# COMPRESSION DE DONNEES

## 1) INTRODUCTION

La compression de données est l'opération qui consiste à réduire la taille des données et ce, afin de pouvoir stocker ou faire circuler le plus de données (informations). Autrement dit, la compression de données est l'opération qui consiste à réduire le nombre de bits nécessaires permettant de stocker des données.

La compression de données est utilisée dans différentes activités telles l'archivage des données, les téléchargements sur Internet, la bande passante sur Internet dont le débit est limité et où, tout dépassement fait l'objet de frais supplémentaires pour le client, la communication par satellite comme, par exemples, le téléphone portable, la télévision par satellite, etc.

Il existe deux catégories d'algorithmes de compression :

- ✓ Compression de données sans perte ;
- ✓ Compression de données avec pertes.

La compression de données sans perte est une compression qui garantie la restitution intégrale des données après décompression. Elle concerne notamment les fichiers textuels. Elle permet de détecter les répétitions dans les données pour les factoriser par la suite.

Comme exemples de logiciels utilisant la compression sans perte, on peut citer 7-Zip et Winrar.

Les algorithmes les plus connus et usités de compression sans perte sont Huffman, RLE et LZW.

La compression de données avec pertes est une compression où les données obtenues après décompression sont voisines des données initiales. Elle concerne notamment l'audio, la vidéo et l'imagerie. La modification des données après décompression est non perceptible par un Humain.

La compression de données avec pertes s'appuie sur des techniques relevant, par exemple, du traitement du signal.

Les algorithmes les plus connus et usités de compression avec pertes sont JPEG, MPEG, Ondelettes et Fractales.

## 2) COMPRESSION SANS PERTE

### 2.1) ALGORITHME RLE (RUN LENGTH ENCODING)

L'algorithme RLE est l'un des algorithmes de compression les plus simples.

Il consiste à remplacer dans toute chaîne tout bit ou tout caractère qui se répète successivement au moins n fois par son nombre d'occurrences suivi de lui-même. On notera que n est donné par l'utilisateur.

Par exemple, pour n=3, le codage de AAAAABBCCCCCCC donne 5ABB7C qui est une chaîne plus courte de 6 caractères, soit un gain de 6/14=43%.

Par exemple, pour n=2, le codage de AAAAABBCCCCCCCAA donne 5ABB7C2A qui est une chaîne plus courte.

Par exemple, pour n=2, le codage de AAAAABBCCCCCCC AA donne 5ABB7C1 2A qui est une chaîne plus courte.

L'encodage RLE est principalement utilisé pour comprimer des images car dans une image, on trouve souvent des pixels ou des couleurs contigus qui se répètent.

Avec l'encodage RLE, le taux de compression est de l'ordre de 40%, ce qui est assez faible.

**Remarques**

- L'encodage RLE n'est pas approprié si la chaîne ne comporte pas bon nombre de caractères qui se répète successivement n fois, avec n assez important.

-Lorsque le nombre de répétition est trop petit (inférieur à n), l'algorithme RLE utilise un caractère spécial non existant dans la chaîne à coder et qui permet de savoir si une compression a eu lieu ou pas.

Par exemple, pour n=3, et en choisissant pour caractère spécial le caractère @,

le codage de AAAAABBCCCCCCCAA

donne @5ABB@7CAA qui est une chaîne plus courte.

Pour décompresser, on utilise l'opération inverse de la compression RLE tout en supprimant le caractère spécial.

- Pour les images, l'encodage RLE est principalement utilisé pour les images au format BMP et PCX ainsi que pour les images en noir et blanc.

## 2.2) ALGORITHME HUFFMAN

L'algorithme Huffman consiste à remplacer les caractères les plus fréquents par des codes courts et les caractères les moins fréquents par des codes longs.

L'algorithme Huffman parcourt les données d'un fichier et opère comme suit.

- ✓ On détermine le nombre d'occurrences de chaque caractère ;
- ✓ On ordonne les caractères selon leur ordre décroissant d'occurrences. En cas d'égalité on ordonne dans l'ordre lexicographique ;
- ✓ On construit un arbre binaire permettant d'obtenir le code binaire de chaque caractère.

La construction de l'arbre binaire se fait comme suit.

