TP5

TP compression de données

Partie I ; Compression R.L.E

Exercice 1 : *Nombre d'occurrences*

On prend l'exemple d'un fichier texte dont on calcule tout d'abord le nombre d'occurrences de chaque caractère dans le fichier.

Exemple : soit le fichier F1.txt dont le contenu est : ''aaaabra cada bra bb''

Caractère :	**' '**	**a**	**b**	**r**	**c**	**d**
Fréquence	**3**	**8**	**4**	**2**	**1**	**1**

Figure 1 : Fréquence des caractères du fichier F1.txt

RLE (Run Length Encoding)

- But : cet algorithme élide les répétitions successives de caractères.

De nombreuses variantes de ce principe peuvent être utilisées.

Exercice 2 : (variante 1)

Algorithme de compression :

- Remplacement de la répétition de caractères par:
  - **un octet** : contenant le nombre d'occurrence
  - **un octet** : le **caractère répété**

Algorithme de décompression :

Durant la lecture du fichier compressé, on effectue l'opération inverse de la compression tout en supprimant la zone contenant N.

ex: **AAAAAAAAAARRRRRRRROLLLLLBBBBBUUTTTTTT (taille = 36 caractères)**

Après compression : **10A8RO8L5BUU6T (taille = 13 caractères)**

Taux de compression :

(taille fichier initiale - taille fichier compressé)/ taille fichier initiale

Le taux de compression est ainsi de (36-13)/36 soit environ 63,88%.

Le taux de compression peut être négatif s'il n'y a pas ou peu de répétition :

➔ compression très couteuse

Exercice 2 : (variante 2)

Algorithme de compression – caractère d'échappement

Règle décompression RLE: compresser lorsque cela est nécessaire et de laisser la chaîne telle quelle lorsque la compression induit un gaspillage.

Seuil de répétition : lorsque **trois éléments ou plus** se répètent alors la méthode de compression RLE est utilisée

- Sinon un caractère de contrôle **(@)** est inséré, suivi du nombre d'éléments de la chaîne non compressée puis de cette dernière.

Remplacement de l'itération de caractères par:

1. un caractère spécial **(@)** indiquant une compression
2. le nombre de fois où le caractère est répété
3. le caractère répété

On choisit comme caractère spécial : **@** et comme seuil de répétition : 3

Après compression :**@10A@8RO@4L@5BUU@6T** ➔ gain : 11 caractères soit 38%

Exercice 3 : (variante 3) : cas de suite bits

3- De nombreuses variantes de ce principe sont utilisées. Par exemple, si la suite considérée est une suite de bits, on peut s'intéresser aux suites de 0 consécutifs situés entre deux 1.

La suite **00000100110000000101011** peut être représentée par la suite 52071100.

Travail à faire.

Ecrire un programme qui réalise la compression-décompression.

Ecrire un programme qui accepte **(argc et argv)**

- le nom de fichier (argv et argc) F1.**ini** comme suffixe (fichier initial)
- caractère spécial de la suite.
- Le fichier compressé porte le nom du fichier initial avec F1**.cmp** comme extension.

Ecrire un programme qui accepte **(argc et argv) :**

- Le fichier compressé **(F1.cmp**)
- Décompresser le fichier **F1.cmp** et produire le fichier **F1.dec** (décompressé)
- Donner le taux de décompression.

Seconde partie : Réalisation de l'arbre de Huffman

Pour chaque caractère (ou noeud de l'arbre), nous allons créer une structure commune qui nous permettra de créer l'arbre et de créer le code. Nous avons appelé cette structure « noeud » mais en réalité elle sera aussi utilisée pour les feuilles de l'arbre.

La structure est donc définie comme suit :

```
struct nœud {
unsigned char c;    /* Caractère initial*/
int occurence;   /* Nombre d'occurrences dans le fichier */
int code; /* Codage dans l'arbre */
int bits; /* Nombre de bits sur lesquels est codée le caractère */
struct noeud *gauche, *droite; /* Lien vers les noeuds suivants */
. };
```