	ecran_lrg	max_charge	prix
sms	842	2.2	1	7	0.6	188	2	2	20	756	2549	9	7	19	1400
hw	510	2	5	45	0.9	168	6	16	483	754	3919	19	4	2	3500
apl	563	0.5	2	41	0.9	145	5	6	1263	1716	2603	11	2	9	2300
xm	615	2.5	0	10	0.8	131	6	9	1216	1768	2769	16	8	11	2000
xm	1821	1.2	13	44	0.6	141	2	14	1208	1212	1411	8	2	15	1200
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
apl	1021	0.5	0	53	0.7	136	3	6	905	1988	2631	17	3	7	2500

Q.8 – Écrire en algèbre relationnelle, la requête qui donne pour résultat les codes et les noms des entreprises du pays *'Chine'*.

Q.9 – Écrire en langage SQL, la requête qui donne pour résultat la capacité de la caméra frontale et la capacité de l'appareil photo principal des mobiles de l'entreprise de nom *'Honor'*, triés dans l'ordre décroissant de la capacité de l'appareil photo principal.

Q.10 – Écrire en langage SQL, la requête qui donne pour résultat la plus grande et la plus petite capacité de la mémoire vive des mobiles.

Q.11 – Écrire en langage SQL, la requête qui donne pour résultat les codes des entreprises, le compte des mobiles pour chaque entreprise. Le résultat doit être trié dans l'ordre décroissant des comptes des mobiles.

Q.12 – Écrire en langage SQL, la requête qui donne pour résultat les noms des entreprises sans répétition, qui fabriquent des mobiles dont la capacité de la mémoire vive est inférieure à la moyenne de la capacité la mémoire vive de tous les mobiles.

Q.13 – Écrire en langage SQL, la requête qui donne pour résultat les noms des entreprises telles que la moyenne des prix des mobiles de chaque entreprise est supérieure à *7500*. Le résultat doit être trié dans l'ordre décroissant de la moyenne des prix.

Dans la suite de cette partie, les données collectées sont rangées dans une liste ***M***, déclarée variable globale. Chaque élément ***Mi*** est une liste qui représente les 15 attributs d'un téléphone mobile.

```
M = [ [ 842, 2.2, 1, 7, 0.6, 188, 2, 2, 20, 756, 2549, 9, 7, 19, 1400 ],
      [ 510, 2, 5, 45, 0.9, 168, 6, 16, 483, 754, 3919, 19, 4, 2, 3500 ],
      [ 563, 0.5, 2, 41, 0.9, 145, 5, 6, 1263, 1716, 2603, 11, 2, 9, 2300 ],
      ...
      [ 1021, 0.5, 0, 53, 0.7, 136, 3, 6, 905, 1988, 2631, 17, 3, 7, 2500 ] ]
```

Dans chaque sous-liste ***Mi*** :

- L'indice ***0*** correspond à l'attribut ***puiss_batterie*** des téléphones mobiles. Cet attribut est représenté par les nombres : **842, 510, 563, 615, 1821, ...1021**
- L'indice ***1*** correspond à l'attribut ***vitesse_micropr*** des téléphones mobiles. Cet attribut est représenté par les nombres : **2.2, 2, 0.5, 2.5, 1.2, ..., 0.5**
- ...
- L'indice ***14*** correspond à l'attribut ***prix*** des téléphones mobiles. Cet attribut est représenté par les nombres : **1400, 3500, 2300, 2000, 1200, ..., 2500**

Dans la liste ***M***, on remarque que les attributs sont sur des échelles très différentes (par exemple ***puiss_batterie*** entre ***501*** et ***1998***, alors que ***vitesse_micropr*** entre ***0.5*** et ***3***). Les différences de l'attribut ***puiss_batterie*** contribuent alors beaucoup dans les calculs, ce qui ferait que cet attribut aurait un poids plus important pour la prédiction.

Il est donc extrêmement important de redimensionner les variables afin qu'elles aient une échelle comparable. L'une de façons courantes de redimensionner est la ***normalisation*** (*moyenne=0, écart type=1*).

