Modélisation

d'après Emmanuel Jeandel

Modalités

Le projet doit être effectué par groupes de deux étudiants.

Le projet nécessite une réalisation logicielle en Java et doit utiliser les classes Java fournies. Il est possible de modifier légèrement les classes fournies (en particulier pour ajouter des méthodes à la classe `Edge`). Des modifications lourdes de ces classes, ou une implémentation n'utilisant pas ces classes, seront sanctionnées.

Le projet est à rendre pour le 27 mai. Le projet sera déposé sur l'elearning sous la forme d'une archive `Nom1Nom2.zip` où `Nom1` et `Nom2` sont les noms des deux étudiants.

L'archive doit contenir les fichiers et répertoires suivants :

— Le logiciel demandé. Le logiciel doit être programmé en Java et utiliser les classes Java fournies.

— un fichier README au format texte qui indique les noms et prénoms des membres du binôme, et comment compiler et lancer le programme ;

— un manuel d'utilisation au format PDF (le fichier docs/user.pdf) indiquant l'ensemble des fonctionnalités et comment les faire fonctionner ;

— une documentation pour développeur au format PDF (le fichier docs/dev.pdf) indiquant les choix pour la structure du programme, les algorithmes et méthodes utilisés, ainsi que les API et classes utilisées pour les implémenter.

— un fichier partie2.pdf répondant aux questions théoriques de la partie 2.

— un répertoire (docs/api) contenant la documentation générée par javadoc.

— deux images au format PGM, la deuxième correspondant au résultat de l'utilisation du logiciel sur la première.

Travail à effectuer

Le but est de réaliser un logiciel permettant de diminuer la taille d'une image en supprimant progressivement des colonnes "inutiles".

Le travail est en deux parties :

— La première partie consiste à produire un logiciel qui enlève 50 pixels par ligne, en enlevant colonne par colonne.

— La seconde partie consiste à répondre à quelques questions théoriques.

L'algorithme est assez lent : même sur une petite image, le temps de traitement pour retirer 50 pixels peut dépasser la minute. Il peut toutefois se généraliser pour le traitement d'objets 3D, plus communément appelés vidéos.

Les images pour tester sont au format pgm P2. Si on dispose d'un fichier jpeg,png, ou autre, on peut le convertir en format pgm P2 avec la commande :

```
convert -compress none toto.jpg toto.pgm
```

On peut afficher un fichier pgm en utilisant la commande `display`, en utilisant un gestionnaire de fichiers ou un navigateur.

1 Introduction

Le fichier `SeamCarving` contient une fonction permettant de lire un fichier au format pgm.

Q 1) Ecrire une fonction `writepgm(int[][] image, String filename)` qui écrit un fichier pgm.

Q 2) Ecrire une fonction `int[][] interest (int[][] image)` qui prend une image et qui renvoie un tableau de la même taille, contenant, pour chaque pixel, son facteur d'intérêt. Le facteur d'intérêt est défini comme suit :

- Le facteur d'intérêt est la différence (en valeur absolue) entre la valeur du pixel, et la moyenne des valeurs de ses voisins de gauche et de droite
- Si un pixel n'a pas de voisin de droite (pixels au bout de la ligne), c'est la différence (en valeur absolue) entre le pixel et le pixel précédent.
- Si un pixel n'a pas de voisin de gauche (pixels en début de ligne), c'est la différence (en valeur absolue) entre le pixel et le pixel suivant.

Par exemple, voici une image 4×3, et le tableau intérêt correspondant :

Image				Intérêt			
3	11	24	39	8	2	1	15
8	21	29	39	13	3	1	10
74	80	100	200	6	7	40	100

