TP2: Compression avec perte

23, 30 janvier, 6 février

Rapport à rendre avant le 7 février minuit.

Le but du rapport est de comparer les taux de compression en fonction de la qualité obtenue pour les différentes méthodes proposées dans le TP.

Il n'y a pas de questions précises auxquelles répondre et vous êtes libres d'orienter le rapport dans votre sens. Les "idées d'améliorations" à la fin peuvent vous donner des idées.

La concision et l'esprit de synthèse seront valorisé. En particulier, mieux vaut un rapport court et intéressant que long et vide. Longueur indicative : 2 pages.

La compression de données traite du problème de réduire l'espace utilisé par des informations en vue de les stocker ou de les transmettre. On distingue principalement deux modes de compression :

- *Compression sans perte* : le fichier compressé permet de retrouver l'intégralité des informations d'origine. Exemple : ZIP, gzip, Huffman, PNG, FLAC,...
- *Compression avec perte* : lorsque l'on veut compresser des données de type visuelles ou sonores, on peut s'autoriser à perdre un peu d'informations si cette perte est suffisamment discrète. Exemple : MP3, DivX, JPEG, MPEG.

Généralement une méthode de compression avec perte est composée d'une phase de transformation où l'on va décider quelles sont les données que l'on va négliger suivie d'une phase de compression sans perte. Dans ce TP, on se propose d'étudier une méthode de compression de courbe avec perte et de la comparer à une méthode naïve. Afin de valider notre méthode, on s'intéressera au facteur de compression pour une qualité donnée.

Pour commencer

Récupérer le fichier "Courbe.java" sur la page web du cours. Il comporte une classe `Courbe` ainsi que deux méthodes de génération aléatoire de courbe. Un second fichier "DessinCourbe.java" est aussi disponible pour vous permettre de visualiser graphiquement vos résultats.

Exercice 1. Méthode naïve

On se propose dans cet exercice d'écrire un algorithme naïf à taux de compression constant : pour obtenir un taux de n, on remplace n valeurs consécutives par leur moyenne.

a. Écrire une méthode `static double[] moyenne(double[] lacourbe, int n)` qui rend un tableau n fois plus petit.

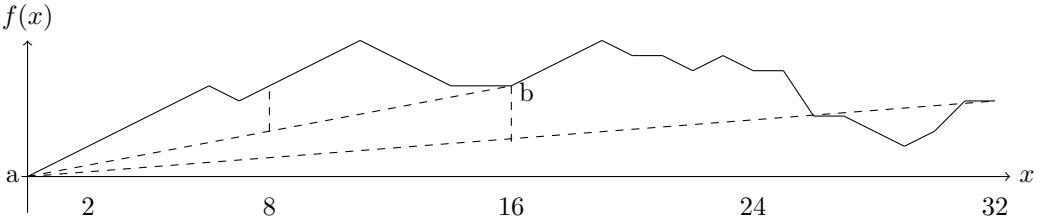
b. On suppose que les fonctions représentées sont à valeur entre 0 et 1. Ajouter à la classe courbe une méthode `double valeur(double x)` rendant la valeur de la fonction au point $x \in [0, 1]$.

c. La fonction de qualité à laquelle on s'intéresse est la distance euclidienne entre les deux courbes : $\sqrt{\int_0^1 (f-g)^2}$. Écrire une fonction permettant de calculer cette distance. On pourra utiliser une valeur approchée de l'intégrale : $\int_0^1 f \approx \sum_{i=0}^{k-1} f(i/k)/k$ pour k assez grand).

d. Calculer la qualité moyenne de compression de votre méthode pour $n = 3$ pour les deux générateurs de courbes donnés.

Exercice 2. Transformation en Arbre

L'idée de cet algorithme est de transformer la courbe représentée par tableau $[f(0), f(1), \ldots]$ en un arbre puis de supprimer les feuilles "peu importantes". Supposons que la courbe dont l'on dispose est 0,1,2,3,4,5,6,5,6,7,8,9,8,7,6,6,6,7,8,9, 8,8,7,8,7,7,4,4,3,2,3,5,5. La courbe obtenue est la suivante :



On peut représenter la courbe par un arbre : on code tout d'abord les deux points extrêmes 0 et 5. Ensuite on construit récursivement un arbre binaire étiqueté :

- l'étiquette de la racine est l'écart à la moyenne entre les deux points extrêmes, (5-0)/(32-0)*16=2.5 pour l'exemple, et la valeur réelle de ce point, 6 dans l'exemple. L'étiquette de la racine est de 3.5 dans l'exemple.
- On construit le reste de l'arbre récursivement : le fils gauche représente la partie gauche de la courbe, le fils droit la partie droite.

a. Écrire une classe `Arbre` permettant de représenter un arbre binaire.

b. Écrire une classe `CourbeArbre` représentant une courbe (ie un `Arbre` plus les valeurs "extrêmes").

Pour la question suivante, on suppose que les tableaux sont longueur 2^N+1 et les arbres binaires complets.

c. Écrire les fonctions permettant de passer de la représentation `CourbeArbre` à la représentation `Courbe`.

Exercice 3. Effeuillage de l'arbre et compression

Revenons au sujet initial : la compression. Pour cela, on propose de supprimer certaines feuilles de l'arbre (en pratique, on remplacera leur étiquette par 0). On appelle taux de compression le rapport entre le nombre d'étiquettes mises à 0 et le nombre d'étiquettes total.

On donne trois méthodes pour supprimer les feuilles, à chaque fois, on demande d'étudier le taux de compression et la qualité de la compression et de comparer à la méthode naïve.

- On supprime les étiquettes inférieures à un certain paramètre ϵ.
- Compression à qualité fixée : lors de la construction de l'arbre, si l'écart maximal entre la courbe et la droite entre le point gauche et droit de la courbe (courbe en pointillés entre les points a et b sur le dessin) est inférieur à ϵ, on supprime les fils.
- Compression à taux de compression fixé : on effectue l'opération ci dessus en faisant varier le facteur ϵ de façon à obtenir le taux de compression voulu.

Exercice 4. Idées d'améliorations

a. Dans les exercices précédents, on s'est limité à des tableau de longueur 2^N+1. Imaginer des algorithmes pour passer outre cette limitation.

b. Que deviennent les résultat si l'on prend des courbes correspondant à des données réelles (exemples : mesures physiques, images,...) ?

c. Proposer (en justifiant) d'autres fonctions de qualité.

d. Proposer d'autres algorithmes d'effeuillages que les trois ci dessus.

e. Proposer une méthode similaire qui pourrait s'appliquer à des images.