Examen Codage Transmission Compression M2IM 2005-2006

Le 8/2/2006, Tout document autorisé

Recopier tel quel des extraits de documents n'apporte jamais de points. Vous devez répondre aux questions qui vous sont posées. Vous devez argumenter (étayer) vos réponses. Une réponse correcte non étayée ne rapporte pas de points, et une réponse incorrecte bien étayée peut en rapporter.

Compression de graphes

Pour simplifier, on travaille sur des graphes orientés , connexes, de plus d'un noeud, sans arcs doubles et qui n'ont **aucune** information sur les arcs ni sur les noeuds. Ceci veut dire que les noeuds et arcs ne sont pas numérotés, on ne peut pas faire la différence entre le graphe $A \to B \to C$ et le graphe $B \to C \to A$ Attention, cela ne veut pas dire que l'orientation n'a pas d'importance.

Il est important que vous discutiez des solutions que vous proposez (avantages, inconvénients, pourquoi ce n'est pas optimum, améliorations possibles, ..)

1. Proposez une méthode de codage (pas compression) des graphes où le nombre d'arcs est petit.
2. Proposez une méthode de codage (pas compression) des graphes où le nombre d'arcs est grand.
3. Proposez une méthode de compression du codage que vous avez défini dans 1.
4. Proposez une méthode de compression du codage que vous avez défini dans 2.

Compression de signal

On travaille avec des images quelconques.

1. Il n'est pas possible d'avoir une méthode unique comprimant correctement (qualité et taux de compression) toutes les images. Pourquoi ?
2. Si l'on dispose d'une palette de méthodes de compression, quels critères proposez-vous pour choisir la méthode de compression à appliquer à l'image ?
3. Donnez l'algorithme de haut niveau inspiré de la question précédente pour comprimer et décomprimer en supposant que l'on applique une unique méthode de compression à toute l'image.
4. Si l'on veut comprimer une séquence d'images, proposez une méthode d'accélération de la compression donnant généralement d'aussi bons résultats que la compression des images indépendemment les unes des autres.
5. Quels sont les inconvénients de la méthode indiquée à la question 3.
6. Proposez une méthode de compression récursive améliorant le taux de compression.
7. Énumérez les inconvénients de votre méthode et proposez pour chacun d'eux une idée d'amélioration.
8. On suppose que vous avez un ordinateur qui calcule **infiniment** rapidement.

   Proposez un algorithme de compression **optimal** pour de la compression de données sans perte. L'algorithme à trouver est très très court.
9. Comment modifier l'algorithme pour pouvoir faire de la compression d'images avec perte ?

Une correction

Codage de graphes avec peu de noeuds

Le plus simple : un tableau d'arcs, un arc étant défini par une paire de numéro de noeuds. Pour le graphe suivant : **1 2 3 1 4 3 4 2 5 4** (1 + 5*2 = 11 entiers)

```
1 -----> 2
^        ^
|        |
|        |
3 <----- 4 <---- 5
```

On utilise rarement ce codage dans la réalité car c'est difficile de parcourir le graphe. Le codage le plus courant est :

Un tableau noeuds, chaque noeuds contenant une table des arcs sortants. Pour le même graphe c'est : **1(2)2()3(1)4(2 3)5(4)** simplifiable en enlevant les numéros de noeuds puisque ce sont les indexes. **2))1)2 3)4)** (1 + 10 entiers) Ce codage est meilleur en place que l'autre si les noeuds ont plus d'un arc sortant.

Le codage de chemins dans le graphe est très joli mais relève plus de la compression que du codage car ils sont difficiles à utiliser. Dans cet exemple en deux chemins : **5 4 3 1 2,4 2** Ou bien en utilisant un marqueur pour empiler/dépiler le noeud courant : **5 4(3 1 2)2**

Codage de graphe avec beaucoup de noeuds

En supposant qu'il n'y a pas d'arêtes doubles. On peut coder le graphe comme un tableau carré de bits.

```
  12345

1 01000
2 00000
3 10000
4 01100
5 00010
```

Ce codage utilise 1 entier + 25 bits. Il est meilleur que le précédent bien qu'il y ai peu de noeuds car le nombre de noeud est faible et que la place occupée par ce codage est fonction du nombre de noeuds au carré.

