Travaux Pratiques d'Informatique n°4
DEUG MIAS / S4 option info

Compression par déplacement en tête de liste

Nous décrivons ici une méthode de compression de données (adaptée aux textes) qui utilise une liste chaînée de mots. La position d'un mot dans la liste est un entier strictement positif (la position du mot en tête de liste est donc 1).

L'algorithme de compression est le suivant :

Pour chaque mot du texte

- Rechercher le mot dans la liste
- Si le mot existe dans la liste, on le code à l'aide de sa position dans la liste, puis on le déplace en tête de liste.
- Si le mot n'est pas dans la liste, on le code par l'entier `0` suivi du mot en clair, et on le rajoute en fin de liste.

Exemple (on ignore ici les séparateurs et la ponctuation) :

On considère le texte suivant :
deux et deux quatre
quatre et quatre huit
huit et huit seize
répétez dit le maître
On code *deux* par `0deux`. La liste est *deux*.
On code *et* par `0et`. La liste devient *deux*→*et*.
On code *deux* par `1` car il est en première position de la liste. La liste n'est pas transformée car *deux* est déjà en tête de liste.
On code *quatre* par `0quatre` puisqu'il n'est pas dans la liste. La liste devient *deux*→*et*→*quatre*.
On code le *quatre* suivant par `3`. La liste devient *quatre*→*deux*→*et*.
On code le *et* suivant par `3`. La liste devient *et*→*quatre*→*deux*.
On code le *quatre* suivant par `2`. La liste devient *quatre*→*et*→*deux*.
Le *huit* est codé par `0huit` et est ajouté en fin de liste qui devient donc *quatre*→*et*→*deux*→*huit*.
Les deux premiers vers sont donc codés par : `0deux 0et 1 0quatre 3 3 2 0huit`.

- Dans l'exemple donné, les séparations et ponctuations ont été ignorées. On considère qu'un texte est formé d'une alternance de mots courants formés de caractères alphabétiques et de mots de ponctuations formés de caractères non alphabétiques.
  On peut intégrer les mots de ponctuation dans la liste précédente ou, dans une deuxième version, utiliser deux listes. Le compresseur et le décompresseur utilisent donc deux listes.
- Remarquer que l'utilisation du mot vide permet de traiter le cas où le fichier contient des mots de taille supérieure à celle du buffer.

  Dans un premier temps, on mettra en place la méthode en lisant sur l'entrée standard et en écrivant sur la sortie les entiers et les mots en clair.

Écriture des entiers sur un nombre variable de bits

Dans la méthode décrite ci-dessus, tous les entiers sont écrits sur quatre octets, ce qui est une perte évidente puisque les mots les plus fréquents du texte seront situés proche de la tête de liste et auront des positions petites dont l'écriture binaire ne comporte que peu de bits significatifs.
On propose de tester deux méthodes d'écriture des entiers.

Écriture sur un nombre variable d'octets

On découpe l'écriture binaire par paquets de 7 bits, le huitième est utilisé pour indiquer si l'écriture se poursuit sur l'octet suivant.
Ainsi $5 = (101)_2$ s'écrit sur un octet `00000101`
et $707 = (1011000011)_2$ s'écrit sur deux octets `10000101 01000011`

Écriture sur un nombre variable de bits

On peut écrire chaque entier n sur un nombre variable de bits, si on indique d'abord le nombre de bits significatifs de l'écriture de n. Cela peut se faire en unaire, c'est à dire par une suite de 1 de longueur égale au nombre de bits utilisés, suivi d'un 0, utilisé comme séparateur.

Le nombre $5 = (101)_2$ est donc transcrit par `1110101` (`111` au début indique que trois bits sont nécessaires, suivi de `0` comme séparateur suivi de `101`) et 128 est transcrit par `11111111010000000`.

De même, la suite de bits `11110 1010 10 0` indique les deux entiers 10 et 0. `11110` indique un entier écrit sur 4 bits : `1010`=10 et `10` indique un entier écrit sur 1 bit : `0`=0.

Pour aller plus loin...

Une variation consiste à ne pas remonter le mot directement en tête, mais à l'échanger avec son prédécesseur. De cette façon un mot peu fréquent ne repoussera pas les mots fréquents. Tester l'efficacité de cette méthode.