TP4

Objectifs
Manipulation des tableaux de deux dimensions.

Définition :

Une image **numérique** peut être représentée par une **matrice** où chaque élément correspond à un **pixel**.

Un **pixel** (abréviation de *picture element*) est la plus petite unité d'une image numérique. Chaque pixel représente une **intensité lumineuse** (dans le cas d'une image en niveaux de gris) ou une **couleur** (dans une image couleur).
Dans une image en niveaux de gris, la valeur d'un pixel est généralement comprise entre **0** (noir) et **255** (blanc).
Ainsi, une image I en niveaux de gris peut être représentée par une matrice d'entiers *I[a][b]*, où *a* et *b* désignent respectivement les **coordonnées du pixel** (ligne et colonne).

Avant d'appliquer les différents filtres, il est important d'initialiser et de vérifier l'affichage de la matrice représentant l'image.

Voici un exemple complet de programme Java permettant de déclarer, initialiser et afficher la matrice *I* :

```
public class ImageTest {
    public static void main(String[] args) {

        int[][] I = {
            { 52, 55, 61, 59, 79, 61, 76, 61, 64 },
            { 62, 59, 55, 104, 94, 85, 59, 71, 83 },
            { 63, 65, 66, 113, 144, 104, 63, 72, 73 },
            { 64, 70, 70, 126, 154, 109, 71, 69, 72 },
            { 67, 73, 68, 106, 122, 88, 68, 68, 65 },
            { 68, 79, 60, 70, 77, 66, 58, 75, 73 },
            { 69, 85, 64, 58, 55, 61, 65, 83, 90 },
            { 70, 87, 69, 68, 65, 73, 78, 92, 101 },
            { 72, 90, 78, 77, 74, 79, 85, 96, 105 }
        };
```

```
        // Affichage de la matrice originale
        System.out.println("Matrice originale :");
        for (int i = 0; i < I.length; i++) {
            for (int j = 0; j < I[i].length; j++) {
                System.out.print(I[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

Exercice 1.

Le ***Filtre moyenneur*** consiste à remplacer un pixel par la moyenne de ses voisins et de lui-même. Il existe deux types de voisinages : connexion 4 et connexion 8.

$\frac{1}{5}$ | 0 | 1 | 0 |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 0 |

connexion 4

$\frac{1}{9}$ | 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

connexion 8

Ainsi pour la connexion 4 nous avons :

$$g(a, b) = 1/5[I(a-1, b) + I(a, b-1) + I(a, b) + I(a, b+1) + I(a+1, b)]$$

et pour la connexion 8 nous avons :

$$g(a, b) = 1/9[I(a-1, b-1) + I(a-1, b) + I(a-1, b+1) + I(a, b-1) + I(a, b) + I(a, b+1) + I(a+1, b-1) + I(a+1, b) + I(a+1, b+1)]$$

Écrivez deux programmes *FiltreMoyenneurCon4.java et FiltreMoyenneurCon8.java* qui appliquent ces deux filtres à la matrice *I*.

Exercice 2.

Filtre conservatif : Le principe de ce filtre consiste à conserver la valeur du pixel si cette valeur est dans l'intervalle déterminé par les valeurs des 8 pixels voisins. Sinon la valeur du pixel sera remplacée par la valeur la plus proche parmi celles des pixels voisins

15	18	20	*filtre* conservatif → *Intervalle=[15,22]*	15	18	20
21	19	17		21	19	17
22	22	16		22	22	16
15	18	20	*filtre* conservatif → *Intervalle=[15,22]*	15	18	20
21	10	17		21	15	17
22	22	16		22	22	16
15	18	20	*filtre* conservatif → *Intervalle=[15,22]*	15	18	20
21	25	17		21	22	17
22	22	16		22	22	16

Étapes

Pour chaque pixel (a,b) :

- Trouver le **minimum** et le **maximum** des 8 voisins.
- Si I[a][b] < min → G[a][b] = min
- Si I[a][b] > max → G[a][b] = max
- Sinon G[a][b] = I[a][b]

Donnez en java le programme *FiltreConservatif.java* et qui applique ce filtre à la matrice *I*

Exercice 3.

Filtre médian : Le principe de cette méthode consiste à remplacer un pixel par la médiane de ses voisins. Cela consiste à ranger par ordre croissant les valeurs des pixels voisins et prendre pour nouvelle valeur du pixel celle se trouvant au milieu.

15	18	20	*filtre* médian → *10 15 16 17 18 20 21 22 22*	15	18	20
21	10	17		21	18	17
22	22	16		22	22	16

Étapes

- Récupérer les valeurs des 9 pixels du voisinage 3×3.
- Trier ces valeurs.
- Prendre la **valeur du milieu (5e)** comme nouvelle valeur du pixel.

Donnez en java le programme *FiltreMedian.java* et qui applique ce filtre à la matrice *I*