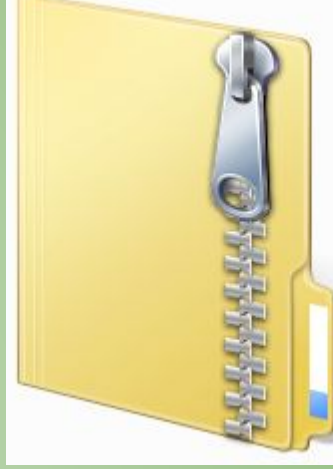
La Compression .zip
Principe de Codage de Huffman

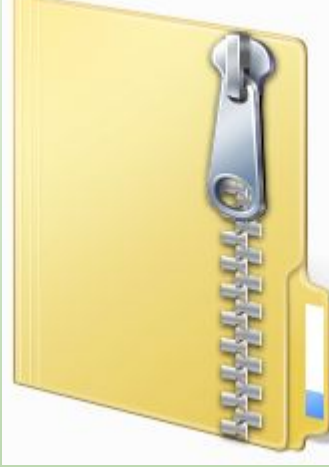
(ou comment compresser des caractères en binaire)

Plan d’exposé :



1. Introduction (point histoire)
2. Principe de Codage de Huffman (Compression)
3. Principe du Codage de Huffman (Décompresion)
4. Probabilité d’apparition / Taux de compression
5. Fichier .zip endommagé
6. Conclusion/Pour aller plus loin

1 . Introduction (point histoire)



Il existe dans un premier temps deux modes de compression :

Compression sans perte : Images, les fichiers textes, exécutables, archives

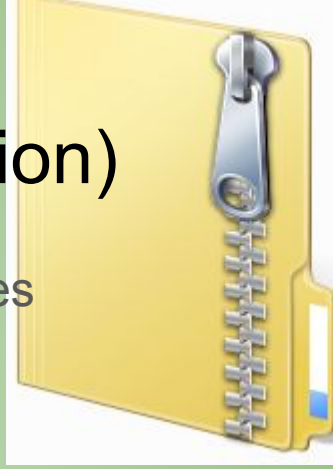
Compression avec perte : Photos, fichiers audios, vidéos

Ici nous allons nous intéresser à un mode de compression sans perte, le mode compression .zip .

David Albert Huffman publie en 1952 ses travaux pour sa thése de doctorat au MIT sur la compression binaire de caractères.

Le principe repose alors sur le codage d’un caractère dans le texte en un code plus simple en fonction de sa probabilité d’apparition.

2. Principe du codage de Huffman (compression)



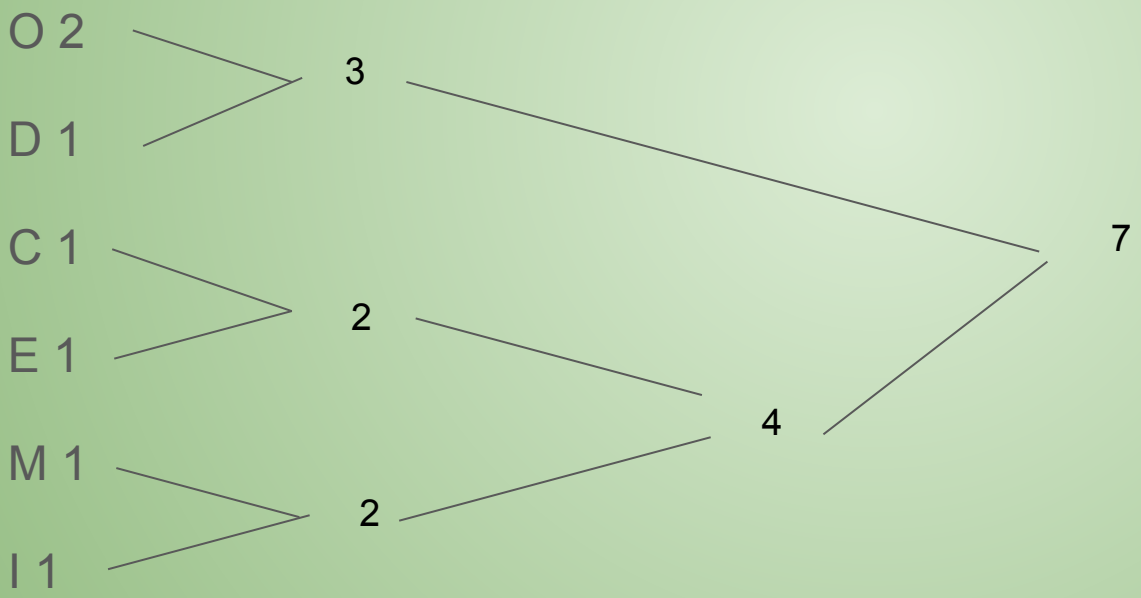
Dans un premier temps il faut commencer par effectuer le décompte des caractères et de leur répétition au sein du texte étudié.

