

Culture numérique – module 2

Compression/Décompression de fichiers

Année 1



Archivage / Compression

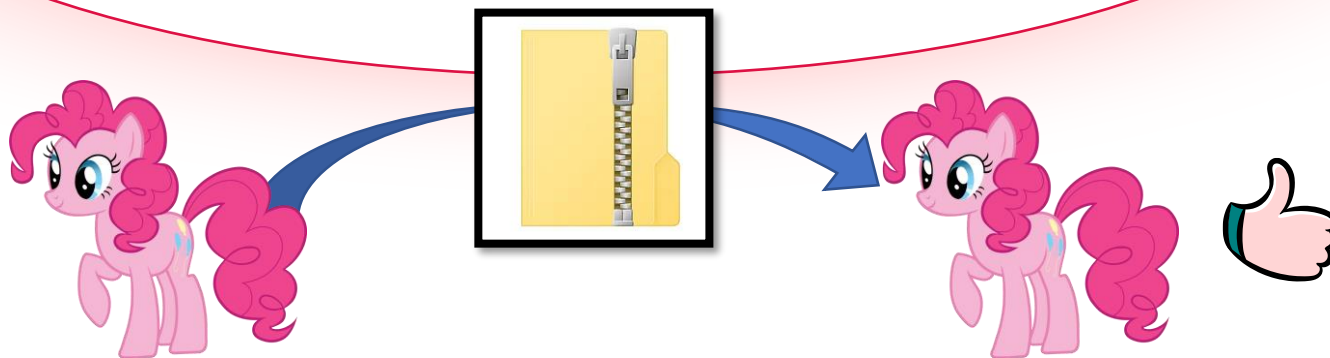
Questions ouvertes :

- À quoi sert l'archivage ?
- À quoi sert la compression ?
- Quels types de compression ?

Archivage / Compression

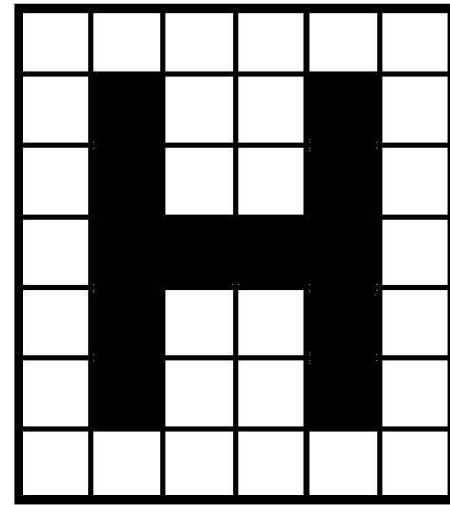
- **Archivage :**
 - Utilisation d'un seul fichier pour stocker plusieurs fichiers et dossiers
 - Permet de garder la structure de dossiers
- **Compression :**
 - Diminution de l'espace occupé sur le support numérique
 - 2 types de compressions :
 - **Compression sans perte :**
 - Les données après décompression sont **identiques aux données originales**
 - Elle sont réécrites d'une manière plus concise
 - **Compression avec perte :**
 - Les données après décompression sont **différentes des données originales**
(mais cette différence est peu perçue par l'œil ou l'oreille humaine)

Compression sans pertes



Compression sans perte

- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc ?



Compression sans perte

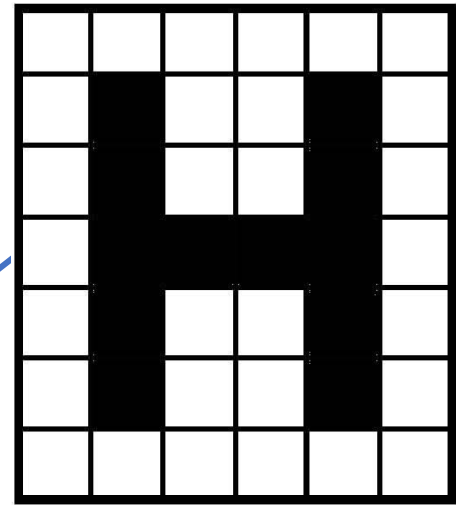
- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc ?

