Examen Codage Transmission Compression M2IM 2007-2008

Le 16/1/2008, 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.

1. Quelle quantité d'information est apportée par le lancé d'un dé à 16 faces ?
2. Quelle quantité d'information est apportée par le lancé d'un dé à 4 faces dont les faces sont numérotés 1, 2, 2, 3 ?
3. Décrivez l'architecture d'un algorithme de compression quelconque. On ne vous demande pas de détailler un cas particulier (son, image) mais de donner les grandes lignes de l'algorithme et des traitements.
4. Proposez un codage compact sous forme de *quadtree* d'images bicolores.
5. Donnez les avantages et inconvénients du codage *quadtree* par rapport au codage des FAX compressés dans le cadre de l'utilisation classique des FAX.
6. Proposez une méthode permettant de compresser un codage *quadtree*.
7. Proposez un modèle *quadtree* un peu plus général afin que le codage des images bicolores soit plus compact qu'avec le *quadtree* classique.
8. Donnez des exemples de sons réels qui se compriment mal avec la compression MPEG (mp3).
9. Proposez un algorithme de compression de fichiers issus de scanner/IRM. Indiquez ses avantages et surtout inconvénients.
10. Dans quels cas la compression est néfaste à l'application alors que les données se compriment bien et que l'on ne fait pas d'accès aléatoire ?
11. Proposez une méthode de compression vidéo à débit constant compatible avec MPEG. Indiquez les avantages et inconvénients de votre méthode.
12. Proposez une méthode de compression/transmission permettant de transmettre un fichier vidéo (pas audio) progressivement. À toute les étapes du transfert la vidéo pourra être affichée, plus l'on sera proche de la fin du transfert, plus le rendu sera parfait.
    Donnez les avantages et inconvénients de votre méthode.
13. De quelle manière les méthodes de compression d'images doivent traiter des images réelles auxquelles ont été ajouté un canal transparence (pour faire le détourage par exemple dans le cas du Web). N'oubliez pas de justifier.
14. Que ce passe-t-il si l'on modifie l'algorithme du *block sorting* pour qu'il travaille avec des caractères codés sur deux octets au lieu d'un ?
15. Que ce passe-t-il si l'on modifie l'algorithme du *block sorting* pour qu'il travaille au niveau du bit ?
16. Des collégiens rendent une rédaction. Vous pouvez utiliser `gzip` pour vous aider à noter les rédactions. Comment ?

Une correction

1. **Quelle quantité d'information est apportée par le lancé d'un dé à 16 faces ?**

   Les apparitions de face sont équiprobables et indépendants. L'information moyenne apportée par un lancé est

   $$\sum_{1}^{16} -\frac{1}{16} log_2 \frac{1}{16} \; = \; -log_2 \frac{1}{16} \; = \; log_2 16 \; = \; 4 bits$$

   Pour calculer rapidement un log de tête, c'est simple, la valeur du log est le nombre de chiffres nécessaires pour exprimer le nombre dans la base du log. $log_{10} 1000 = 3$ il faut 3 chiffres pour écrire les nombres jusqu'à 1000.

2. **Quelle quantité d'information est apportée par le lancé d'un dé à 4 faces dont les faces sont numérotés 1, 2, 2, 3 ?**

   On applique la même formule mais les événements ne sont plus équiprobables.

   $$-\frac{1}{4} log_2 \frac{1}{4} \; - \; \frac{2}{4} log_2 \frac{2}{4} \; - \frac{1}{4} log_2 \frac{1}{4} \; = \; \frac{1}{2} \; + \; \frac{2}{4} \; + \; \frac{1}{2} \; = 1.5 bits$$

3. **Décrivez l'architecture d'un algorithme de compression quelconque. On ne vous demande pas de détailler un cas particulier (son, image) mais de donner les grandes lignes de l'algorithme et des traitements.**

   Élimination des informations inutiles (bruits, trop grand précision, redondance, ...).