Si un attribut x à une moyenne $\overline{x}$ et un écart type σ , on peut le normaliser en le remplaçant par : $\frac{x - \overline{x}}{\sigma}$

Q.14- Écrire la fonction ***moyenne (j)*** qui reçoit en paramètre un entier ***j*** qui représente l'indice d'un attribut dans la liste ***M***. La fonction retourne la moyenne de cet attribut.

$$moyenne\ (j) = \frac{1}{n} * \sum_{i=0}^{n-1} M_{ij} \qquad n\ est\ la\ taille\ de\ M$$

Exemple : ***moyenne (5)*** retourne le nombre **140.249**

Q.15- Écrire la fonction ***ecart_type (j)*** qui reçoit en paramètre un entier ***j*** qui représente l'indice d'un attribut dans la liste ***M***. La fonction retourne la valeur de l'écart type de cet attribut

$$ecart_type\ (j) = \sqrt{\frac{1}{n} * \sum_{i=0}^{n-1} (M_{i,j} - m)^2}$$

$n\ est\ la\ taille\ de\ M, et\ m\ est\ la\ moyenne\ de\ l'attribut\ d'indice\ j\ dans\ M.$

Exemple : ***ecart_type (0)*** retourne le nombre **439.3083377967573**

Q.16- Écrire la fonction ***normalisation ( )*** qui effectue la normalisation des données de tous les attributs de ***M*** sauf l'attribut cible (le dernier attribut), en remplaçant chaque élément $M_{i,j}$ par la valeur de l'expression suivante :

$$M_{i,j} \leftarrow \frac{M_{i,j} - moyenne(j)}{ecart_type(j)}$$

Exemple :

Après l'appel de la fonction ***normalisation ( )***, la liste des données ***M*** devient ainsi :

M = [[-0.9, 0.83, -0.76, -1.31, -1.38, 0.39, 0.34, 1.35, -1.1, -1.41, -1.15, -0.78, 0.28, 1.46, 1400],
[-1.66, 0.59, 0.16, 1., 0.71, 1.66, 1.38, 0.78, 0.65, -0.37, -1.15, 1.59, -0.41, -1.65, 3500],
[-1.54, -1.25, -0.53, -0.65, 0.49, 0.44, 1.38, 0.13, 0.21, 1.39, 1.07, -0.31, -0.86, -0.37, 2300],
...
[-0.5, -1.25, -0.99, -0.65, 1.16, 0.47, 0.69, -0.12, -0.66, 0.59, 1.7, 1.11, -0.64, -0.73, 2500]]

Algorithme KNN :

L'algorithme KNN est un algorithme de Machine Learning qui appartient à la classe des algorithmes d'apprentissage supervisé, et qui peut être utilisé pour résoudre les problèmes de régression.

L'algorithme KNN reçoit un ensemble de données qui est étiqueté avec des valeurs de sorties correspondantes sur lequel il va pouvoir s'entraîner et définir un modèle de prédiction. Cet algorithme pourra par la suite être utilisé sur de nouvelles données afin de prédire leurs valeurs de sorties correspondantes.

Pour la classification d'un nouveau téléphone mobile ***X***, le principe de l'algorithme ***KNN*** est le suivant :

- Pour mesurer la similarité, on définit une fonction qui calcule la distance entre le nouveau téléphone mobile ***X*** et un autre téléphone mobile de ***M*** ;
- On calcule toutes les distances entre chaque téléphones mobiles de ***M***, le téléphone mobile ***X*** ;
- On choisit un nombre entier strictement positif ***k*** ;
- On retourne la moyenne des ***prix*** des ***k*** téléphones mobiles de ***M*** les proches au nouveau téléphone mobile ***X***.

Q.17- Pour la mesure de similarité, on propose d'utiliser la distance de *Manhattan*.

La liste ***X*** représente un nouveau téléphone mobile, et la liste ***e*** représente un téléphone mobile de ***M***. La distance de *Manhattan* entre ***X*** et ***e*** est définie par la formule suivante :

$$distance\ (X, e) = \sum_{i=0}^{13} |X_i - e_i|$$

Écrire la fonction ***distance (X, e)*** qui reçoit en paramètres les deux téléphones mobiles ***X*** et ***e*** et qui retourne la distance euclidienne entre ***X*** et ***e***.