ici nous avons donc :

DCODEMOI :

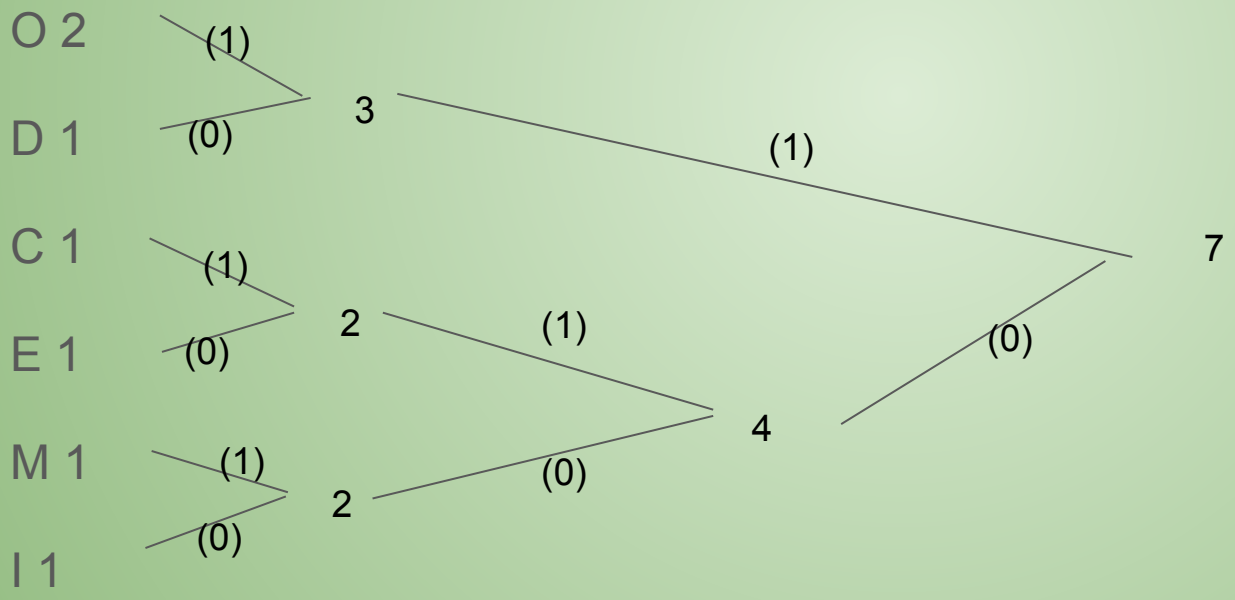
O 2	M 1	C 1
D 1	I 1	E 1

Nous allons nous appuyer sur un arbre qui se construit assez simplement afin de mieux comprendre la prochaine étape :



On commence par relier ensemble les branches de plus petit poids jusqu' à remonter au noeud principal,ici caractérisé par 7

La prochaine étape consiste à placer sur les branches descendantes la valeur 1 et la valeur 0 sur les branches montantes.