2 Algorithme

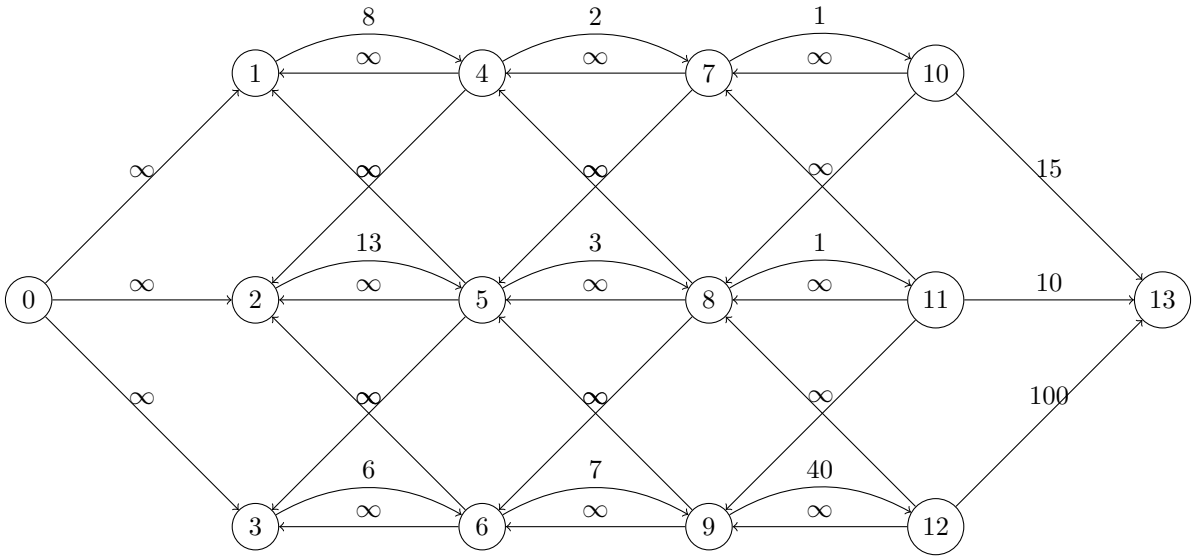
Dans toute la suite (i, j) représente le pixel ligne i colonne j.

On va résoudre le problème de trouver, pour chaque ligne, quel est le pixel à supprimer par une méthode de recherche de coupe minimale dans un graphe.

On construit le graphe suivant :

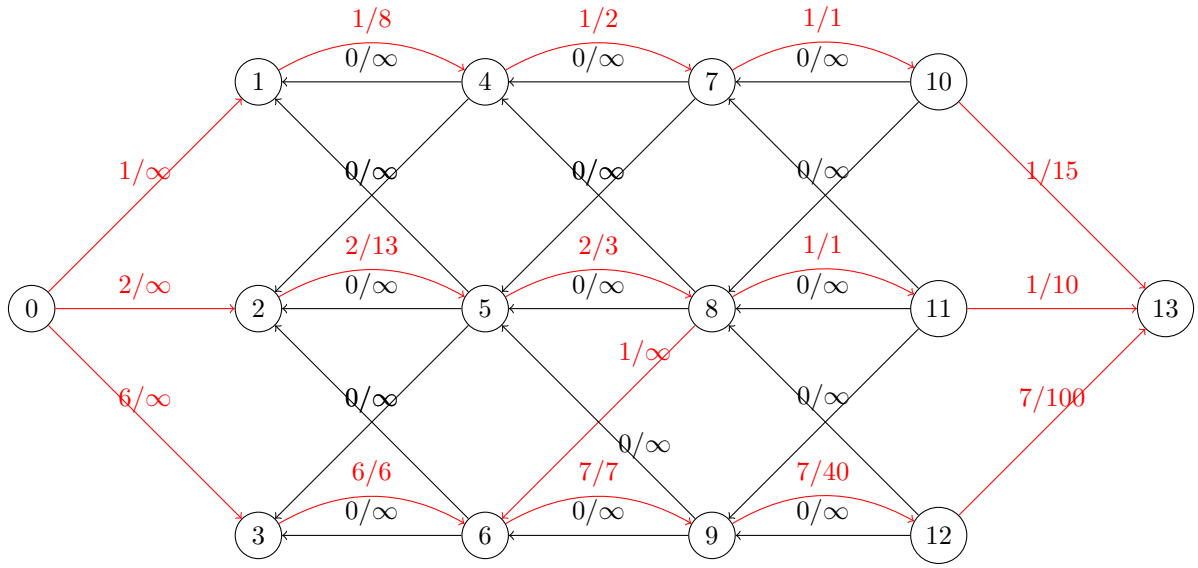
- On crée un sommet par pixel.
- On crée un sommet source s, reliée à tous les pixels du bord gauche de l'image par une arête de capacité $+\infty$.
- On crée un sommet destination t, relié à tous les pixels du bord droit de l'image par une arête dont la capacité est égale à l'intérêt de ce pixel.
- 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 capacite $+\infty$.

Voici ce qu'on obtient sur l'exemple :

