Algorithmique Avancée Projet 9B

Compression par gestion de graphe non orienté

Dibassi Brahima 19005521

3 Janvier, 2022

Contents

1 Introduction

Dans le cadre de la validation du cours de programmation avancée,
j'ai choisi pour projet final la Compression d'image par gestion de graphe non orienté a l'aide des listes de successeurs.
Je vais dans ce rapport expliquer le code de mon projet, les choix fait et pour finir voir les avantages et défauts de notre algorithme

1.1 Énoncé du projet

Nous considérons ici qu'une image est un graphe où chaque pixel est un noeud lié au plus à ses quatre voisins.
Nous ajoutons une contrainte : un pixel p est lié à son voisin v, si et seulement si p et v sont effectivement voisins (au sens des quatre voisins) et ont la "même" couleur.
Étudier ce graphe consiste à le gérer, avec son million de noeuds.
Mettez en place une méthode de recherche des parties connexes du graphe.
Votre méthode doit vous avoir fourni un numéro pour chaque partie connexe.
Il suffit maintenant de parcourir une partie connexe en cherchant les sommets qui n'ont pas quatre successeurs, ce sont les sommets de la frontière de la région.
Ces sommets seront conservés dans une liste (ou un vecteur).
Il est conseillé de se souvenir aussi des pixels liés ou des pixels non liés (au choix).
La liste de ces listes compose les pixels de bords des régions. Ils organisent donc une partition de l'image.
Avec cette liste de listes et la couleur qui y est associée, vous créez une nouvelle façon de stocker l'image.
Il faut maintenant, mettre en place un système de sauvegarde et un système de restauration de l'image.

2 Documentation

2.1 Utilisation

2.1.1 Demo Compression et Décompression Rapide

```
bash ./Exec.sh [FileName] [0 >= Max_Delta <= 20]
```

FileName : Un nom de fichier sans son extension dans Img/

Max_Delta: La somme des écarts maximaux entre les 3 composantes RGB de deux couleurs qui est acceptée.

La sortie de la Décompression se trouve dans le fichier Demo_Output.ppm

2.1.2 Compression

```
./Comp [FilePath] [0 >= Max_Delta <= 20]
```

FilePath : Un Path d'un fichier.ppm

Max_Delta: La somme des écarts maximaux entre les 3 composantes RGB de deux couleurs qui est acceptée.

La sortie de la Compression se trouve dans le fichier [FilePath].Compressed

2.1.3 Décompression

```
./Decomp [FilePath]
```

FilePath : Un Path d'un fichier.Compressed

La sortie de la Décompression se trouve dans le fichier ./Demo_Output.ppm

2.2 Fichiers

Résumé des fichiers du projets et leurs contenu

List : Gestion des listes chaînée génériques

Ima : Chargement/Ecriture Fichier ppm

Shared_Comp_Decomp : Structures et Fonctions partagées entre la compression et la décompression

Comp : Compression d'une image Decomp : Décompression d'une image

3 Déroulé du projet

3.1 Choix

3.1.1 Suppression ou non des petites zones

L'une des plus forte limitation de cette algorithme de compression est les petites zones.
Dans ce projet j'ai fait le choix après réflexion de ne pas les supprimer, car je n'aime pas l'idée d'induire de la perte "non contrôlée".
Lorsque je supprime les zones puis que la décompression essaye de les remplir j'ai des résultats très différents d'un test à l'autre.
De plus je trouve que leurs supression cache les faiblesses de l'algorithme

3.1.2 Limitation de la taille de l'image a 4096 * 4096

Tout simplement car une immense partie des images sont inférieures à cette taille.
De plus cette limitation permet d'écrire les index des pixels non plus sur 4 octets mais sur 3 octets.
On gagne instantanément en taux compression.

3.2 Problèmes Rencontrés

3.2.1 Décompression

J'ai eu beaucoup de mal à trouver comment recréer mon image à partir des bords des zones.
Après pas mal de réflexion j'ai eu l'idée d'utiliser des paires de pixels de bords cela me permet de dire avec certitude qu'elle est la couleur entre ses deux extrémités.

J'ai pu grâce à cette idée réduire énormément le nombre de bords que j'écris dans mon fichier compressé.
J'écris désormais que les bords à l'extrémité de chaques "lignes" Dans les meilleurs cas, il est possible d'écrire une zone entière sur seulement 15 octets.

3.2.2 Code Complexe

Lors de mon premier jet du projet j'ai écris du code très difficile à déboguer et donc a amélioré.
Lorsque j'ai repris le cours et après avoir compartimenté chacun des modules du projet, j'ai pu refaire le projet sur de bonnes base ce qui m'a permis d'aller plus loin.

4 Améliorations

4.1 Apportées

4.1.1 Region Merging

L'idée est asser simple puisque maintenant l'image est ecrite en zone de couleur,
il s'agit simplement ici de regrouper avant l'écriture les zones par couleurs.
Cela permet dans une moindre mesure d'absorber les zones à faible nombre de pixels.

4.1.2 Bords Pairs

L'idée ici est de ne pas écrire tous les bords mais des paires de bords Debut_Ligne Fin_Ligne.
Nous savons qu'il ya que des Pixels(Bords inclus) de notre Zone entre ses deux bornes.

4.2 Possibles

Ici ce sont juste des hypothèses dont il faudrait tester pour certaines l'efficacité

4.2.1 Huffman Encoding

Simplement encoder le [file.Compressed] selon le codage d'huffman.
On gagne pas mal en compression.
Puisque ce n'est pas le sujet du projet je ne l'ai pas ajouté à l'archive.

4.2.2 Analyze Automatique

Le but serait d'ajouter un choix de [Max_Delta] automatique, par couleurs.
Cela demande de trouver une formulation mathématique de la différence "visuel" entre deux couleurs autre que la simple distance euclidienne
(Je n'ai pas pu le faire par moi même).
Un Exemple du type de formulation mathématique **Ici**.

5 Conclusion

5.1 Analyze de l'algorithme

Avec de nombreux tests sur des images très diverses est variée on se rends compte des faiblesse et des force de notre algorithme.

5.1.1 Pros

- Taux de compression spectaculaire sur image adapté
- Extrêmement Rapide sur image adapté
- Possibilité avec ou sans perte

5.1.2 Cons

Cet Algorithme a une grosse faiblesse c'est qu'il est extrêmement dépendants de l'image que ce soit en mémoire,vitesse ou taux de compression en tout cas dans l'implémentation que j'en ai fait.

5.1.3 Exécutions

Jeux d'exécutions avec des delta qui permettent de retrouver l'image sans différences visibles.
Toutes les images se trouvent dans l'archive

Image	Max_Delta	Taux Compression	Temps Compression
Tintin	20	44.92%	14 sec
Tintin	10	21.92%	30 sec
Black	0	98.04%	8 sec
Red	0	98%	0.66 sec
Code_Cat	5	97%	24 sec
Cool_Cat	10	96 %	2.5 sec
Among_us	10	97%	0.23 sec
Pixel_Art	5	41%	4 sec
SpriteSheet1	0	68.64%	1.69 sec
SpriteSheet2	0	81.04%	0.56 sec