

## Devoir surveillé n°1

Durée : 2 heures, calculatrices et documents interdits

# Premier problème : une base de donnée

Un professeur d'informatique a créé une base de données pour gérer les notes de ses interrogations hebdomadaires. Pour cela, il a créé trois tables : **etudiants**, **interros**, **notes**.

**etudiants** : Contient une liste des données des étudiants. Cette table possède 4 attributs :

**id** : Identifiant de l'étudiant dans la base de données, de type **INTEGER**, formant une clé primaire.

**nom** : Nom de l'étudiant, de type **VARCHAR**.

**prenom** : Prénom de l'étudiant, de type **VARCHAR**.

**date\_naissance** : Date de naissance de l'étudiant, de type **VARCHAR**, au format **aaaa-mm-jj**.

**interros** : Contient la liste des données des interrogations. Cette table contient 4 attributs :

**id** : Identifiant de l'interrogation, de type **INTEGER**, formant une clé primaire.

**titre** : Titre de l'interrogation, de type **VARCHAR**.

**sujet** : Sujet de l'interrogation, de type **VARCHAR**.

**date** : Date du jour où a été donnée l'interrogation, de type **VARCHAR**, au format **aaaa-mm-jj**.

**notes** : Contient les notes des étudiants aux interrogations. Cette table contient 3 attributs :

**id\_etudiant** : Identifiant de l'étudiant, de type **INTEGER**.

**id\_interro** : Identifiant de l'interrogation, de type **INTEGER**.

**note** : Note obtenue, de type **INTEGER**.

- 1) Quelles sont les clés étrangères de la table **notes** ?
- 2) Donner une clé primaire de la table **notes**.
- 3) Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants de la classe.
- 4) Écrire une requête SQL permettant d'obtenir la liste des prénoms des étudiants de la classe, sans doublon.
- 5) Les enfants du professeur se sont amusés à rentrer des données factices, que le professeur aimerait retrouver afin de les effacer ensuite. Écrire une requête SQL permettant d'obtenir la liste des identifiants des étudiants ayant pour nom "**reinedesneiges**".
- 6) Écrire une requête SQL permettant d'obtenir la date de naissance de l'étudiant le plus jeune de la classe.
- 7) Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants ayant obtenu au moins un 20 à une des interrogations.
- 8) Écrire une requête SQL permettant d'obtenir la liste des noms, prénoms d'étudiants et titres d'interrogations pour chaque note de 0 obtenue.
- 9) Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants, avec la moyenne des notes de chaque étudiant.
- 10) Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants, suivis pour chaque étudiant du nombre d'interrogations rendues.

- 11) Écrire une requête SQL permettant d'obtenir le nom, le prénom et le nombre de copies rendues par l'étudiant ayant rendu le plus de copies (on suppose qu'il n'y en a qu'un).

On rappelle que l'instruction `LIMIT k` permet de tronquer une table à ses `k` premières lignes.

- 12) Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants ayant rendu au moins 10 copies, suivis du nombre de copies rendues.
- 13) Écrire une requête SQL permettant d'obtenir la liste des titres des interrogations, suivie pour chaque interrogation du nombre d'étudiants qui n'ont pas rendu de copies.
- 14) Écrire une requête SQL permettant d'obtenir la liste des titres des interrogations pour lesquelles tous les étudiants ont rendu une copie.


## Second problème : traitement d'image



Image A

Les ordinateurs et de nombreux dispositifs électroniques (caméras numériques, écrans, etc.) représentent les images comme des matrices de nombres entiers. Dans ce problème, on se limite aux images rectangulaires en teintes de gris.

En machine, une telle image est la donnée d'une *hauteur*  $h$ , d'une *largeur*  $l$ , et d'une matrice  $M$  d'entiers de  $h$  lignes et  $l$  colonnes. L'image est divisée en éléments ou *pixels* définis par leurs numéros de ligne  $i$  et de colonne  $j$ . Chaque entier de la matrice définit le *ton de gris* associé au pixel de coordonnées  $(i, j)$ . Ce ton varie entre zéro, qui rend l'absence de lumière et donc le noir, et une valeur maximale dite *profondeur*  $p$  qui rend le blanc. Les valeurs intermédiaires rendent diverses teintes de gris de plus en plus claires. On notera que le pixel de coordonnées  $(0, 0)$  est conventionnellement situé en haut et à gauche de l'image.

Par exemple, pour cette image  une échelle de 16 teintes de gris allant du noir au blanc, on a  $h = 1$ ,  $l = 16$ ,  $p = 15$  et  $M$  ne possède qu'une seule ligne qui contient 16 entiers de 0 à 15 en ordre croissant.

