Examen Codage Transmission Compression M2IM 2006-2007

Le 9/1/2007, 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. Donnez l'architecture **générale** de toutes les méthodes de compression.
2. Peut-on employer la méthode JBIG pour comprimer du texte ?
3. Citez 3 méthodes d'égalisation de probabilités.
4. Quelle quantité d'information est apportée par le lancé d'un dé ?
5. La transformée DCT est-elle sans perte ?
6. Comment programmeriez-vous un système de fichier compressé ?
7. Que ce passe-t-il si vous compressez un fichier déjà compressé ?
8. Les compresseurs de texte compriment-ils efficacement de l'UTF8 ?
9. Les compresseurs de texte compriment-ils efficacement du PBM ?
10. Donnez différentes méthodes permettant de faire des pages web plus rapides à transférer.
11. Dans quels cas le codage arithmétique est-il utile ?
12. Comment comprimer des matrices de voxels ?
13. Pourquoi les compresseurs de texte fonctionnent mal sur les photos ?
14. Vous avez un fichier de log d'un serveur Web (date/client/url), comment le comprimer efficacement sans perte d'information ?
15. Les langues humaines (français, anglais, ...) sont-elles comprimées ?
16. Quelle format d'image faut-il utiliser pour faire des boutons sur une page web ?
17. Dans quels cas faut-il utiliser SVG ?
18. Quels traitements doit-on faire sur une page HTML si l'on veut la diffuser comprimée ?
19. Proposez un codage des octrees (voxels binaires) optimum.
20. Que spécifient les normes de compression ?
21. Un fichier PDF de quelques centaines d'octets peut-il afficher des fractals détaillés ?
22. *Qui* utilise la notation redondante des entiers ?
23. Le RLE est-il utilisé pour autre chose que des images ?
24. Le Block-Sorting est-il efficace sur les images synthétiques ?
25. Pour les méthodes vues en cours, indiquez si le compresseur pourra être amélioré sans toucher au décompresseur et comment.
26. Quels sont les méthodes de compression vues en cours qui supportent la transmission *temps réel*.

Une correction.

1. Une pipeline pouvant contenir les étapes suivantes :
   - Décorrélation des événements (RVB devient HLS).
   - Élimination des informations inutiles de la donnée initiale.
   - Utilisation d'un modèle (DCT par exemple ou LZW).
   - Élimination des informations inutiles du modèle.
   - Codage du modèle.
   - Équilibrage des probabilités sur le codage final.

2. Une utilisation directe comprimera mal le texte. Par contre on peut concevoir le texte comme un signal 1d avec comme couleur le code ascii du caractère. En utilisant les 3 caractères précédents, on utilise déjà une table avec 2^{24} probabilités. Le caractère sera codé (grace au codage arithmétique) sur moins d'un bit. Cette méthode peu donner théoriquement un bon taux de compression pour le texte mais la vitesse sera très lente car il faudra travailler avec de nombreuses probabilités.

3. Shannon-Fano, Huffman, codage arithmétique. Le Move to Front et RLE n'égalisent pas les probabilités car le flux de sortie ne contient pas autant de bits à 0 ou à 1.

4. Chaque face apporte la même quantité d'information car le dé n'est pas pipé. Les lancés sont indépendants les uns des autres. Donc c'est $-log_2 \frac{1}{6} \approx 2.58 bits$

5. Mathématiquement oui, avec des flottants presque, et avec des entiers il y a pas mal d'erreur.

6. On veux une solution transparente pour l'utilisateur.

   Choix 1 : Le système de fichier s'occupe seulement de faire les décompressions au vol des fichiers (cela peu être long si l'on veut la fin d'un gros fichier, il faut donc faire une compression supportant l'accès aléatoire). Un démon parcours régulièrement le disque (quand la machine n'est pas occupée) pour comprimer au mieux les fichiers non comprimés.

   Choix 2 : Le système de fichier comprime et décomprime des blocs élémentaires du disque (4Ko à 16Ko). La compression/décompression est transparente, l'accès aléatoire est sans problème. Le taux de compression est faible car les fichiers petits. La méthode de compression ne connais pas le type du fichier.

   Bonus : Cherche les fichiers identiques dans le système pour ne pas les stocker plusieurs fois.

   Bonus : Si le système contient de nombreux petits fichiers (quelques octets) il est intéressant de les regrouper ensembles, surtout s'il sont dans le même répertoire avec les même attributs (propriétaire, mode, ...)

7. Avec la même méthode : le fichier grossis. Avec une méthode différente : il y aura une faible variation de taille. Sinon, si on comprime des signaux, on peut réduire la taille du fichier en le décomprimant et en recomprimant avec une qualité inférieure.

8. L'UTF8 code un caractère sur un nombre entier d'octets il est donc compatible avec les méthodes de compression de texte fonctionnant sur les textes. D'autre part le compresseur de texte va pouvoir enlever la redondance de l'UTF8, le taux de compression sera donc meilleur que pour un texte ISO8859-1. Par contre, pour un même fichier initial, la taille du fichier compressé sera la même que pour les 2 codage possibles du fichier initial.

9. Sur le PBM ASCII : super compression sans problème. Ce n'est pas optimum car on ne tient pas compte du 2D.

   Sur le PBM RAW : Les mêmes motifs décalés d'un pixel ne seront pas vus comme identiques par le compresseur de texte. Il y aura donc un moins bonne compression.

10. Il faut :

    - Extraire le CSS et le JavaScript de la page.
    - Éliminer la redondance grace au CSS et JavaScript.
    - Enlever ce qui est inutile du HTML : espaces, commentaires, retour à la ligne.
    - Normaliser les TAG (tous en minuscule) et l'ordre des attributs pour optimiser la compression.
    - Compresser la page avec un compresseur standard.