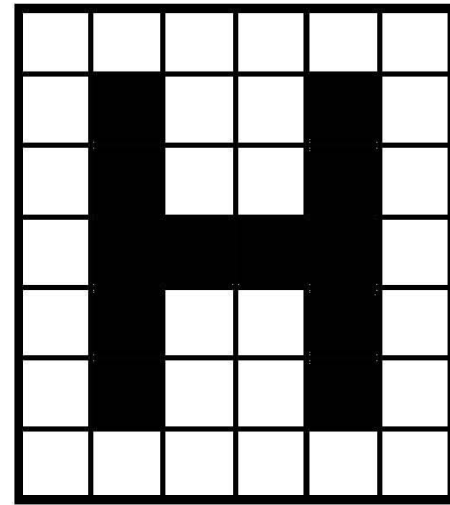
7,6,111111101101101101100001101101101101111111

Taille
image

ligne1

ligne7





- Codage par répétition* (RLE : *run-length encoding*)

* Aussi dit « par plage »

1	2	3	4	5	6
7	1	1	2	1	1
2	1	1	2	1	1
2	1	2	3	4	1
2		etc.			

- Codage par répétition (RLE : *run-length encoding*)

1	2	3	4	5	6
7	1	1	2	1	1
2	1	1	2	1	1
2	1	2	3	4	1
2		etc.			

- Codage par répétition (RLE : *run-length encoding*)

7,6,7B1N2B1N2B1N2B4N2B1N2B1N2B1N2B1N7B

(8%)

(représentation explicite des noirs et blancs)

1	2	3	4	5	6
7	1	1	2	1	1
2	1	1	2	1	1
2	1	2	3	4	1
2		etc.			

- Codage par répétition (RLE : *run-length encoding*)

7,6,7B1N2B1N2B1N2B4N2B1N2B1N2B1N2B1N7B

(8%)

(représentation explicite des noirs et blancs)

7,6,1:7121212124212121217

(45%)

(représentation implicite (premier chiffre = blanc, second = noir))

Compression sans perte

- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc ?

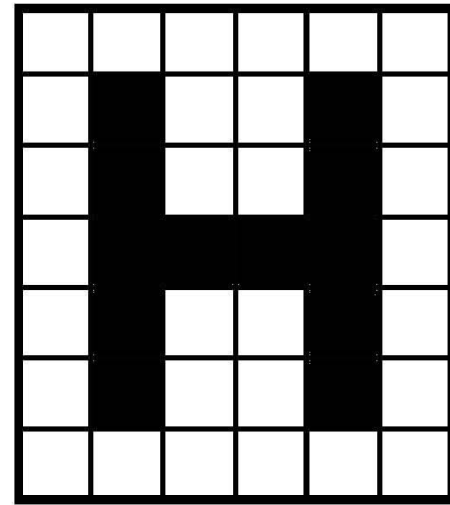
7,6,111111101101101101100001101101101101111111

Taille
image

ligne1

ligne7

- Codage par répétition (RLE : *run-length encoding*)



Compression sans perte

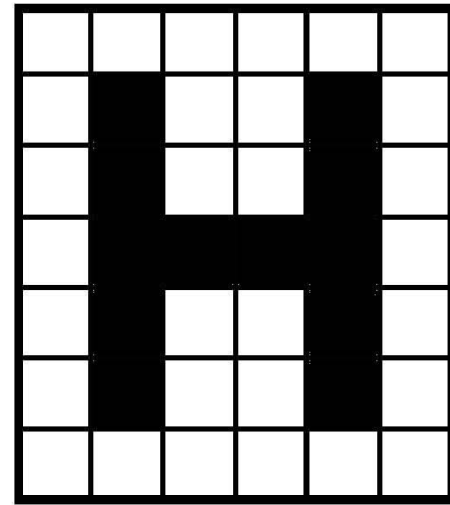
- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc ?

