Licence 2
Algorithmique & Structures de données 2

Université de Nantes
2015/2016

Projet 2
Compression d'images bitmap

4 séances

L'objet du projet est de réaliser la compression d'images bitmap au moyen d'une représentation arborescente de celles-ci.

1 Description du problème à traiter

Les images bitmap sont composées d'une description de la couleur de chaque pixel de l'image. Cette couleur est typiquement codée par ses trois composantes, rouge (R), vert (V) et bleu (B), chacune codée dans un octet (8 bits), produisant une palette de $256^3 = 16\,777\,216$ de couleurs. Le principal souci des images bitmap est leur poids en octets : une image de largeur L et de hauteur H pèse a minima [1] $L * H * 3$ octets ; par exemple, le poids d'un fond d'écran 1920×1080 est au minimum de $6\,220\,800\,o \approx 6Mo$. Le format standard PNG (https://fr.wikipedia.org/wiki/Portable_Network_Graphics) utilise un algorithme de compression qui permet de réduire cette taille. Il exploite, entre autres, la présence de zone de couleur identique. Cette compression est toutefois sans perte : la couleur de chaque pixel de l'image compressée reste identique à celle de l'image non compressée.

L'objectif du projet et de réaliser une compression avec perte d'une image bitmap enregistrée au format PNG et d'en observer les effets sur sa *qualité* et son *poids*. Pour ce faire on utilisera une représentation de l'image sous forme d'un arbre de recherche d'arité 4 appelé *quadtree* spécialement conçu pour représenter un espace 2D (http://en.wikipedia.org/wiki/Quadtree, http://s.gury.free.fr/QuadTree/QuadTree.PDF).

1.1 Représentation d'une image par un *quadtree*

Un quadtree est un arbre enraciné 4-aire complet (tous les nœuds internes ont degré 4), les fils de chaque nœud étant ordonnées (numérotés de 0 à 3). Un quadtree de hauteur n peut être utilisé pour représenter une image de $2^n \times 2^n$ pixels. **Par simplicité, nous ne traiterons que des images bitmap carrées dont le côté est une puissance de deux.**

Chaque nœud du quadtree à profondeur k (entre 0 et n) représente une région carrée de l'image, de taille $2^{n-k} \times 2^{n-k}$ pixels et stocke la couleur moyenne des pixels de la région, calculée inductivement comme la moyenne des couleurs (moyennes) des 4 régions filles. Chaque feuille (à profondeur n) représente ainsi 1 pixel et stocke la couleur exacte de ce pixel.

Le chemin reliant la racine de l'arbre à un nœud détermine la position de la région représentée par ce nœud dans l'image ; pour une feuille, ce chemin détermine ainsi la position (x, y) du pixel correspondant dans l'image : le k-ième bit de x (resp. y) est le bit de poids fort (resp. faible) du numéro du fils (entre 0 et 3) utilisé depuis le nœud de profondeur k sur le chemin. Réciproquement, le chemin depuis la racine correspondant à un pixel de coordonnées (x, y) donné est directement défini par les valeurs de x et de y : au nœud de profondeur k, le numéro du fils (entre 0 et 3) où poursuivre la recherche est donné par la concaténation des bits de rang k de x et de y.

Ainsi, la racine correspond à toute l'image, son fils 0 représente le quart Nord-Ouest ($x = 0\ldots, y = 0\ldots$), son fils 1 le quart Sud-Ouest ($x = 0\ldots, y = 1\ldots$), son fils 2 le quart Nord-Est ($x = 1\ldots, y = 0\ldots$) et son fils 3 le quart Sud-Est ($x = 1\ldots, y = 1\ldots$) de l'image, chaque quartier étant lui-même subdivisé de même aux profondeurs suivantes du quadtree.