Compression de graphes avec peu de noeuds

- On renumérote les noeuds pour que des noeuds proches sur le graphe aient des numéros proches.
- On remplace dans le codage les numéros par la différence du numéro avec le numéro précédent.
- Si le graphe contient des répétitions de sous graphes il va y avoir des répétitions dans le codage. Il est donc intéressant d'utiliser un compresseur classique comme LZW ou le *block sorting*

Compression de graphes avec beaucoup de noeuds

On applique une méthode de compression d'image noir et blanc sans perte (le RLE des FAX par exemple).

On peut en plus faire des échanges de lignes/colonnes afin de faire apparaître de plus grandes zones avec des 0 ou des 1.

On peut utiliser un méthode d'optimisation numérique pour trouver les échanges permettant d'optimiser la taille du fichier après compression (mais c'est long).

Méthode unique

C'est impossible car les méthodes sont spécialisées pour un type d'image. Pour les autres types elles compriment mal ou bien impliquent des pertes. Par exemple pour les images naturelles et les images synthétiques.

Palette de méthode

Les critères sont :

- La méthode doit respecter la qualité demandée.
- Elle doit donner un fichier court.

Algo

Vous comprimez votre image avec toutes les méthodes,

Vous utilisez celle qui respecte le mieux votre critère.

Vous stockez le numéro de la méthode utilisé ainsi que le fichier compressé.

Pour décompresser, on lit le numéro de la méthode et on décompresse.

Séquence

Tester toutes les méthodes cela prend du temps. Pour l'image suivante on utilise d'abord la méthode utilisée à l'image précédente car les changements sont faibles.

Si la taille du fichier compressé augmente trop ou si l'on a comprimer 25 images ont va réessayer toutes les méthodes de compression.

Inconvénients de la méthode globale

Elle ne tient pas compte du fait qu'une image puisse contenir des zones non homogènes. Par exemple les sous-titres dans un film.

Méthode récursive

- On comprime avec la méthode de la question 3 (globale)
- On coupe l'image en 4 et on comprime soit-même chacun des 4 morceaux. C'est donc récursif.
- On choisi des 2 méthodes précédentes celles qui donne le fichier le plus petit.

Inconvénient

Lente.

Il faut trouver des critères permettant de décider s'il faut découper ou non.

Ou bien il faut partir des pixels et remonter vers le haut comme quand on fait un quadtree.

Méthode optimale sur machine à vitesse infinie

Il y a deux manières différentes d'imaginer la méthode optimale.

- Vous voulez un programme compresseur et un programme décompresseur.
- Vous voulez que l'image comprimée soit son propre décompresseur.

La première méthode est conventionnelle. De plus le compresseur/décompresseur peut être utilisé sur des machine réelles car on peut le choisir aussi en fonction de sa vitesse d'exécution. Il y a des problèmes :

- On doit limiter la taille des programmes compresseur et décompresseur car elle peut être gigantesque.
- On veut que les images courantes soient mieux comprimée que les autres. Ce problème est très complexe. Il nécessite la construction d'une base de test énorme.

La deuxième méthode est très courte mais elle suppose d'avoir un bibliothèque de fonctions de base. Par exemple pour éviter de coder la décompression DCT dans l'image comprimer. C'est celle-ci que je vais décrire.

```
Programme compresseur(Image I)
{
   for(N = 1;; N++)
      for(tous les programmes de taille N)
         Si l'exécution du programme donne I alors retourner le programme.
}
```

On suppose :

- On donne une liste explicite de fonctions utilisables. (printf, cos, calcul_dct, ...) en évitant les fonctions dangereuses.
- On arrête l'exécution on cas de bouclage.

Le programme peut être écrit en C ou en shell. On peut supposer qu'implicitement on le comprime avec `gzip`. Si ce n'est pas fait systématiquement, le compresseur peut le découvrir seul et générer par exemple :

```
zcat<<%|sh
ljkhfsdlhfal du shell comprimé par gzip kfhwelfhweriowuhrw
%
```

On perd donc 13 caractères. Mais si le compresseur trouve mieux que `zcat` il l'utilisera.

Méthode optimale et perte

Vous devez avoir une fonction quantifiant les pertes.

L'algorithme est le même que le précédent, il refuse simplement les images ne respectant pas la qualité.