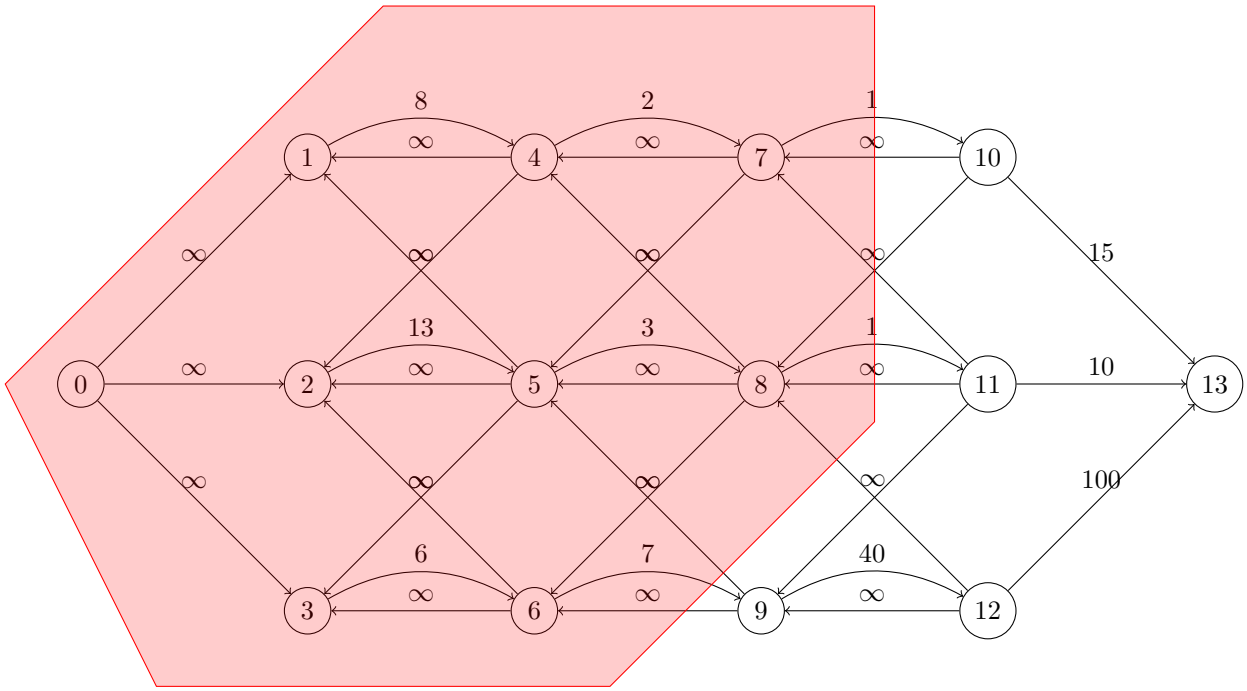


Le but est de trouver le flot maximum dans ce graphe ou, plus exactement la coupe minimale dans ce graphe.

Dans l'exemple le flot maximum est de valeur 9 et est obtenu de la façon suivante (les arêtes en rouge représentent les arêtes dont le flot est non nul) :



et qui correspondent à la coupe suivante :



Au vu de la coupe, les pixels à enlever sont ceux correspondant aux sommets sources des arêtes qui traversent la coupe, c'est à dire les sommets $6, 7$ et 8 sur le dessin.

3 Calcul d'une coupe minimale

Q 1) Ecrire une fonction `Graph toGraph(int[][] itr)` qui crée le graphe correspondant au tableau `itr`.

Q 2) Ecrire une fonction qui trouve le flot maximum entre s et t, et donc la coupe minimale entre s et t, dans un graphe.

On rappelle l'algorithme :

— On part d'un flot f de s à t initialement vide.

— On cherche un chemin augmentant. Deux cas peuvent se produire :

  — Le chemin existe. Dans ce cas, calculer la quantité maximum de flot qui peut passer sur ce chemin. En déduire un nouveau flot meilleur et recommencer l'algorithme

  — Il n'y a pas de chemin de s à t. Dans ce cas, on a atteint le flot maximum. Si on note S les sommets atteignables à partir de s, et T les autres, alors (S, T) est une coupe minimale du graphe.

Quelques détails pour l'implémentation :

— Vous devez utiliser les classes fournies.

— Ne pas partir d'un flot vide, mais d'un flot pas trop mauvais. Vous devrez expliquer dans le rapport comment vous avez obtenu ce flot.

— Pour chercher un chemin de s à t, utiliser un algorithme de parcours en largeur, et pas de parcours en profondeur.

Q 3) Utiliser toutes les fonctions précédentes pour écrire le logiciel demandé.

4 Partie théorique

On se place dans le cas d'une image quelconque, et pas uniquement de l'image 4×3 donnée en exemple. Soit (S, T) la coupe minimale.