On obtient alors le codage suivant :

O : 11
D : 10
C : 011
E : 010
M : 001
I : 000

Nous avions donc : O : 11 D : 10 C : 011 E : 010 M : 001 I : 000

Le mot codée nous donnera donc :

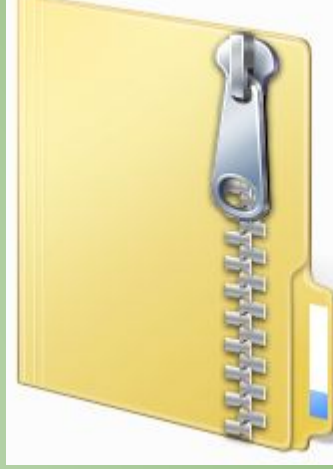
10 011 11 10 010 001 11 000 **- Soit 16 bits pour coder ce mot**

Observons maintenant le codage non compressé de notre texte en ASCII :

01100100 01100011 01101111 01100100 01100101 01101101 01101111 01101001
- **Soit 64 bits !**

On observe qu'il faut 4 fois moins de mémoire pour coder le même mot !

3. Principe de Huffman (Décompression)



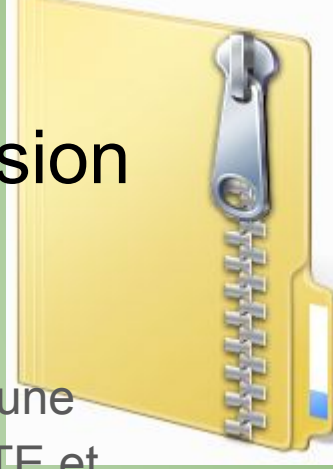
Travaillons sur le mot en binaire suivant :

111010010110

Afin de décompresser ce mot il va falloir utiliser le dictionnaire

P 111 E 110 T 10 A 0

4. Probabilité d’apparition / Taux de Compression

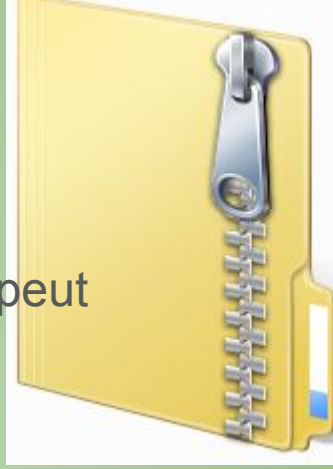


Afin de nous rendre compte que la probabilité d’apparition fréquente a une importance quant au taux de compression, comparons les mots PATATE et MANGER :

Manger = 1111101011000100 - 16 bits

Patate = 11101001011 - 10 bits

5. Fichier .zip endommagé



A présent on comprend que la perte du moindre bit dans le texte codé peut entraîner une impossibilité de lire le fichier archivé.

Reprenons l’exemple précédent mais cette fois avec 2 bit en moins :

1111001010

P 111 E 110 T 10 A 0

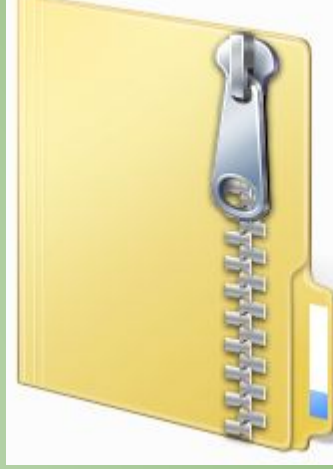
6. Conclusion

La Méthode de Huffman est une méthode qui permet une compression **sans pertes** avec un **gain de mémoire variable** selon les probabilités d'apparition des caractères.

Cependant il existe d'autres méthodes de codage équivalentes en terme d'efficacité, comme le **codage par intervalle** ou encore le **codage arithmétique** dont la mise en place est **plus complexe**.

Des systèmes comme le Bzip2 permettent une compression plus forte encore que le zip classique en ajoutant un nouveau processus de "triage" preliminaire.

Sources :



- Wikipédia, https://fr.wikipedia.org/wiki/Codage_de_Huffman
- Wikipédia, https://fr.wikipedia.org/wiki/Compression_de_données
- Dcode, https://www.dcode.fr/codage-huffman-compression
- Olivier Levêque, https://www.youtube.com/watch?v=UAY-wpHZCs4