7,6,111111101101101101100001101101101101111111

Taille image
ligne1

ligne7

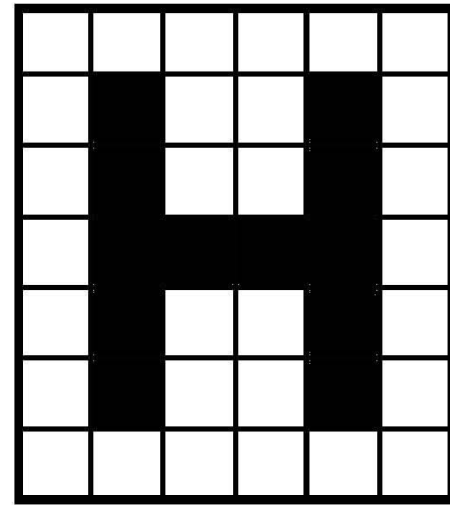
- Codage par répétition (RLE : *run-length encoding*)
- Codage par entropie (Morse, Huffman)



- **Codage par entropie (Morse, Huffman) : principes !**
 1. On coupe la donnée en succession de symboles :
 - Par exemple : 00 / 01 / 10 / 11
 - 2 premières lignes du H : 11 / 11 / 11 / 10 / 11 / 01
 2. On fait des stats pour trouver les symboles les plus fréquents :
 - Ici : sur les deux premières lignes : 11 est le plus fréquent.
 3. On attribue un code, des symboles, à ces caractères en s'arrangeant pour que le plus courant ait le code le plus petit !

1	1	1	1	1	1
1	0	1	1	0	1

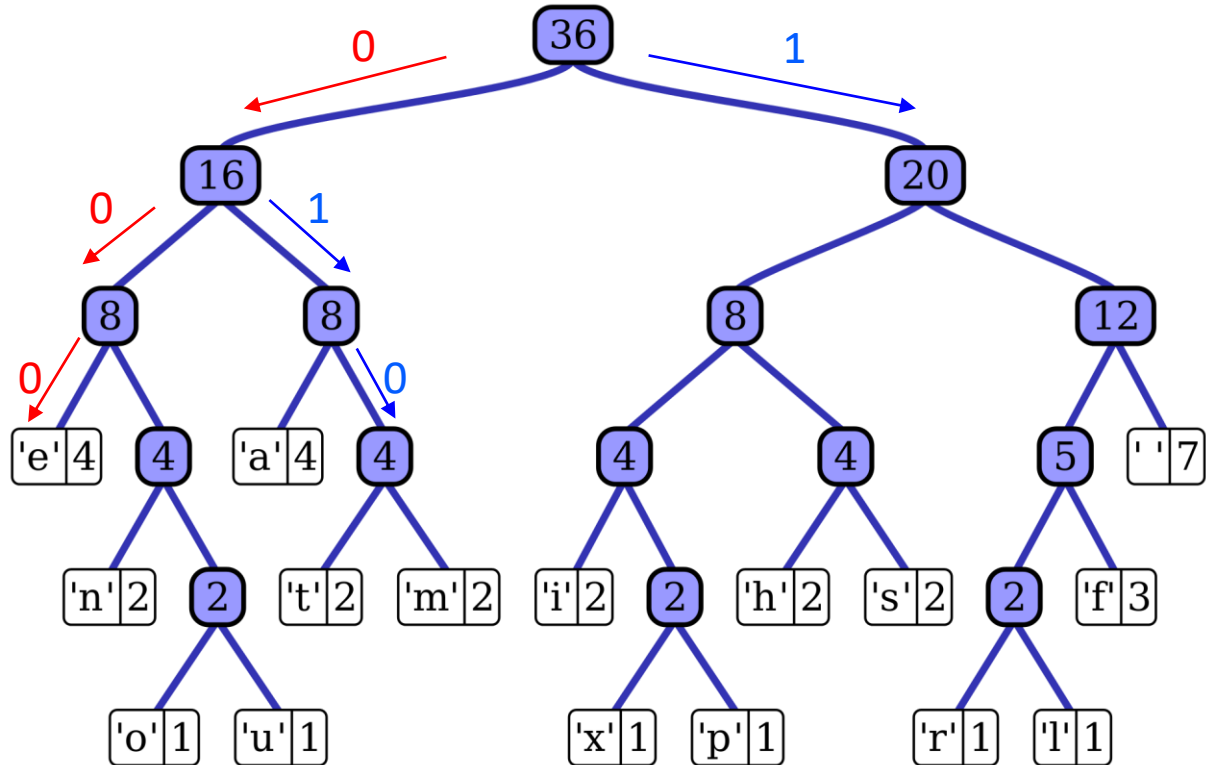
- **Codage par entropie (Morse, Huffman) : conditions**
 - Il faut que ces codes soient « préfixes », c'est-à-dire qu'aucun de ces codes ne soient le début d'un autre !
 - 0 comme premier code et 00 comme second est interdit !
 - Car 000 pourrait être interprété de deux manières
- En pratique c'est assez facile : On construit « *l'arbre de Huffman* ».