Exemple :

X = [0.87, 0.46, -0.53, -0.15, -0.61, -0.45, -0.35, -0.06, -1.54, -1.35, -1.6, -1.26, -0.86, 0.36]

e = [-0.9, 0.83, -0.76, -1.31, -1.38, 0.39, 0.34, 1.35, -1.1, -1.41, -1.15, -0.78, 0.28, 1.46, 1400]

distance (X, e) retourne **10.91**

Q.18- Écrire la fonction ***indices ( )*** qui retourne la liste ***D*** des indices des éléments de ***M***.

Exemple :

indices ( ) retourne la liste *D* = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, ...]

Q.19- Écrire la fonction ***tri_indices (D, X)*** qui reçoit en paramètres la liste ***D*** des indices des éléments de ***M***, la liste ***X*** qui représente un nouveau téléphone mobile. La fonction trie les éléments ***Di*** de ***D*** dans l'ordre croissant de la distance de *Manhattan* entre ***X*** et chaque élément dans ***M*** d'indice ***Di***. (Ne pas utiliser la fonction *sorted()* et la méthode *sort()*)

Exemples :

X = **[0.87, 0.46, -0.53, -0.15, -0.61, -0.45, -0.35, -0.06, -1.54, -1.35, -1.6, -1.26, -0.86, 0.36]**

Après l'appel de la fonction ***tri_indices (D, X)***, on obtient la liste des indices suivante :

[55, 14, 49, 72, 17, 9, 178, 21, 141, 42, 11, 50, 177, 113, 19, 156, 39, 10, 16, 88, 332, 38, ...]

- Le téléphone mobile M_{55} est le premier plus proche téléphone mobile à ***X*** ;
- Le téléphone mobile M_{14} est le deuxième plus proche téléphone mobile à ***X*** ;
- Le téléphone mobile M_{49} est le troisième plus proche téléphone mobile à ***X*** ;
- ...

Q.20- Écrire la fonction ***knn (k, X)*** qui reçoit en paramètres un entier strictement positif ***k*** et un nouveau téléphone mobile ***X***. En utilisant l'algorithme ***KNN***, la fonction retourne le prix prédit pour le nouveau téléphone mobile ***X***.

Exemple :

X = **[0.87, 0.46, -0.53, -0.15, -0.61, -0.45, -0.35, -0.06, -1.54, -1.35, -1.6, -1.26, -0.86, 0.36]**

knn (10, X) retourne ***2705.3***

Q.21- On suppose que la liste ***T*** représente les téléphones mobiles crées par la nouvelle entreprise : Chaque élément ***Ti*** est une liste qui représente les **14** attributs d'un nouveau téléphone mobile.

Exemple :

T = [[1.33, 0.22, -0.07, -1.22, 1.03, -0.04, 1.52, 0.01, -0.6, -0.54, 1.01, 0.16, 0.51, 1.28],
[1.63, -1.25, -0.99, -0.44, 1.03, 1.32, -0.23, -1.64, -0.3, -0.24, -1.31, 0.88, -0.64, -1.1],
[0.47, -1.25, -0.99, 1.16, 0.69, 0.95, 1.08, 0.67, -0.58, -0.96, -0.95, 1.11, -1.09, 1.65],
[-1.66, -1.13, -0.53, -1.27, -1.39, -1.34, 0.21, 0.84, 1.11, -0.06, -1.49, 1.59, 0.97, 0.18],
...]

Écrire la fonction ***estimation_prix (k, T)*** qui reçoit en paramètres un entier strictement positif ***k***, et la liste des téléphones mobiles ***T***. En utilisant l'algorithme ***KNN***, la fonction retourne une nouvelle liste ***P*** telle que : chaque élément ***Pi*** est le prix prédit pour le téléphone ***Ti***.