1. Outre la couleur, d'autres informations (translucidité, ...) sont généralement stockées.

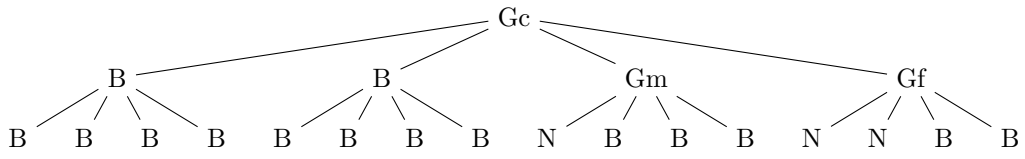

FIGURE 1 – Un quadtree représentant la lettre "i" dans une image 4×4 bichromatique.

Exemple 1 *La figure 1 illustre un quadtree représentant une image* 4×4*. Les couleurs y sont codées par des lettres :* $B = (255, 255, 255)$ *[blanc],* $N = (0, 0, 0)$ *[noir],* $Gm = (191, 191, 191)$ *[gris moyen],* $Gf = (127, 127, 127)$ *[gris foncé],* $Gc = (207, 207, 207)$ *[gris clair].*

Tous les pixels se trouvent à profondeur 2 *et sont noirs ou blancs. La première feuille noire à partir de la gauche a pour coordonnées* $(2, 0)$ *puisque le chemin qui la relie à la racine suit : racine* $\rightarrow$ *fils* 2 $(= 10_b)$ $\rightarrow$ *fils* 0 $(= 00_b)$*. Inversement, le pixel de coordonnées* $(0, 2)$ $= (00_b, 10_b)$ *correspond au chemin : racine* $\rightarrow$ *fils* 1 $(= 01_b)$ $\rightarrow$ *fils* 0 $(= 00_b)$ *; ce qui permet de voir que sa couleur est blanc.*

Les nœuds internes contiennent les couleurs moyennes de leurs fils, qui sont différentes nuances de gris. Par exemple, la couleur Gm résulte de l'opération $\frac{0+255+255+255}{4}$ *pour chacune de ses composantes. Les coordonnées du nœud portant cette couleur sont obtenues comme celles des feuilles mais en remplaçant par* 0 *les bits manquants : racine* $\rightarrow$ *fils* 2 $= 10_b$ *donne donc* $(10_b, 00_b) = (2, 0)$*. On constate que c'est aussi la coordonnées de son fils* 0 *! Sa profondeur* 1 *indique cependant qu'il représente une région de taille* $2^{2-1} \times 2^{2-1} = 2 \times 2$ *: il "contient" en effet les pixels de coordonnées* $(2, 0)$*,* $(2, 1)$*,* $(3, 0)$ *et* $(3, 1)$*.*

1.2 Compression de *quadtree*

Pour gagner de la place, il est possible de réduire l'arbre par élagage (élimination de feuilles) : si toutes les feuilles d'un même nœud ont la même couleur, on peut alors supprimer ces quatre feuilles ; on fait ainsi diminuer l'arbre (gain d'espace mémoire) et on rend l'accès à la valeur d'un pixel plus rapide (gain de temps). Cette compression se fait sans perte, i.e., l'image n'est pas dégradée.

Par exemple, la Figure 2 illustre la compression sans perte du quadtree de la Figure 1.

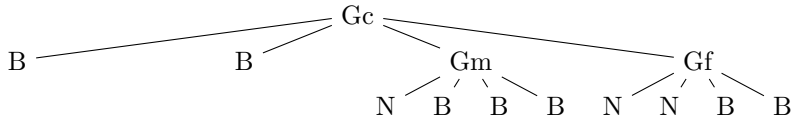

FIGURE 2 – Quadtree de la figure 1 compressé (sans dégradation).

En autorisant un certain niveau de perte, on peut réduire l'arbre plus fortement encore. Nous proposons deux méthodes, l'une basée sur une qualité minimale, l'autre sur un poids maximal :

1. On peut définir une dégradation maximale autorisée sous la forme d'un entier Δ : les quatre feuilles $F_0, \ldots, F_3$ d'un même nœud N peuvent être supprimées ssi la différence maximum de luminance (moyenne[2] des trois composantes) entre la couleur d'une feuille et la couleur moyenne du nœud n'excède pas Δ :

$$max_{i \in 0..3} \frac{|(F_i.R + F_i.V + F_i.B) \ - \ (N.R + N.V + N.B)|}{3} \leq \Delta.$$

   Voir Figure 3 par exemple.

