Modélisation

marine.minier@loria.fr

Le devoir doit être effectué par groupe de 1 *à* 2 *personnes, chaque personne appartenant à exactement un groupe. Je remercie Emmanuel Jeandel pour son sujet.*

Seam Carving

Le Seam Carving est une réduction de la taille d'une image basée sur l'algorithmique des graphes. L'idée est d'essayer de réduire l'image en éliminant les pixels inutiles et en gardant les pixels importants.

Voici l'effet de cette réduction sur un exemple

L'image initiale (d'une taille de 600x375) est présentée à la figure 1. On cherche à la réduire à une taille de 300x375.

La première technique classique consiste à changer l'échelle de l'image, et on obtient l'image de gauche de la figure 2, qui n'est pas très réussie. La deuxième technique consiste à ne garder qu'une portion de l'image de taille 300x375, par exemple, la partie au centre de l'image. On obtient l'image du milieu de la figure 2, qui est aussi un échec retentissant.

Le Seam Carving permet d'obtenir le résultat sur l'image de droite, qui est clairement meilleur.

FIGURE 1 – Image initiale

FIGURE 2 – A gauche, changement d'échelle (pas bien). Au milieu, recadrage (pas bien). A droite, l'algo de Seam Carving (bien)

Principe de l'algo

Pour supprimer 300 colonnes, l'algorithme enlève une colonne, puis une autre colonne, jusqu'à en avoir enlever 300.

On va donc expliquer comment enlever une colonne.

L'algorithme fonctionne en deux parties :

— On commence par calculer, pour chaque pixel de l'image, l'intérêt du pixel. Il y a plein de façons de définir l'intérêt d'un pixel, par exemple, on peut par exemple regarder la différence de couleur entre le pixel et ses deux voisins.

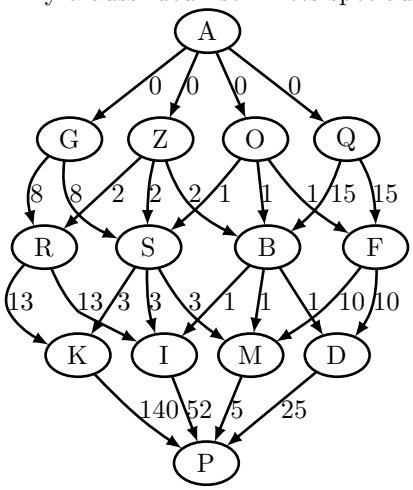
Dans l'exemple, voici ce qu'on obtiendrait :

Plus le pixel est noir, moins il est important.

— Ensuite, on cherche à éliminer une colonne de l'image. Eliminer une colonne veut dire éliminer un pixel par ligne, de sorte à former un chemin entre le haut et le bas de l'image. Pour trouver la colonne à éliminer, on cherche le chemin du haut en bas de l'image dont la somme des intérêts est la plus petite possible. C'est donc un problème de chemin de coût minimal dans un graphe.
Supposons par exemple que l'image soit une image de format 4x3 (et non pas l'image des chats), et que l'intérêt des pixels soient représentés dans le tableau suivant :

8	2	1	15
13	3	1	10
140	52	5	25

— On construit alors le graphe suivant : il y a un sommet par pixel, et chaque sommet est relié au pixel immédiatement en dessous de lui. Il y a aussi deux sommets spéciaux, un en haut et un en bas



— On cherche alors le chemin du haut en bas du graphe de coût minimal. Une fois qu'on a trouvé ce chemin, il nous donne les pixels inutiles, et on peut les éliminer. On a ainsi enlevé une colonne de l'image et on peut recommencer

1 Implémentation

Le fichier `SeamCarving` contient une fonction permettant de lire un fichier au format pgm. Dans ce format, chacun des pixels est représenté par un niveau de gris, entre 0 et 255.

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

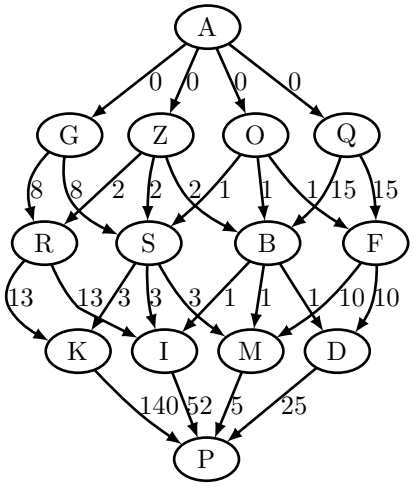
Q 2) Ecrire une fonction `int[][] interest (int[][] image)` qui prend une image et qui renvoie un tableau de la même taille, contenant, pour chaque pixel, son facteur d'intérêt. Le facteur d'intérêt est défini comme suit :

— 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
200	60	25	0	140	52	5	25

Q 3) Ecrire une fonction `Graph tograph(int[][] itr)` qui crée le graphe correspondant au tableau `itr`. Voici par exemple le graphe correspondant au tableau précédent :



Q 4) Ecrire une fonction `Bellman_Ford(Graph g, int s,t)` qui calcule le chemin de coût minimal entre les sommets s et t dans le graphe G en utilisant l'algorithme de Bellamn-Ford.