- ✓ Chaque caractère du fichier est une feuille de l'arbre ;
- ✓ On relie deux à deux les caractères avec le plus petit nombre d'occurrences et on crée un nouveau nœud auquel on affecte la somme des nombres d'occurrences des deux caractères ;
- ✓ On réitère l'étape précédente en prenant en compte le nœud nouvellement créé et en ôtant les deux caractères précédemment sélectionnés et ce, jusqu'il n'y ait plus de caractère à relier ;
- ✓ On affecte 0 à chaque branche de gauche, et on affecte 1 à chaque branche de droite ;
- ✓ La concaténation des étiquettes des branches d'un chemin de l'arbre depuis la racine jusqu'à une feuille donne le code binaire de la feuille.

**Remarque**

- Le nombre d'occurrences de la racine de l'arbre de Huffman est égal au nombre de caractères.

**Exemple**

Supposons que l'on veuille coder la chaine JAVA. A cette fin,

- ✓ On détermine le nombre d'occurrences de chaque caractère ;
- ✓ On ordonne les caractères selon leur ordre décroissant d'occurrences. En cas d'égalité on ordonne dans l'ordre lexicographique.

| A | J | V |
|---|---|---|
| 2 | 1 | 1 |

- ✓ On relie deux à deux les caractères J et V qui ont la plus petite occurrence et on crée un nouveau nœud N1 auquel on affecte la somme des occurrences de J et de V ;
- ✓ On utilise une des deux cases J ou V pour stocker le contenu du nœud N1, et on supprime l'autre case.

| A | N1 |
|---|---|
| 2 | 2 |

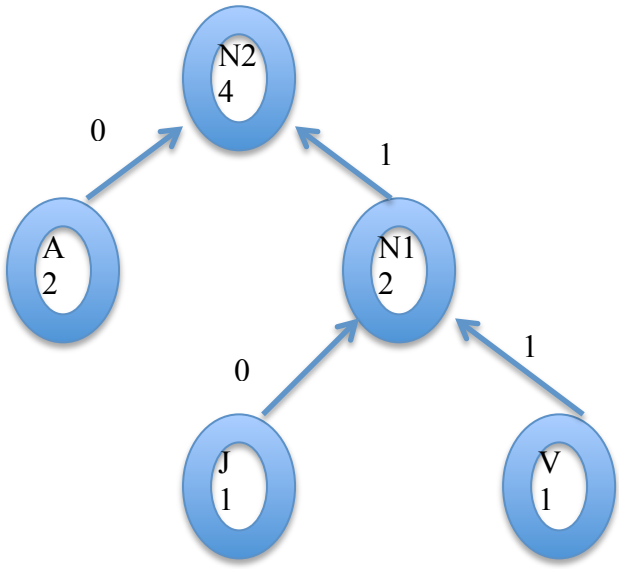
Comme le nombre de case restant n'est pas de 1, on réitère le processus.

- ✓ On relie deux à deux les caractères A et N1 qui ont la plus petite occurrence et on crée un nouveau nœud N2 auquel on affecte la somme des occurrences de A et de N1 ;
- ✓ On utilise une des deux cases A ou N1 pour stocker le contenu du nœud N2, et on supprime l'autre case.

| N2 |
|---|
| 4 |

Comme le nombre de case restant est de 1, on s'arrête.

L'arbre de Huffman engendré est le suivant.




Ainsi,

A a pour code 0

J a pour code 10

V a pour code 11

Le codage de JAVA donne donc 100110

**Remarques**

- Comme cela se doit, le caractère A le plus fréquent est le plus court (1 bit).

- Pour décompresser, on lit le codage binaire au fur et à mesure.

**1**00110 pas de code correspondant

**10**0110 correspond à J

10**0**110 correspond à A

100**1**10 pas de code correspondant

100**11**0 correspond à V

10011**0** correspond à A

- Pour les sons, l'encodage Huffman est principalement utilisé pour les sons au format MP3.
- Pour les images, l'encodage Huffman est principalement utilisé pour les images au format PNG et JPEG.
- Avec l'encodage Huffman, le taux de compression est compris entre 30% et 60%.

**Exercice**

Appliquez l'algorithme Huffman à la phrase bonjour a toutes et a tous