# La compression de données

USTHB - M1 IV

Dr A. DAHMANE

# Compression de données

**Pourquoi ?**

1280 x 720 pixels, 30 trames/sec (fps)

➔ ~ 220 Go  pour un filme

➔  nécessite une compression de 1/50

**Comment ?**

- Enlever la redondance dans les données.
- Supprimer l'information non significative.

# Compression de données

Compression **sans perte**:

Identifier les éléments constituants de l'image et exploiter leur structure.

Compression **avec perte**:

Le but est de perdre le minimum d'information de manière à garder un aspect visuel le plus proche.

# Compression de données

La compression est **symétrique** lorsque la même méthode est utilisée pour compresser et décompresser l'information, tandis que la compression **asymétrique** demande plus de travail pour l'une des deux opérations.

# Compression de données

Critères des algorithmes de compression:

- **Efficacité**: le taux de compression (rapport entre la taille du fichier compressé et sa taille initiale) est significatif pour un type d'image.
- **Qualité** de compression: avec perte ou sans perte.
- **Vitesse** de compression/décompression: complexité de l'algorithme.
- **Accessibilité**: sous licence ou libre de droit.

# Compression de données

Taux de compression:

Sert à mesurer la performance d'un algorithme de compression

$$T = \frac{taille\ des\ données\ compressées}{taille\ initiale\ des\ données}$$

En gain:

$$Gain = 1 - T$$

# Compression RLE

Méthode **RLE (Run Length Encoding)**:

- Simple.
- Au lieu de coder plusieurs fois une valeur redondante, on code le nombre de fois où elle est présente.
- Taux de compression important que pour les données possédant de nombreux éléments consécutifs (exemple: images possédant de larges parties uniformes)

# Compression RLE

Méthode **RLE** (exemple):

| | | | |
|---|---|---|---|
| 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 |

5 255 1 0 10 255

# Compression RLE

<u>La méthode du fax :</u>

Si une valeur se répète 3 fois ou plus, le nombre de répétitions est codé d'une manière différente de celle où la valeur n'est pas redondante.

Code pour répétition:


non répétition:


<u>Exemple :</u>

La chaine XXYXXXXXYX  sera compressée en:

00000011 XXY 10000101 X 00000010 YX  en décimal: 3XXY133X2YX

# Compression de Huffman

Inventé en 1952 par *D.A.Huffman.*

**Principe***:*

Une analyse statistique des données,

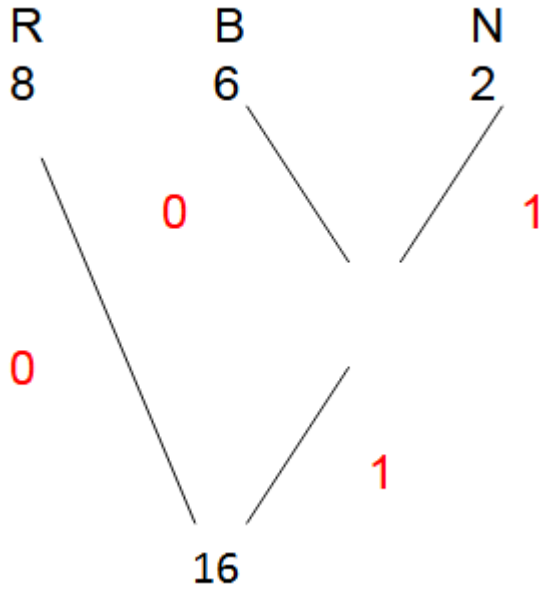
- associer aux données le plus souvent présentes, les codes les plus courts.
- Inversement, les données les plus rares se verront attribuer les codes les plus longs.

# Compression de Huffman

Exemple:

| | | | |
|---|---|---|---|
| b | b | b | b |
| b | b | r | r |
| n | r | r | r |
| n | r | r | r |

B : 6/16
R : 8/16
N : 2/16




Ceci nous donne :
R : 0
B : 10
N : 11

# Compression LZW

L’algorithme LZ77 inventé en 1977 par Abraham Lempel et Jakob Ziv et amélioré par Terry Welch en 1984.

LZW est à la base de la compression ZIP et des formats d’image gif et tiff.

**Principe**:

Il est basé sur la construction d’une table de chaines de caractères où chaque chaine est remplacée par un code.

# Compression LZW

## Algorithme:

Initialiser dictionnaire D
W ← lire un caractère
Tant qu'il reste des caractères à lire faire
    a ← lire caractère
    si ( w + a  est dans D ) alors
    w ← w + a
    sinon
    écrire le code de w
    ajouter w + a dans D
    w ← a
    fin si
fin tant que
écrire le code de w

# Compression LZW

## Algorithme de décompression:

Initialiser dictionnaire D

a ← lire caractère

écrire D(a)

Tant qu'il reste des caractères à lire faire

    b ← lire caractère

    si b est dans D alors

        w← D(b)

    sinon

        w ← D(a) + premier caractère de D(a)

    fin si

    écrire w

    ajouter D(a)+premier caractère de w à D

    a←b

fin tant que