Partie II :

Réduction d'image
Algorithme 'Seam Carving'

Dans cette partie, on suppose que le module ***numpy*** est importé : `import numpy as np`

Dans cette partie, on manipule des matrices de nombres entiers positifs.

Exemple :

```
M = [ [238   83   20  120   25  240   46    5   46  153]
      [238  152  110  251  148   20  246  144  251  203]
      [ 96  197  169   12   47  166   37  229  115   57]
      [216    9   49   12  152  206  195  164   34  108]
      [ 85   44    4   94  153  225  195  198   18  121]
      [ 14  159   96  152   30  204  182  146  244  238]
      [ 40   92   67  167    9  125  127  154  152   72]
      [ 32  137  191  143   33  113   72  187  173   36] ]
```

Pour plus de clarté, dans la suite de cette partie, on utilisera cette matrice dans les exemples des fonctions de la section (A) ci-dessous.

A- *Chemin vertical dans une matrice*

Dans une matrice ***M***, un chemin vertical est un chemin du haut vers le bas, qui part d'une case de la première ligne de ***M*** jusqu'à une case de la dernière ligne de ***M***. Depuis une case $M_{i,j}$ (qui ne se trouve pas sur la dernière ligne), les seuls déplacements autorisés sont :

- Si $M_{i,j}$ est sur la première colonne, on peut se déplacer vers $M_{i+1,j}$ ou bien vers $M_{i+1,j+1}$.
- Sinon si $M_{i,j}$ est sur la dernière colonne, on peut se déplacer vers $M_{i+1,j-1}$ ou bien vers $M_{i+1,j}$.
- Sinon, on peut se déplacer vers $M_{i+1,j-1}$ *ou bien* vers $M_{i+1,j}$ ou bien vers $M_{i+1,j+1}$.

Pour représenter un chemin vertical, on utilise une liste **C** telle que :

- La taille de ***C*** est égale au nombre de lignes de ***M*** ;
- Les éléments de ***C*** sont des indices de colonnes dans ***M***.

Par exemple, on considère le chemin vertical qui part de la case $M_{0,2}$, et qui passe par les cases suivantes :

```
[ [238   83  [20] 120   25  240   46    5   46  153]
  [238  152  110 [251] 148   20  246  144  251  203]
  [ 96  197  169  [12]  47  166   37  229  115   57]
  [216    9  [49]  12  152  206  195  164   34  108]
  [ 85 [44]    4   94  153  225  195  198   18  121]
  [ 14  159 [96]  152   30  204  182  146  244  238]
  [ 40   92   67 [167]   9  125  127  154  152   72]
  [ 32  137  191  143 [33] 113   72  187  173   36] ]
```

Le chemin $M_{0,2}$ (=20) → $M_{1,3}$ (=251) → $M_{2,3}$ (=12) → $M_{3,2}$ (=49) → $M_{4,1}$ → $M_{5,2}$ → $M_{6,3}$ → $M_{7,4}$ (=33) est représenté par la liste C = [2, 3, 3, 2, 1, 2, 3, 4]

La somme des valeurs des cases par les quelles passe un chemin est appelée : ***coût*** du chemin.

Par exemple, le coût du chemin ci-dessus est : ***20 + 251 + 12 + 49 + 44 + 96 + 167 + 33 = 672***

Q.22- Écrire la fonction ***cout_chemin (M, C)*** qui reçoit en paramètres une matrice ***M*** de nombres entiers, et une liste C qui représente un chemin vertical dans ***M***. La fonction le coût du chemin ***C***.

Exemple :

cout_chemin (M, [2, 3, 3, 2, 1, 2, 3, 4]) retourne ***672*** = $M_{0,2} + M_{1,3} + M_{2,3} + M_{3,2} + M_{4,1} + M_{5,2} + M_{6,3} + M_{7,4}$

Ils existent plusieurs chemins verticaux partent d'une cases $M_{0,j}$. Dans le but de trouver le chemin vertical de coût minimal, la méthode ***'glouton'*** consiste à se déplacer, à chaque embranchement, vers la case de plus petite valeur.