Exemple d'arbre sur une phrase :

- On commence par le bas à partir des symboles les moins courants.
- On remonte peu à peu pour construire l'arbre en groupant les moins courant entre eux.
- On attribue ensuite les codes à partir du haut (un 0 à gauche, un 1 à droite par exemple)

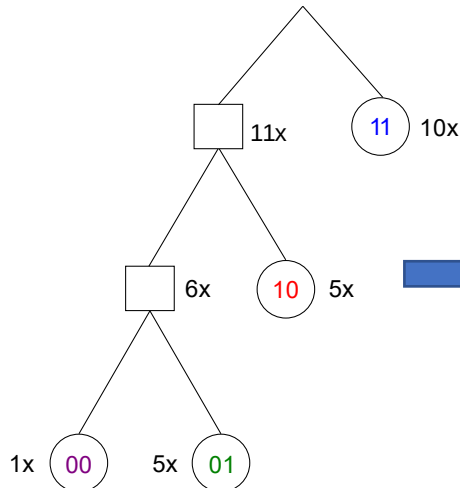
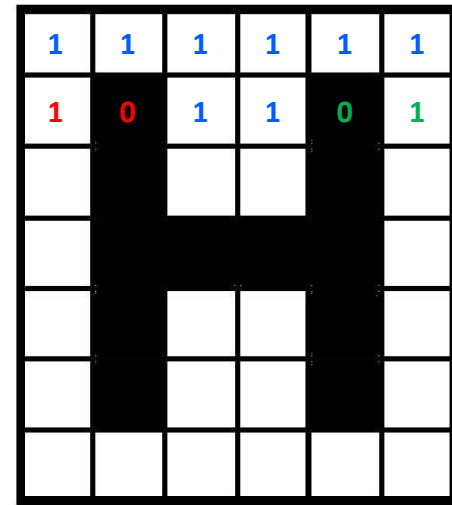
Ici e → 000 et a → 010



- Exemple avec notre H, séparons le en groupe de 2 !

11 / 11 / 11 / 10 / 11 / 01 / 10 / 11 / 01 / 10 / 00 / 01 / 10 / 11 / 01 / 10 / 11 / 01 / 11 / 11 / 11

Il y a : 10 fois les symboles 11, 5 fois le symbole 10, 5 fois le symbole 01 et une fois le symbole 00



Code :

À gauche = 0, à droite = 1

1 => 11

01 => 10

000 => 00

001 => 01

11 / 11 / 11 / 10 / 11 / 01 / 10 / 11 / 01 / 10 / 00 / 01 / 10 / 11 / 01 / 11 / 11 / 11



1 / 1 / 1 / 01 / 1 / 001 / 01 / 1 / 001 / 01 / 00 / 001 / 01 / 1 / 001 / 01 / 1 / 1 / 1

(11,9%) Plus les codes d'origines sont complexes et long, plus cette compression sera efficace !

- **Codage par entropie (Morse, Huffman) : défauts et avantages**
 - Dépend beaucoup du **choix du nombre de symboles** en entrée (ici on a coupé par bloc de 2, mais ça aurait pu être 4)
 - Dépend beaucoup de la **fréquence d'apparition d'un symbole** dans les données, mais cela peut évoluer au cours du temps ! (il faut alors améliorer l'algorithme)
 - Ne prends pas en compte les **spécificités du type de donnée** (texte, images etc.)

Compression sans perte

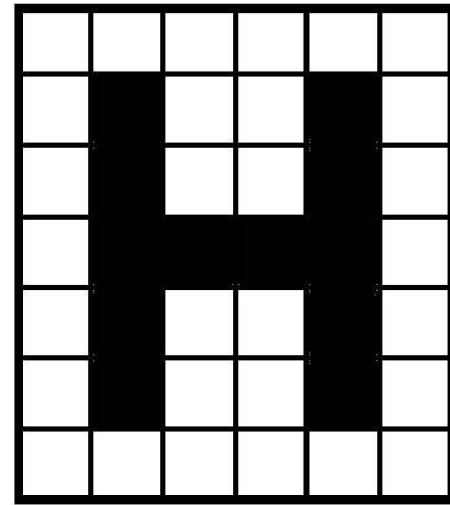
- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc ?