Attention cependant, il est nécessaire d'apporter quelques précisions pour l'implémentation de Bellman-Ford :

- Il ne faut pas garder toutes les lignes au fur et à mesure de l'éxécution (les lignes qui calculent la distance) sinon cela explose en taille. Il est donc plus judicieux d'en garder SEULEMENT deux consécutives et de s'arrêter SOIT quand les deux lignes sont identiques SOIT quand on fait $S-1$ itérations.
- Il faut également dans le tableau des pères, identifier précisément pour chaque sommet quel est le sommet père.

Q 5) Utiliser les fonctions précédentes pour écrire le logiciel demandé.

2 Fichiers fournis

2.1 Images

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.

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

L'image de format 4x3 fournie correspond à l'exemple donné dans l'énoncé.

2.2 Graph

La classe `Graph` est une classe abstraite permettant de représenter des graphes dont les sommets sont numérotés par des entiers. `GraphArrayList` est une implémentation à base d'`ArrayList`. Le constructeur `GraphArrayList(int N)` crée un graphe dont les sommets sont les entiers entre 0 et $N-1$. On peut ensuite ajouter des arêtes au graphe en utilisant la fonction `addEdge(Edge e)`. Les arêtes sont définies dans la classe `Edge` et ont une source (`e.from`), une destination (`e.to`) et un coût (`e.cost`).

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

3 Ce qu'il faut rendre

Dans un seul fichier .zip, merci de fournir :

- Le logiciel demandé (sources `.java`, et éventuellement un `.jar` en sus). Le logiciel doit être programmé en Java et utiliser les classes Java fournies. Il doit fonctionner sur une machine Linux, distribution Debian 8, disposant de 4 processeurs Intel-Xeon à 2.80 Ghz et où les paquets `openjdk-8-jdk` et `openjdk-8-jre` sont installés.
- Un rapport (un `.txt` est suffisant) expliquant
  - Comment la répartition du travail s'est effectuée entre les différents membres d'un même groupe. Cette partie n'est facultative que pour les groupes d'une personne et sera prise en compte dans la notation.
  - Une description de l'utilisation du logiciel, et des modifications apportées par rapport au sujet.
  - Il est important que le loigciel compile.
- Deux images au format PGM, la deuxième correspondant au résultat de l'utilisation du logiciel sur la première. L'image doit être différente des images fournies par l'enseignante.

4 FAQ

Q : Est-ce que je peux modifier les fichiers `Graph.java` ou `Edge.java` ?
A : Vous ne pouvez rien enlever, mais vous pouvez ajouter des méthodes et des champs si vous voulez.

Q : J'ai l'impression que mon code est bon, mais pourtant l'affichage du `pgm` est mauvais.
A : Vous utilisez OS X ? OS X est incapable d'afficher correctement un fichier `pgm` dont la résolution horizontale n'est pas multiple de 4. Convertissez le fichier pgm en fichier png (avec l'utilitaire convert) avant de

regarder si l'affichage est bon. Ou débrouillez-vous pour que votre fichier pgm ait une résolution horizontale multiple de 4.

5 Exemples

FIGURE 3 – Fichier initial.

FIGURE 4 – Résultat en utilisant la méthode proposée.

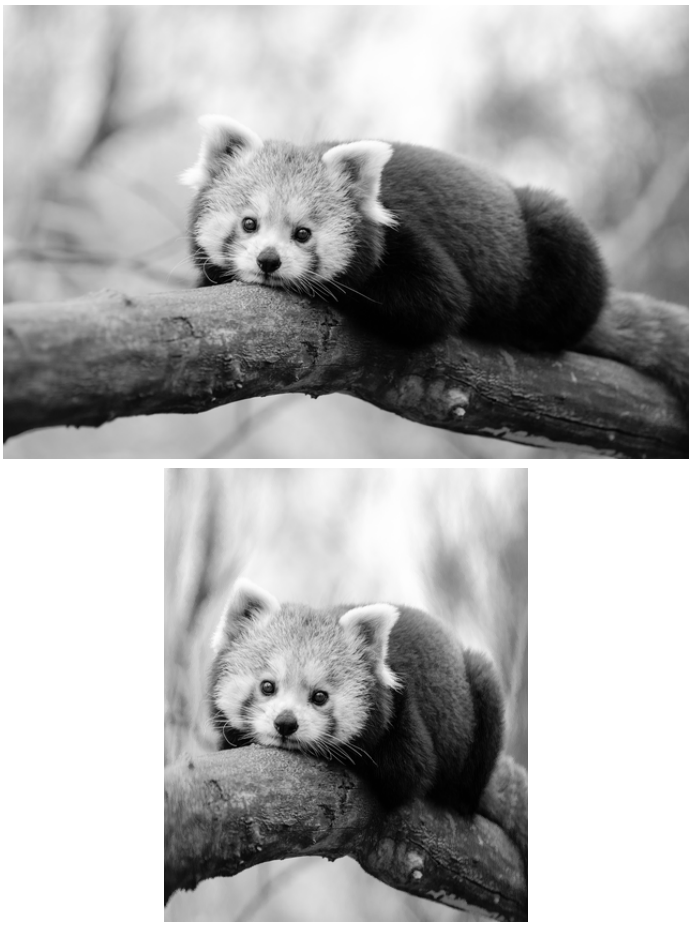


FIGURE 5 – On peut aussi utiliser des pandas roux

FIGURE 6 – Paysage