On suppose définie une classe `Image` permettant de représenter en `Python` des images en teintes de gris. On crée une image par l'appel de la primitive `Image(h, l, p)`. On accède aux composants d'une image `img` par les attributs `img.h` (hauteur), `img.l` (largeur), `img.p` (profondeur) et `img.m` (matrice). On accède aux éléments de la matrice par la notation `img.m[i, j]` étant entendu que les indices commencent à zéro (un indice de ligne est donc un entier  $i$  compris entre 0 et  $h - 1$  au sens large, tandis qu'un indice de colonne est un entier  $j$  compris entre 0 et  $l - 1$  au sens large).

### Partie I. Opérations élémentaires

- 1) Écrire une fonction `inverser(img)` qui renvoie l'image inverse de l'image `img`, c'est-à-dire que le ton d'un pixel de la nouvelle image est  $p - v$  où  $v$  est le ton du pixel correspondant de l'image d'origine. Par exemple, l'image B de la figure 1 résulte de l'application de `inverser` à l'image A de l'introduction.
- 2) Écrire une fonction `flipH(img)` qui renvoie la transformée de l'image `img` par la symétrie d'axe vertical passant par le milieu de l'image. Par exemple, l'image C de la figure 1 résulte de l'application de `flipH` à l'image A de l'introduction.
- 3) Écrire une fonction `poserv(img1, img2)` qui prend en arguments deux images `img1` et `img2` de même largeur et profondeur, et qui renvoie la nouvelle image obtenue en posant `img1` sur `img2`. Par exemple, l'image D de la figure 1 résulte de l'application de `poserv` aux images B et C.

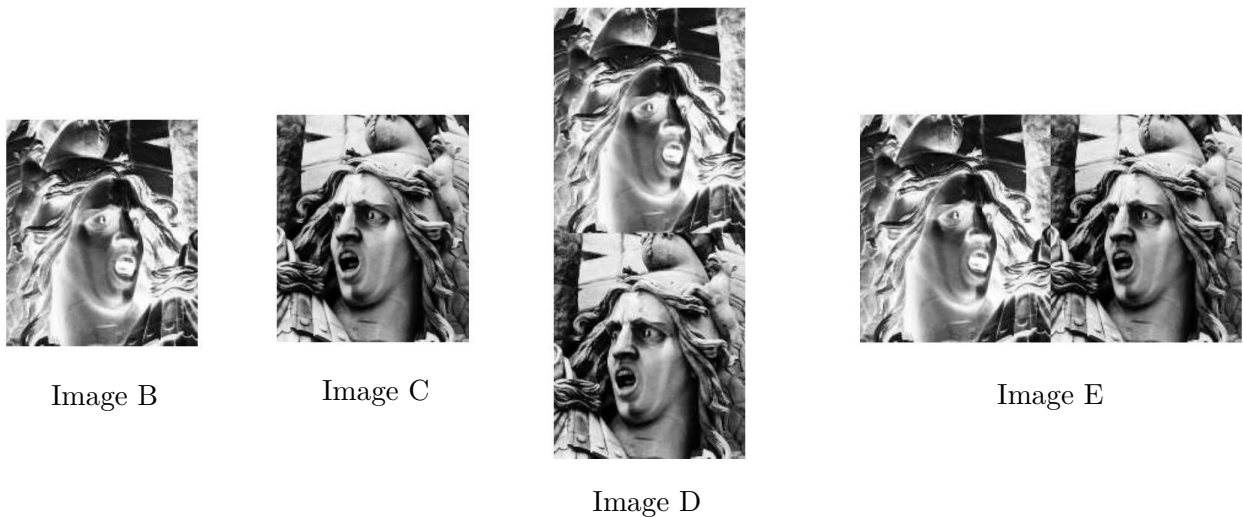


FIGURE 1 – Opérations élémentaires

- 4) Écrire une fonction `poserH (img1,img2)` qui prend en arguments deux images `img1` et `img2` de même hauteur et profondeur, et qui renvoie la nouvelle image obtenue en posant `img2` à droite de `img1`. Par exemple, l'image E de la figure 1 résulte de l'application de `poserH` aux images B et C.

## Partie II. Transferts

Certaines transformations des images sont simplement l'application d'une fonction aux tons, dont on rappelle qu'ils sont des entiers compris entre 0 et  $p$  (profondeur de l'image) au sens large.

Une telle fonction de transfert peut s'appliquer vers des images dont la profondeur n'est pas nécessairement  $p$ , mais une nouvelle profondeur  $q$ .

Une fonction de transfert est représentée par la donnée de la profondeur cible  $q$  et d'un tableau d'entiers  $t$  de taille  $p + 1$ , dont les cases contiennent des entiers entre 0 et  $q$  au sens large.