Q.23- Écrire la fonction ***glouton (M, j)*** qui reçoit en paramètres une matrice ***M*** de nombres entiers, et un entier ***j*** qui représente l'indice d'une colonne dans ***M***. En utilisant la méthode ***'glouton'***, la fonction retourne une liste qui représente le chemin vertical de plus petit coût dans ***M***, et qui part de la case $M_{0,j}$.

Exemple :

glouton (M, 2) retourne le chemin ***[2, 2, 3, 3, 2, 2, 2, 1]***, *son coût est 458*

Q.24- Est-ce que la méthode 'glouton' permet de trouver le chemin vertical de plus petit coût ?

Justifier votre réponse.

Dans toute la matrice ***M***, la recherche du chemin vertical de coût minimal est un problème qu'on peut résoudre en utilisant la méthode ***'Bottom-Up'*** de la programmation dynamique, dont le principe est le suivant :

On commence par créer une matrice ***R*** de même dimension que ***M*** : **Chaque élément $R_{i,j}$ représente le coût du chemin vertical minimal qui part de la case $M_{i,j}$.** Les éléments de ***R*** sont calculés du bas vers le haut, à l'aide de l'algorithme suivant :

- Si l'élément $R_{i,j}$ se trouve sur la dernière ligne de **R**, alors $R_{i,j} \leftarrow M_{i,j}$
- Sinon si l'élément $R_{i,j}$ se trouve sur la première colonne de **R**, alors $R_{i,j} \leftarrow M_{i,j} + min\ (R_{i+1,j}\ ,\ R_{i+1,j+1})$
- Si l'élément $R_{i,j}$ se trouve sur la dernière colonne de **R**, alors $R_{i,j} \leftarrow M_{i,j} + min\ (R_{i+1,j-1}\ ,\ R_{i+1,j})$
- Si l'élément $R_{i,j}$ ne se trouve ni sur la première colonne, ni sur la dernière colonne de **R**, alors $R_{i,j} \leftarrow M_{i,j} + min\ (R_{i+1,j-1}\ ,\ R_{i+1,j}\ ,\ R_{i+1,j+1})$

Q.25- Écrire la fonction ***couts (M)*** qui reçoit en paramètre une matrice ***M*** de nombres entiers. En utilisant l'algorithme ci-dessus, la fonction retourne la matrice ***R***.

Exemple :

couts (M) retourne la matrice ***R*** suivante :

[[625	383	320	420	270	485	291	617	658	799]
[473	387	300	441	338	245	714	612	694	646]
[235	336	308	190	225	484	468	615	501	443]
[346	139	179	178	318	431	492	516	386	460]
[171	130	198	166	225	297	441	532	352	467]
[86	231	220	194	72	246	340	334	352	346]
[72	124	204	200	42	158	199	226	188	108]
[32	137	191	143	33	113	72	187	173	36]]

Q.26- Écrire la fonction ***chemin_min (M)*** qui reçoit en paramètre une matrice ***M*** de nombres entiers. La fonction retourne la liste qui représente le chemin vertical de coût minimal dans la matrice ***M***.

Exemple :

[[625	383	320	420	**270**	485	291	617	658	799]
[473	387	300	441	338	**245**	714	612	694	646]
[235	336	308	190	**225**	484	468	615	501	443]
[346	139	179	**178**	318	431	492	516	386	460]
[171	130	198	**166**	225	297	441	532	352	467]
[86	231	220	194	**72**	246	340	334	352	346]
[72	124	204	200	**42**	158	199	226	188	108]
[32	137	191	143	**33**	113	72	187	173	36]]

chemin_min (M) retourne le vecteur ***[4, 5, 4, 3, 3, 4, 4, 4]***.

(son coût est : $M_{0,4} + M_{1,5} + M_{2,4} + M_{3,3} + M_{4,3} + M_{5,4} + M_{6,4} + M_{7,4} = 25+20+47+12+94+30+9+33 = 270$)

Q.27- Écrire la fonction ***supprime_chemin (M, C)*** qui reçoit en paramètre une matrice ***M*** de nombres entiers, et ***C*** un vecteur qui représente un chemin vertical dans ***M***. La fonction supprime les éléments de ***M*** correspondants au chemin ***C***.