   Rendre les informations indépendantes les unes des autres (espace couleur, codage relatif, ...)

   Coder la différence par rapport à un modèle.

   Éliminer les répétitions (RLE).

   Équilibrer les probabilités (Huffman, codage arithmétique, ...)

4. **Proposez un codage compact sous forme de *quadtree* d'images bicolores.**

   Pour coder un *quadtree* il faut trois symboles. Dans un quadtree, il y a 3 fois moins (environ) de noeuds que de feuilles, le noeud doit dont être codé sur 2 bits car il est impossible qu'il soit l'événement le plus probable. On codera donc la couleur la plus probable sur 1 bits. On code donc 11 pour un noeud et 0 et 10 pour les feuilles d'une couleur ou de l'autre.

   Une image 8x8 avec un pixel noir dans un coin : 11 11 11 10 0 0 0 0 0 0 0 0 0
   Ceci occupe 17 bits au lieu de 64.

5. **Donnez les avantages et inconvénients du codage *quadtree* par rapport au codage des FAX compressés dans le cadre de l'utilisation classique des FAX.**

   Le *quadtree* permet l'affichage progressif, il comprime bien les grandes zones uniformes (les FAX aussi, 3 bits par ligne dans la zone) mais il comprime très mal les zones ou il y a un mélange de couleurs, donc le texte. De plus il faut décomprimer toute l'image pour l'imprimer.

6. **Proposez une méthode permettant de compresser un codage *quadtree*.**

   On peut appliquer un algorithme à base de dictionnaire pour éliminer les redondances du code (elles sont nombreuses) par exemple LZW ou le *blocksorting* en les appliquant bien à l'événement et non à un suite d'octets.

Le RLE n'est pas forcément une bonne idée car il ne devrait pas y avoir de grandes suites d'événements identiques (7 codes couleurs identiques successifs sont rigoureusement impossible).

Un bonne idée par contre est de remplacer les arbres contenant 4 feuilles par un code spécial. En effet le codage est de la forme 11 A B C D mais A B C D ne peuvent être identiques par définition. Ensuite on applique LZW.

7. **Proposez un modèle *quadtree* un peu plus général afin que le codage des images bicolores soit plus compact qu'avec le *quadtree* classique.**

   La version classique coupe en quatre au milieu sans se soucier du contenu de l'image.

   On peut donc peut-être améliorer les choses en indiquant comment le découpage est fait. L'exemple de la question 4 deviendrait : 11 000 000 10 0 0 0
   Les deux groupes 000 indiquant l'endroit ou l'on découpe. Le découpage n'a plus qu'un niveau de profondeur.

8. **Donnez des exemples de sons réels qui se compriment mal avec la compression MPEG (mp3).**

   Tout les son *aléatoires* donc les *bruits* : souffle électronique (mettez votre chaine HiFi à fond sans entrée), cascades, déferlement de vagues, bruit de moteur, ...

   Les bruits ont normalement un spectre plat (ou gaussien) donc sans pics. La DCT ne permet donc pas de retrouver un modèle.

   Le fréquence absolue du son n'a aucune importance, ce qui compte est sa fréquence par rapport aux autres son qu'elle peut masquer créant ainsi des 0.

9. **Proposez un algorithme de compression de fichiers issus de scanner/IRM. Indiquez ses avantages et surtout inconvénients.**

   DCT 3D, Ondelette 3D.

   Les ondelettes nécessitent un parcours globale pour être intéressante. La DCT peut travailler par petit cubes.

   Les ondelettes nécessitent moins de temps de calcul que la DCT à taille de bloc égales.

   Il est préférable d'avoir des gros blocs car la taille des gros blocs vides est plus grande en 3D.

   Remarque amusante, le volume de la sphère unité dans R^n tend vers 0 quand n tend vers l'infini.

10. **Dans quels cas la compression est néfaste à l'application alors que les données se compriment bien et que l'on ne fait pas d'accès aléatoire ?**

    Quand les données sont traitées numériquement et qu'il y a des pertes.