- 5) Écrire une fonction `transférer(img,q,t)` qui prend en arguments une image `img`, ainsi qu'une fonction de transfert donnée par un entier  $q$  et un tableau d'entiers  $t$ . La fonction `transférer` renvoie une nouvelle image, de même taille que `img`, de nouvelle profondeur  $q$  et dont chaque pixel  $(i,j)$  a pour ton  $t[img.m[i,j]]$ .
- 6) Écrire une nouvelle fonction `inverser` (cf. question 1)) qui utilise la fonction `transférer` de la question précédente.

L'**histogramme** d'une image de profondeur  $p$  est un tableau `h_i` d'entiers de taille  $p + 1$  tel que `h_i[v]` compte le nombre de pixels de l'image dont le ton est  $v$ .

- 7) Écrire une fonction `histogramme (img)` qui prend en argument une image `img` et retourne l'histogramme de celle-ci.

Soit `img` une image de hauteur  $h$ , de largeur  $l$  et de profondeur  $p$ . Soit  $h_i$  son histogramme et soit  $v_{\min}$  le ton de gris le plus sombre se trouvant dans l'image `img`. On égalise les tons en transformant chaque ton  $v$  en  $v'$  défini ainsi :

$$v' = p \times \frac{\left( \sum_{k=0}^v h_i[k] \right) - h_i[v_{\min}]}{h \times l - h_i[v_{\min}]} \quad \text{pour } v_{\min} \leq v \leq p$$

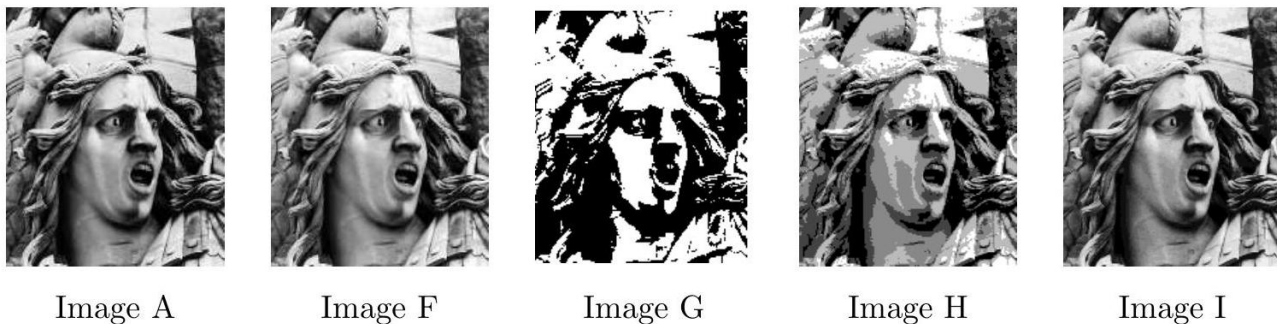


FIGURE 2 – Transferts

On note que  $v'$  n'est pas défini pour  $v$  tel que  $0 \leq v < v_{\min}$ , ce qui n'a pas d'importance, ces tons  $v$  étant absents de l'image. On note surtout que la valeur de  $v'$  ci-dessus n'est généralement pas un entier. On rappelle à cette occasion que la commande `int(round(x,0))` renvoie l'entier le plus proche de  $x$ .

- 8) Écrire la fonction `egaliser (img)` qui prend une image `img` en argument et qui renvoie une nouvelle image qui est `img` dont les tons de gris sont égalisés. Par exemple, l'image F de la figure 2 résulte de l'application de `egaliser` à l'image A.
- 9) Que renvoie la fonction `egaliser` appliquée à une image uniformément blanche ?

On cherche maintenant à réduire la profondeur d'une image de  $p$  vers  $q$  avec  $q \leq p$ . Une technique consiste à remplacer un ton  $v$  par un ton  $v'$  tel que  $\frac{v'}{q}$  est le plus proche possible de  $\frac{v}{p}$ .

- 10) Écrire une fonction `reduire (img,q)` qui renvoie une nouvelle image dont la profondeur est réduite à  $q$ . Par exemple, les images G, H et I de la figure 2 résultent des réductions à 1, 4 et 16 de la profondeur de l'image A.