    La page sera lisible avec un navigateur normal, mais elle pourra être de 10 a 100 fois plus petit (si si).

11. Quand on doit coder une suite d'événements extrêmement probables. Quand on veut un équilibrage optimum.

12. Si les voxels ont des valeurs binaires, on peut imaginer un JBIG en 3 dimensions (au minimum les 13 voxels précédents), des octree grossier avec des feuilles en JBIG. Si les valeurs sont flottantes on peut appliquer les ondelettes ou la DCT en 3D (Cela ne change pas la complexité des algorithmes).

13. Une photo contient du bruit, le compresseur de texte ne trouvera donc pas de motifs à comprimer. Il ne pourra qu'éliminer du codage les couleurs non utilisées.

14. Voici un codage possible du fichier de log, il permet de faire la compression au vol.

    - Coder la date par rapport à la date précédente en faisant la différences des deux dates converties en secondes.
    - Coder le client comme : **0 + client** si c'est un nouveau client ou un indice (le plus petit possible) permettant de retrouver le client parmi les clients précédents. Si le client charge plusieurs fichier à la suite, on va coder plusieurs fois 1.
    - Découper l'URL en deux : chemin et paramètres. Coder le chemin et les paramètres comme pour le client. Cela est améliorable car ces événements sont dépendants de la dernière URL chargée par le client.

    Pour la compression chacune des 4 suites d'information est compressée via un algorithme de compression classique.

15. Oui, car;

    - Les mots les plus utilisés sont les plus courts.
    - Les préfixes et suffixes remplacent des constructions.
    - Les constructions grammaticales qui évitent de répéter des morceaux de phrases. **Elles** permettent de raccourcir les textes.
    - Le fait de nommer les choses évite de les décrire chaque fois que l'on en parle. C'est pour cela qu'il y a autant de mots.

- Les acronymes (UCBL), et raccourcis c.a.d **s'il** va au **ciné**.
- En français le xsy c'est souvent transformé en $\hat{x}y$ pour enlever le s.
- En anglais on élimine les redondances : *The dogs / The dog* ou We eat / They eat
- Certaines langues alphabétiques n'écrivent que les consonnes, c'est le lecteur qui doit retrouver les voyelles.

Les amateurs de SMS sont en train de la comprimer violemment...

16. Les formats standards sont GIF, JPG, PNG. Si le bouton est une photos alors on prendra JPG, sinon on prendra PNG car il est sans perte et comprime mieux que GIF (et il y a la transparence).

17. On utilise SVG dans le cas ou l'image est vectorielle et que l'on veut être indépendant de la résolution afin de toujours avoir la meilleur qualité. Si l'image est utilisée à une résolution fixe, alors on utilisera du SVG si le fichier GIF/PNG est plus gros que le fichier SVG. Autre atout du SVG, il peut être généré par du Javascript et donc être intéractif.

18. Déjà indiqué dans la question 10.

19. Un bit à 0 indique un noeud, il est suivi de descriptions. Un bit à 1 indique une feuille et est suivi de la valeur de la feuille. Si l'octree représente de vrais données volumiques, le nombre de feuilles est proportionnel à la surface des objets représentés. Si l'octree à une résolution maximale connue, alors le bit indiquant que c'est une feuille de résolution maximale n'est pas nécessaire, en effet ce n'est pas possible que cela soit un noeud.

20. Les normes de compression spécifient le format des données compressées et comment les décompresser. Cela laisse le champs libre pour différents type de compresseurs qui produiront tous des fichiers compatibles.

21. Le PDF n'est pas un langage de programmation, on ne peut donc pas décrire la formule récursive permettant d'afficher le fractal. En PostScript, on pourrait. Le seul intérêt de PDF est d'avoir un interpréteur simple car il n'interprète pas un langage mais seulement des données.

22. Les calculs arithmétiques dans les processeurs ainsi que tous les calculs manipulant des nombres très grands, par exemple le **codage arithmétique**.

23. Dans tous les cas ou des séquences d'événements identiques apparaissent : après la DCT, après le *move to front*, après un prédiction de signal peu bruité, ...

24. Dans une image synthétique il y a de très nombreuses répétitions de motifs. Les motifs contiennent peu de couleurs. Donc le block-sorting sera très efficace, même si l'image est complexe et contient du texte. Pas aussi efficace que du PNG car il ne cherche pas des motifs en 2 dimensions.

25. Voici les méthodes :

    - Shanon-Fano : On peut le transformer en Huffman sans toucher au décompresseur
    - Huffman : Si le décompresseur supporte un événement permettant de retransmettre l'arbre. Le boulot du décompresseur sera de trouver à quel moment changer d'arbre.
    - LZ77 : Choix de la fenêtre en fonction du fichier.

- LZ78, LZW : Ne pas envoyer l'événement optimum mais un moins bon à court terme car à moyen terme on y gagne (cela ralentie considérablement la compression car on doit analyser le futur)
- DCT : Pas d'idée.
- JPEG/MPEG : Modifier l'image globalement pour que la compression se fasse mieux sans que cela se ressente visuellement.
- MPEG : Rechercher les blocs ressemblants plus loin.
- BlockSorting : Trouver le meilleur découpage par blocs du fichier.

26. Temps réel peut vouloir dire 2 choses :

    - Les données sont comprimées au vol. Ceci veut dire que l'on ne peut pas avoir plusieurs centaines d'octet de retard. Donc que le compresseur n'as pas un gros buffer de texte à compresser. Ceci exclut le RLE, LZ77 et le block sorting.
    - Il n'y a pas de demande de retransmission. Ceci exclu toutes les méthodes dynamiques qui s'appuient sur toutes les données précédentes. LZ78, LZW, MPEG (avec le motif IPPPPPPPPPPPPPPPPP...)