Exemple :

Après l'appel de la fonction ***supprime_chemin (M, [4, 5, 4, 3, 3, 4, 4, 4])***, la matrice ***M*** devient ainsi :

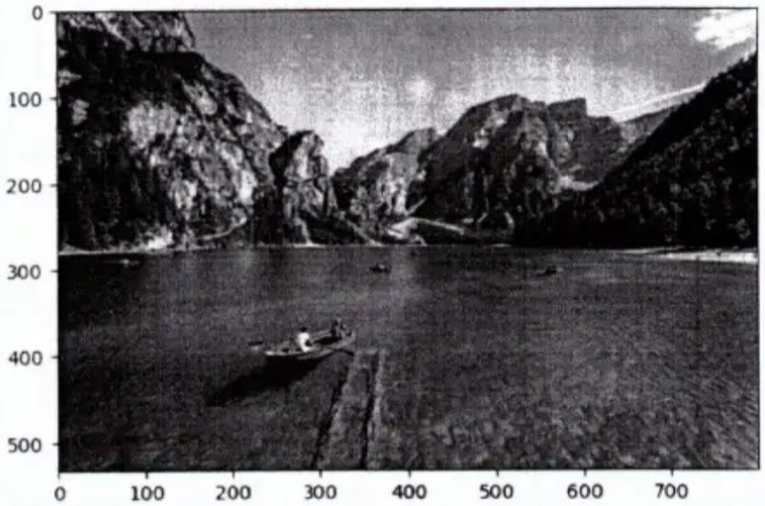
[[238	83	20	120	240	46	5	46	153]
[238	152	110	251	148	246	144	251	203]
[96	197	169	12	166	37	229	115	57]
[216	9	49	152	206	195	164	34	108]
[85	44	4	153	225	195	198	18	121]
[14	159	96	152	204	182	146	244	238]
[40	92	67	167	125	127	154	152	72]
[32	137	191	143	113	72	187	173	36]]

B- *Réduction d'une image par la méthode 'Seam Carving'*

Une image numérique en **niveaux de gris** est représentée par une matrice à deux dimensions. Chaque élément de la matrice est appelé **pixel**. La valeur de chaque pixel est un entier compris entre 0 et 255, ainsi l'image est codée sur **256** niveaux de gris :

- Le nombre **0** correspond au noir, et 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.

Exemple :



L'image en niveaux de gris ci-dessus, est représentée par la matrice ***M*** suivante *(533 lignes, 800 colonnes)* :

```
[ [129  139  77  100  127  78  86  105  98  103  ... ]
  [ 78   78  98  106  127  105  44  60  93  84  ... ]
  [ 60   66  125  129  142  136  57  51  58  74  ... ]
  [ 84  110  129  141  141  132  112  77  111  122  ... ]
  [ 64  100  100  124  130  136  165  129  60  117  ... ]
  ...
]
```

Pour plus de clarté, dans la suite de cette partie, on utilisera cette image dans les exemples des fonctions.

Le redimensionnement est une transformation applicable à une image numérique qui consiste à en modifier la taille, que ce soit pour l'agrandir ou pour la rétrécir, comme le ferait un zoom.

Le **seam carving**, ou recadrage intelligent, est un algorithme de réduction d'image développé par Shai Avidan et Ariel Shamir en 2007. Cet algorithme réduit une image par la suppression de chemins de pixels dits de moindre énergie, ce qui permet de moins la déformer.

L'énergie d'un pixel est en général mesurée par son contraste comparé à ses plus proches voisins, mais d'autres techniques, comme la détection de forme, peuvent être utilisées.

Calcul du gradient de l'image sur chaque pixel :

On définit une matrice ***G*** de même dimension que ***M***. Chaque élément $G_{i,j}$ correspond à l'intensité du gradient de l'image sur le pixel $M_{i,j}$: plus il est grand, plus il y a de forte variation de couleur au niveau de ce pixel.