### Partie III. Tramage

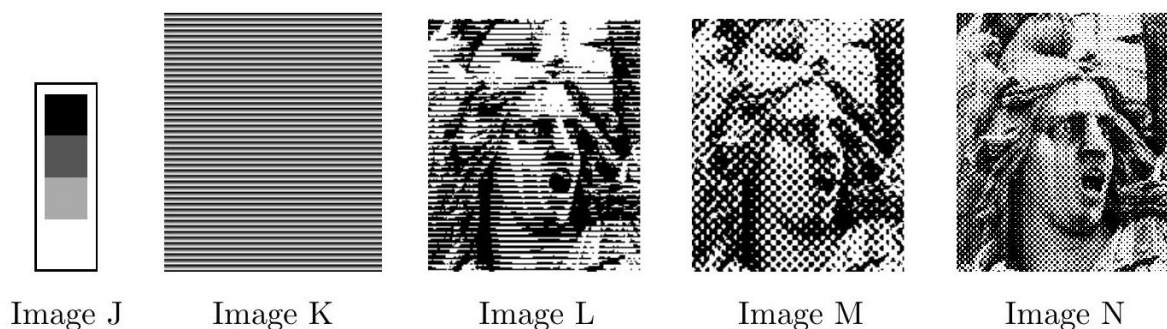


FIGURE 3 – Tramage

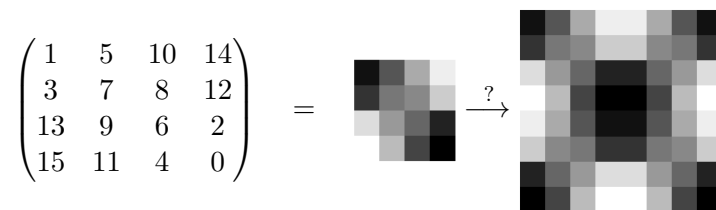
Lorsqu'il s'agit d'imprimer nos images, on se retrouve confronté à une difficulté : l'encre est noire, le papier est blanc. Il faut donc transformer les images en tons de gris en images en noir et blanc au sens strict. L'appel `reduire (img,1)` est un moyen de procéder, toutefois le résultat n'est pas très satisfaisant (voir l'image G de la figure 2). Cette partie examine la technique du tramage qui produit des images en noir et blanc (c'est-à-dire de profondeur 1) plus plaisantes, telles les images L, M et N de la figure 3. On peut voir l'image G comme produite par seuillage, les tons de gris plus grands qu'un certain seuil

deviennent blancs, tandis que les autres deviennent noirs. Une idée pour améliorer le rendu des images consiste à faire varier le seuil selon les pixels. Cela revient à représenter les seuils par une matrice, en fait par une image que l'on appelle une trame. Une trame est le plus souvent constituée par la répétition d'une petite image, dite **motif**, répétition selon les axes de coordonnées qui finit par paver toute l'image. Par la suite, le seuillage selon une trame sera simplement désigné comme le seuillage selon le motif dont dérive cette trame.

Par exemple, l'image J de la figure 3 est une échelle de gris verticale de profondeur 3 ( $h = 4, l = 1, p = 3$ ) et la matrice M est un vecteur colonne contenant les entiers de 0 à 3 du haut vers le bas). Sa répétition produit l'image K. On notera que l'image J est présentée agrandie par rapport à l'image K. Le motif J est adéquat pour seuiller les images de profondeur 4, par exemple l'image H de la figure 2, ce qui donne l'image L de la figure 3.

- 11) Écrire une fonction **tramer** (**img**,**tr**) qui prend deux images **img** et **tr** en arguments, et qui renvoie l'image de profondeur 1 obtenue en seuillant l'image **img** selon la trame **tr**. Le seuillage est défini précisément ainsi : étant donnés un ton  $v$  de l'image et un ton  $w$  de la trame, on obtient un pixel blanc si et seulement si  $v > w$ , et un pixel noir sinon.
- 12) Écrire une fonction **tramerTelevision**(**img**) qui prend en argument une image **img** de profondeur  $p$  et qui renvoie l'image de profondeur 1 obtenue en seuillant l'image **img** selon l'échelle de gris verticale de profondeur  $p - 1$ .

En imprimerie on utilise rarement des trames constituées de lignes horizontales comme la trame K. On préfère les trames constituées de lignes inclinées de points, ou trames diagonales. On obtient une trame diagonale de profondeur 15 par répétition du motif  $8 \times 8$  représenté ci-dessous, à droite :



Le motif de droite dérive de l'image  $4 \times 4$  représentée à gauche. On supposera par la suite qu'une variable globale **deuxQuarts** contient l'image de gauche.

- 13) Écrire une fonction **tramerJournal** (**i**) qui prend en argument une image  $i$  de profondeur 16, et qui renvoie l'image de profondeur 1 obtenue en seuillant l'image  $i$  selon la trame diagonale de profondeur 15. Par exemple, l'image M de la figure 3 résulte de l'application de **tramerJournal** à l'image I de la figure 2.
- 14) Indiquer un procédé simple qui permet d'obtenir l'image N de la figure 3 à partir de l'image I de la figure 2 et à l'aide de la trame diagonale de profondeur 15. On notera que l'image N est bien une image de profondeur 1, et qu'elle est représentée réduite (d'un facteur 2) par rapport à l'image M. Coder ensuite votre procédé comme une fonction qui prend une image de profondeur arbitraire en argument.