2. On peut également définir la dégradation en fonction d'un poids maximum souhaité, ce qui se traduit par un entier Φ représentant le nombre maximum de feuilles autorisées dans l'arbre. Les feuilles sont alors élaguées par ordre croissant de différence de luminance maximum (cf. supra). Voir Figure 4 par exemple.

2. Normalement, cette moyenne est pondérée par des coefficients sur les composantes reflétant leur "contribution relative" à la luminosité de la couleur considérée.

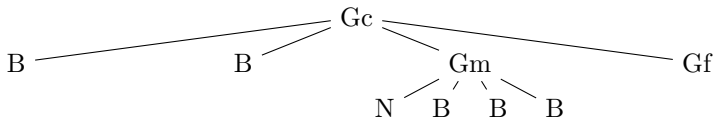

FIGURE 3 – Quadtree de la figure 1 compressé (dégradation avec $\Delta = 128$).

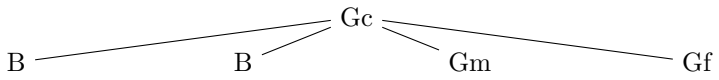

FIGURE 4 – Quadtree de la figure 1 compressé (dégradation avec $\Phi = 4$).

2 Travail demandé

Le projet devra être rendu avant le **lundi 2 mai à 12h00** par dépôt d'une archive ZIP sur Madoc. L'archive sera intitulée aux noms des membres du binôme. Elle comprendra :

— le code source documenté de votre projet ;
— un fichier texte donnant les instructions de compilation et d'utilisation de votre programme ;
— votre rapport (environ 10 pages) au format PDF.

2.1 Implémentation

Votre implémentation s'appuiera sur le code source fourni sur Madoc et qui comprend :

— La bibliothèque LodePNG (`http://lodev.org/lodepng/`), encapsulée pour ce projet (fichier `liblodepng.a`). Pour compiler un programme utilisant cette bibliothèque, il faudra ajouter les instructions `-std=c++11 -llodepng` à votre commande de compilation usuelle.
— La classe `ImagePNG` (fichier `imagepng.hpp`) qui représente une image bitmap en mémoire.
— Une ébauche de la classe `QuadTree` (fichier `quadtree.hpp`) à compléter.
— Des programmes de tests (fichiers `test_imagepng.cpp` et `test_quadtree.cpp`).

Outre ces fichiers sources, vous trouverez sur Madoc un échantillon d'images PNG de tailles variables pour tester votre programme (fichier `pngs.zip`).

Vous devrez fournir le code source complété, ainsi que le programme de chronométrage que vous aurez utilisé pour mesurer les performances pratiques des méthodes publiques de `QuadTree`.

2.2 Rapport

Votre rapport comportera les parties suivantes :

1. Une introduction présentant les objectifs du projet dans le cadre du module (selon vous).
2. Une présentation des algorithmes que vous avez mis en œuvre pour chaque méthode de la classe `QuadTree` que vous avez implémentée. Vous discuterez en particulier les choix importants que vous avez fait pour leur implémentation.
3. Une analyse de la complexité spatiale d'un quadtree.
4. Une analyse théorique de la complexité temporelle des algorithmes présentés (ordre de grandeurs justifiés informellement).
5. Une analyse expérimentale de la complexité temporelle des algorithmes présentés. Elle sera réalisée en jouant sur les nombreux paramètres du problème (taille de l'image, variété des couleurs dans l'image, valeurs de Δ et Φ, ...). Votre protocole expérimental et votre base de jeux de tests seront présentés et leurs pertinences sera discutée. Les résultats devront être comparés à l'étude théorique menée par ailleurs.
6. Une conclusion en forme de discussion critique sur votre travail et les extensions ou variations envisageables, avec une estimation de leur impact en terme de complexités spatiale et temporelles (par exemple, d'autres structures d'arbres, ...).

La présentation et la rédaction du rapport seront prises en compte dans son évaluation.