info223 : Science informatique
TD 4 : compression

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Partie 1 : Compression par bloc, algorithme de Huffman

Question 1. Construisez un code préfixe optimal en utilisant l'algorithme de Huffman pour la répartition de lettres suivante :
- `a` :2000, `b` :2000, `c` :3000, `d` :3000, `e` :3000, `f` :5000,

Même question pour la répartition
- `a` : 500, `b` : 900, `c` : 300, `d` : 500, `e` : 100, `f` : 300, `g` : 400, `h` : 400, `i` : 400, `j` : 800, `k` : 400, `l` : 600, `m` : 1500.

Estimez, dans les deux cas, la taille du texte compressé.

Question 2. Que se passe-t'il si on applique l'algorithme de Huffman pour coder une suite de `A` et `B` où 99% des caractères sont des `A`.

Quelles améliorations proposez vous ?

Partie 2 : compression par dictionnaire, algorithme LZ78

Question 1. Rappelez le fonctionnement de l'algorithme de compression par dictionnaire LZ78.

Question 2. Détaillez le fonctionnement de l'algorithme de compression LZ78 sur les chaines
- `lalala`
- `abracadabra`
- `aaaaaaa`, `aaaaaaaa` et `aaaaaaaaa`

Donnez, en séparant bien les deux, le résultat ainsi que le dictionnaire construit.

Donnez, à chaque fois, la taille du résultat compressé (en octets) si on suppose que chaque nombre est codé sur 1 octet.

Question 3. Détaillez l'algorithme de décompression LZ78 sur les suites d'octets suivantes :
- `0x00-O 0x00-L 0x00-E 0x01-L 0x03`
- `0x00-R 0x00-A 0x00-P 0x00-L 0x02-P 0x04-A`

Chaque nombre est donné en hexadécimal (1 octet = 2 chiffres) et chaque lettre est donnée par elle même (1 octet = 1 code ASCII).

Question 4. Essayez de donner une approximation de la taille de la compression de `aa...aa` (n fois) si on suppose qu'on stocke toujours les numéros de mots du dictionnaire sur 1 octet.

Partie 3 : pour aller plus loin

Question 1. La compression "RLE" (Run Length Encoding) consiste à remplacer `cccc` par `c`$\underline{4}$ où le nombre $\underline{4}$ est codé sur 1 octet.

Plus généralement, on remplace `c...c` (n fois) par `c`$\underline{n}$, où le nombre n est codé sur un octet.
- Comment est compressé `aa...aa`, où il y a exactement 500 `a`?
- Comment est compressé `a`?
- Comment est compressé `abab...abab` où il y a exactement 500 `ab`?

Question 2. On essaie d'améliorer cet algorithme pour ne pas perdre de place sur les caractères seuls : on remplace

- un `c` tout seul par `c`,
- une répétition `c...c` (n fois, avec $n > 1$) par `c`$\underline{n}$, où n est codé sur un octet.

Ainsi, la chaine `abab...ab` est "compressée" par elle même.

Quel problème cela pose-t'il ?

Question 3. Pour corriger le problème de la question précédente, on peut remplacer

- un `c` tout seul par `c`,
- une répétition `c...c` (n fois, avec $n > 1$) par `cc`$\underline{n}$, où n est codé sur un octet.

Donnez un exemple de chaine dont la taille augmentera avec cette méthode de compression.

Question 4. Donnez un exemple de (grande) chaine dont la taille augmente lorsqu'on la compresse avec l'algorithme LZ78.

Question 5. Donnez un exemple de (grande) chaine dont la taille augmente lorsqu'on la compresse avec l'algorithme de Huffman.

Question 6. Justifiez l'affirmation suivante : "Pour tous les programmes de compression existants (`gzip`, `bzip2`, `zip` ou autre), il existe des fichiers dont la taille augmente si on essaie de les compresser."