    Quand la compression implique une latence de transmission qui nuit à l'application. (Cette latence existe **même** si la compression n'utilise pas du tout de temps CPU, en effet le compresseur a besoin d'accumuler des données avant d'émettre une information compressée.

    Quand des bits sont perdus lors de la transmission.

11. **Proposez une méthode de compression vidéo à débit constant compatible avec MPEG. Indiquez les avantages et inconvénients de votre méthode.**

    On comprime l'image en ce fixant une taille limite. Si on la dépasse, on recomprime en diminuant la qualité.

C'est évidemment très lent et pas fait pour le temps réel.

Et si l'on utilise des images bidirectionnelles, on doit choisir plusieurs facteurs de qualité pour optimiser le groupe.

12. **Proposez une méthode de compression/transmission permettant de transmettre un fichier vidéo (pas audio) progressivement. À toute les étapes du transfert la vidéo pourra être affichée, plus l'on sera proche de la fin du transfert, plus le rendu sera parfait.**
    Donnez les avantages et inconvénients de votre méthode.

    Idée de base : ondelettes 3D.

    Avantage : compression optimum car tient compte de nombreuses images. En transférant 1/8 du fichier on aura un vidéo 2*+ courte (15 images secondes) avec 4*- de pixels. Avec 1/64 du fichier on aura un vidéo 4*+ courte (6 images secondes) avec 16*- de pixels. La vidéo n'aura pas d'artefact car les ondelettes font naturellement l'antialiasing.

    Inconvénient : il faut utiliser beaucoup de mémoire.

13. **De quelle manière les méthodes de compression d'images doivent traiter des images réelles auxquelles ont été ajouté un canal transparence (pour faire le détourage par exemple dans le cas du Web). N'oubliez pas de justifier.**

    D'abord, diminuer la précision de codage des pixels en fonction de la transparence. À l'extrême, on a besoin d'aucun bit de stockage pour stocker un pixel transparent. Un pixel avec une transparence de 0.5 n'a pas besoin de ses derniers bits pour les valeurs car il disparaîtra dans la division par 2. On peut donc ajuster la qualité de compression en fonction de la transparence.

    Il faut compresser séparément le canal transparence. Ce n'est pas un canal bruité, une compression de type LZW ou RLE devrait bien fonctionner.

    Les canaux normaux sont compressés avec une méthode classique.

14. **Que ce passe-t-il si l'on modifie l'algorithme du *block sorting* pour qu'il travaille avec des caractères codés sur deux octets au lieu d'un ?**

    Il devrait un petit peu mieux compresser sur les fichiers qui utilisent des caractères long.

    Mais il sera inutilisable tel quel car l'algorithme du *move to front* sera extrêmement lent si de très nombreux caractères différents sont utilisés.

15. **Que ce passe-t-il si l'on modifie l'algorithme du *block sorting* pour qu'il travaille au niveau du bit ?**

    En faisant cela, on s'affranchit du codage de la donnée à transmettre. La compression devrait donc mieux fonctionner sur les fichiers "*binaire*" non alignés sur l'octet. Mais moins bien fonctionner sur les fichiers textes.

    Le *move to front* n'a plus d'utilité, il faut directement faire un RLE avec nombre de 0, nombre de 1, ...

    Le tri ne sera pas plus long car l'ordre du tri par octet est le même que celui du tri par bits. D'ailleurs, on peut directement faire un tri sur des `long` cela donnera le même résultat.

16. **Des collégiens rendent une rédaction. Vous pouvez utiliser `gzip` pour vous aider à noter les rédactions. Comment ?**

Le taux de compression donne une indication sur la quantité de vocabulaire utilisé. Sur les répétitions de structures dans les phrases.

Évidemment cela ne marchera pas du tout pour la poésie.

Si l'on comprime tous les appariements de copies possible, les paires avec le plus fort taux de compression seront celles qui contiennent 2 copies qui se ressemblent le plus.