Pour calculer les éléments de la matrice des gradients ***G***, on utilise la formule (à droite) correspondante à chaque élément de ***G*** (à gauche) :

$G =$

1	5	...	5	5	2
7	9	9	...	9	8
⋮	9	...	9	⋮	8
7	⋮	9	...	9	⋮
7	9	...	9	9	8
3	6	6	...	6	4

1. $G_{i,j} = |M_{i,j} - M_{i+1,j}| + |M_{i,j} - M_{i,j+1}|$
2. $G_{i,j} = |M_{i,j} - M_{i+1,j}| + |M_{i,j} - M_{i,j-1}|$
3. $G_{i,j} = |M_{i,j} - M_{i-1,j}| + |M_{i,j} - M_{i,j+1}|$
4. $G_{i,j} = |M_{i,j} - M_{i,j-1}| + |M_{i,j} - M_{i-1,j}|$
5. $G_{i,j} = |M_{i,j} - M_{i+1,j}| + |M_{i,j+1} - M_{i,j-1}|$
6. $G_{i,j} = |M_{i,j} - M_{i-1,j}| + |M_{i,j+1} - M_{i,j-1}|$
7. $G_{i,j} = |M_{i,j} - M_{i,j+1}| + |M_{i-1,j} - M_{i+1,j}|$
8. $G_{i,j} = |M_{i,j} - M_{i,j-1}| + |M_{i-1,j} - M_{i+1,j}|$
9. $G_{i,j} = |M_{i+1,j} - M_{i-1,j}| + |M_{i,j+1} - M_{i,j-1}|$

Le numéro de chaque élément $G_{i,j}$ *(à gauche) correspond au numéro de la formule (à droite) permettant de calculer la valeur de cet élément* $G_{i,j}$.

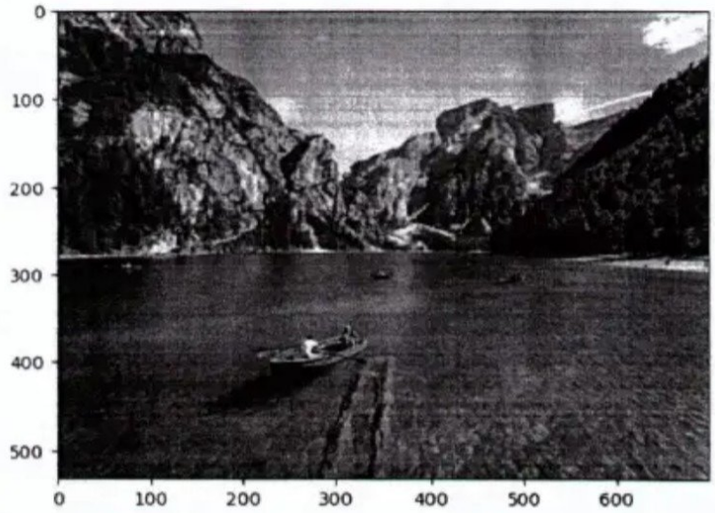
Q.28- Écrire la fonction ***gradients (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris. La fonction retourne la matrice des gradients ***G*** correspondante à ***M***.

Pour réduire une image en niveau de gris ***M****, la méthode* ***'Seam Carving'*** *consiste à :*

- ***a.*** *Construire la matrice des gradients* ***G*** *correspondante à* ***M*** *;*
- ***b.*** *Trouver le chemin vertical* ***C*** *de plus petit coût dans la matrice* ***G*** *;*
- ***c.*** *Supprimer le chemin* ***C*** *dans l'image* ***M****.*

Q.29- Écrire la fonction ***reduction (M, n)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en niveaux de gris, et ***n*** un entier strictement positif. En utilisant l'algorithme Seam Carving, la fonction supprime dans l'image ***M*** les pixels de ***n*** chemins verticaux de plus petits coûts dans la matrice des gradients correspondante à ***M***.

Exemple : L'image suivante est obtenue après la suppression de ***100*** chemins verticaux.



########## **FIN** ##########

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

- Bases de Python : expressions, types, instructions, fonctions
- Complexité des algorithmes itératifs et récursifs
- Tri par fusion et tri rapide

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