Q 1) Montrer que sur une ligne, il est impossible d'avoir un sommet de T suivi d'un sommet de S. En déduire que sur une ligne, on a d'abord des sommets de S et ensuite des sommets de T. En déduire qu'il y a exactement un sommet (i, j) par ~~colonne~~ ligne tel que $(i, j) \in S$ et $(i, j+1) \in T$ (autrement dit : on enlève bien exactement un pixel par ligne)

Q 2) Soit (i, j) tel que $(i, j) \in S$ et $(i, j+1) \in T$. Montrer que

— Soit $(i+1, j) \in S$ et $(i+1, j+1) \in T$

— Soit $(i+1, j-1) \in S$ et $(i+1, j) \in T$

— Soit $(i+1, j+1) \in S$ et $(i+1, j+2) \in T$

(Autrement dit : les pixels qu'on enlève sur des lignes successives sont voisins les uns des autres)

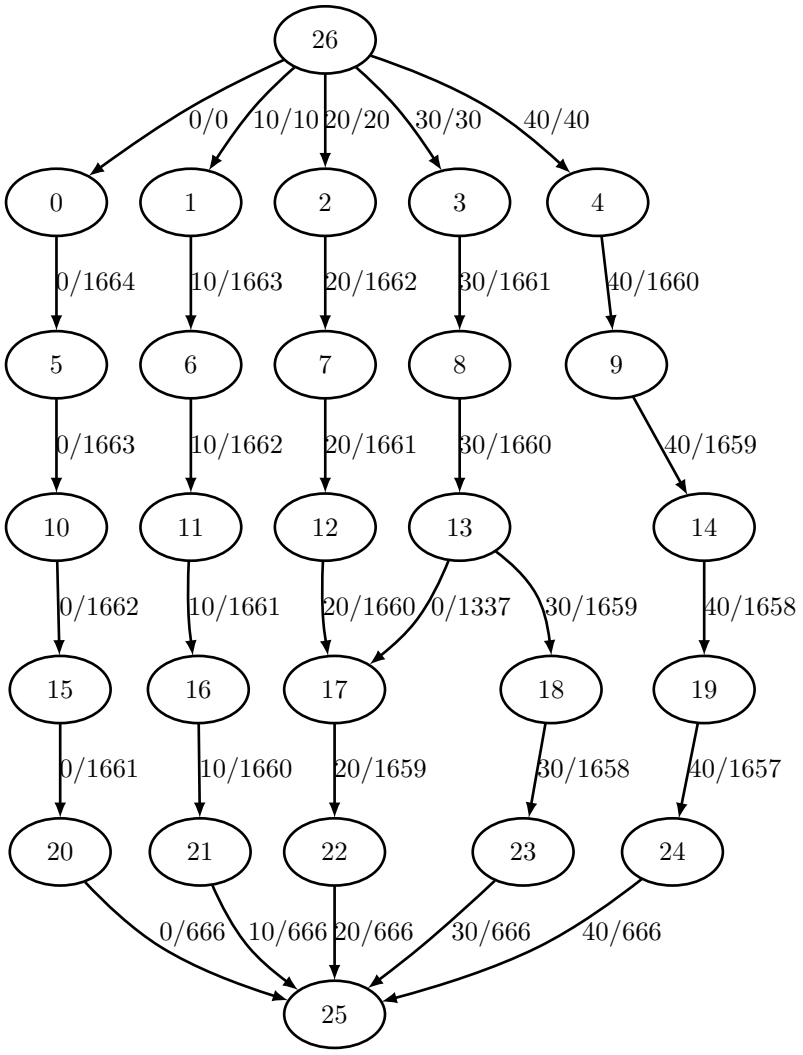


FIGURE 1 – Graphe construit par le fichier de test `Test`

5 Classes fournies

La classe `Graph` permet de représenter des graphes dont les sommets sont numérotés par des entiers. Le constructeur `Graph(int vertexCount)` crée un graphe dont les sommets sont les entiers entre 0 et `vertexCount`-1. On peut ensuite ajouter des arêtes au graphe en utilisant la fonction `addEdge(Edge edge)`. Les arêtes sont définies dans la classe `Edge` et ont une source (`from`), une destination (`to`) , une capacité (`capacity`) et un usage (`used`) correspondant à la quantité de flot qui circule actuellement sur l'arête.

La méthode `adjacent(int vertex)` énumère les arêtes adjacentes au sommet `vertex`. Il peut s'agir d'arêtes dont la source est `vertex`, ou d'arêtes dont la destination est `vertex`.

La classe `Graph` dispose d'une fonction `writedot` qui permet de transformer le graphe en un fichier `.dot`. Il est ensuite possible de visualiser le fichier .dot en utilisant la bibliothèque `graphviz`, ou un site web comme `http://sandbox.kidstrythisathome.com/erdos/`.

Une utilisation de la classe `Graph` est fournie dans la classe `Test`. Elle construit un graphe sous forme de grille et lance un parcours en profondeur sur ce graphe.

FIGURE 2 – Fichier initial

6 Remarques

Les images `ex1.pgm` et `ex2.pgm` sont dans le domaine public. L'image `ex3.pgm` a été fournie par Alexandre Buisse.

FIGURE 3 – Résultat en utilisant la méthode de la première partie puis en utilisant une autre méthode par fonction d'énergie avant