FERNANDES Jean-Luc
ANTEUR Neil

Projet Algorithmique 4
Modélisation

Dans le projet, on représente les capacités infinis de Edge par une capacité de 1000 (valeur inatteignable par le calcul de l'intérêt car les valeurs de couleurs des pixels varie entre 0 et 256)

```
public static void writepgm(int[][] image, String filename) ;
```

Elle prend en paramètre :

un tableau 2d correspondant à la valeur de la couleur de chaque pixel de l'image.
Le nom du fichier à créer.

Elle crée un fichier .pgm du nom filename avec les valeurs du tableau image.

```
public static int[][] interest (int [][] image) ;
```

Elle prend en paramètre un tableau 2d correspondant à la valeur de la couleur de chaque pixel de l'image.
Elle renvoie un tableau 2d des facteurs d'intérêt de chaque pixel.

Le facteur d'intérêt est calculé comme suit :

```
for(int i = 0; i<inter.length; i++){
	for(int j = 0; j<inter[i].length; j++){
		if(j == 0)
			inter[i][j] = Math.abs(image[i][j] - image[i][j+1]);
		else if(j+1 == inter[i].length)
			inter[i][j] = Math.abs(image[i][j] - image[i][j-1]);
		else
			inter[i][j] = Math.abs(image[i][j] - (image[i][j+1] + image[i][j-1]) / 2);
	}
}
```

```
public static Graph toGraph(int itr[][]) ;
```

. On construit un graphe à partir d'un tableau 2d de facteur d'interet. On procede comme suit :

-On crée un sommet par pixel.

-On crée un sommet source s, reliée à tous les pixels du bord gauche de l'image par une arête de capacité 1000.

-On crée un sommet destination t, relié à tous les pixels du bord droit de l'image par une arête dont la capacité est égale à l'intérêt de ce pixel.

-Chaque pixel (i, j) est relié au pixel (i, j + 1) par une arête dont la capacité est l'intérêt du pixel (i, j)

-Chaque pixel (i, j) est relié au pixel (i, j − 1),(i − 1, j − 1) et (i + 1, j − 1) par une arête de capacité 1000.

Cette fonction est appelé depuis le main.

```
public void initGraph(int itr[][]) ;
```

Elle prend en paramètre un tableau 2d d'interet crée par la methode interest.

Pour chaque ligne du tableau itr[][]
On initialise un tableau min[] à l'indice de la ligne avec la plus petite valeur contenu dans le tableau itr[][] sur cette ligne.

Pour chaque Edge e non infini on initialise setUsed à la valeur contenant dans min[indice_de_sa_ligne].

Elle est appelé depuis le main.

```
public int ameliorationResiduelle(ArrayList <Edge> edges, int from, int to) ;
```

```
/** Methode qui renvoie l'amelioration residuel d'un edge
   * @param edges La liste de tout les Edge du graph.
   * @param from La source du Edge
   * @param to La destination du Edge
   * @return l'amelioration residuel d'un edge
*/
```

L'amelioration residuelle est calculé comme suit : e.capacity – e.used;

Cette méthode est appelé depuis *augmentationMax()*

```
public int augmentationMax(int[] pred )
```

Elle prend en paramètre un tableau de prédécesseur qui représente un chemin.
Elle suit le chemin donne par pred[] et renvoie l'amélioration résiduelle du chemin.

Elle est appelé depuis calculFlotMax.

```
public int calculFlotMax() ;
```

Elle cherche des chemins augmentants et les augmente grâce à la methode updateGraph.
Elle s'arrête lorsqu'il n'y a plus de chemin augmentant.

```
public void upgradeGraph(int augmentation, int pred[]) ;
```

Elle prends en argument un int augmentation et un tableau de prédécesseur pred[] qui représente le chemin.
Elle augmente tout les edges du chemins de la valeur augmentation grâce à la methode upgradeEdge.

Elle est appelé dans calculFlotMax .

```
public void upgradeEdge(int to, int from, int augmentation) ;
```

Elle prends en parametre des entiers to et from qui correspondent au sommet source et destination du edge à augmenter. Elle prends aussi en paramètre augmentation qui est la valeur dont on doit

augmenter le edge.

La methode cherche le edge dans le graph et l'augmente de la valeur *augmentation.*

Cette methode est appelé dans *upgradeGraph.*

```
public int[] verifChemin() ;
```

Pour faire un parcours en largeur on utilise cet algorithme :

Initialement, tous les sommets sont non-marqués et la file est vide.
Marquer et insérer le sommet s de départ dans la file.

Tant que la file n'est pas vide
- Supprimer le sommet P situé en tête de file.
- Pour chaque successeur non marqué Q de P et edge(p,q) non plein
    - Marquer et insérer Q dans la file

Fin Pour
Fin Tant que

Renvoie le chemin pred[].

Cette méthode est appelé dans calculFlotMax.

```
public int[] coupeMinimal(int ligne, int colonne);
```

On parcours toute la première ligne

Pour chaque edge de la première ligne
Si il existe un edge PLEIN à la ligne suivante, pour la colonnes i-1, i ou i+1.
On initialise le tableau indiceCoupe[ligne] = i-1 , i ou i+1 selon le edge trouvé.
On refait la même chose avec la ligne suivante, avec i l'indice de la colonne du edge trouvé.
Fin Si.
On s'arrete lorsqu'on trouve un edge à la dernière ligne.

On renvoie un tableau indiceCoupe[] contenant les indices des sommets a supprimer pour chaque ligne.

```
public static int[][] cutImage(int[][] imageOriginal, int[] indiceCoupe) ;
```

Elle prends en argument un tableau 2d d'une image, *imageOriginal[]* et le tableau *indiceCoupe[]*

renvoyé par la méthode *coupeMinimal*

Elle parcourt le tableau imageOriginal[] et crée un nouveau tableau 2D en copiant les valeurs du tableau imageOriginal SAUF les valeurs définie par indiceCoupe.

Elle renvoie cette nouvelle image *newImage[][]*