7,6,111111101101101101100001101101101101111111

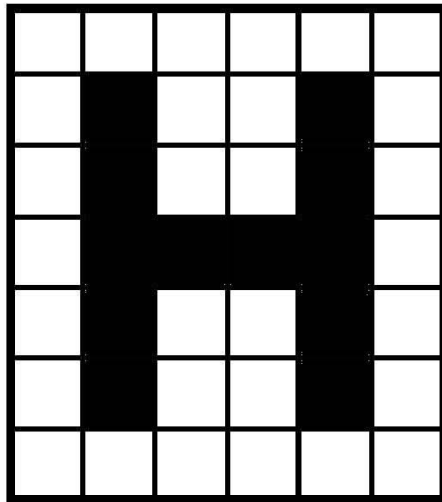
Taille image
ligne1

ligne7

- Codage par répétition (RLE : *run-length encoding*)
- Codage par entropie (Morse, Huffman)



Compression sans perte



- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc ?

7,6,111111101101101101100001101101101101111111

Taille ligne1
image

ligne7

- Codage par répétition (RLE : *run-length encoding*)
- Codage par entropie (Morse, Huffman)
- Codage par dictionnaire
 - Recherche de motifs récurrents (et longs)

A:111111:B:101101:C:100001

7,6,ABBCBBA

(15%)

Compression sans perte

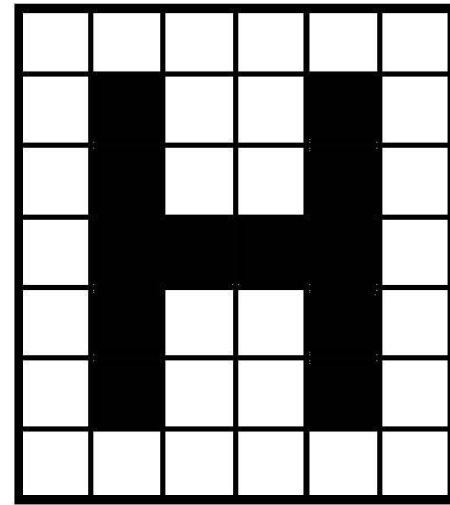
- Comment pourrait-on compresser (sans perte) le contenu de cette image en noir et blanc :

7,6,111111101101101101100001101101101101111111

Taille image
ligne1

ligne7

- Codage par répétition (RLE : *run-length encoding*)
- Codage par entropie (Morse, Huffman)
- **Codage par dictionnaire**
 - > Le dictionnaire peut être défini une fois pour toute ou mis à jour au fur et à mesure en fonction de règles plus ou moins complexes.
 - > Il peut également être séparé (en début du code) ou on peut faire référence à des bouts du code précédent



Compression sans perte

Exemple de **codage par dictionnaire** faisant référence à des mots du code (pas de dictionnaire inclut explicitement) :

Original : Hello everyone! Hello world!
Encodé : Hello everyone! <16,6>world!

Le 16,6 veut dire : 16 caractère avant, et le mot à récupérer fait 6 lettres.

Quelques format de compression sans pertes

Questions ouvertes :

- Connaissez-vous des formats de compression de fichiers ?
 - **Zip**
 - 7z
 - Rar
 - Gzip (tar.gz) / Bz2 (tar.bz)
- Comparaison des compressions : <http://rlwpx.free.fr/WPFF/complc.htm>

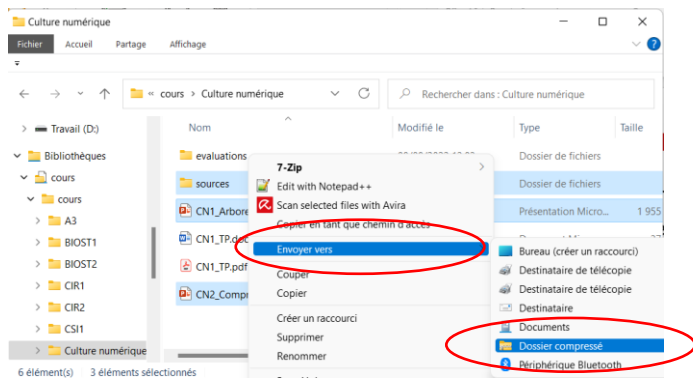
Format Zip



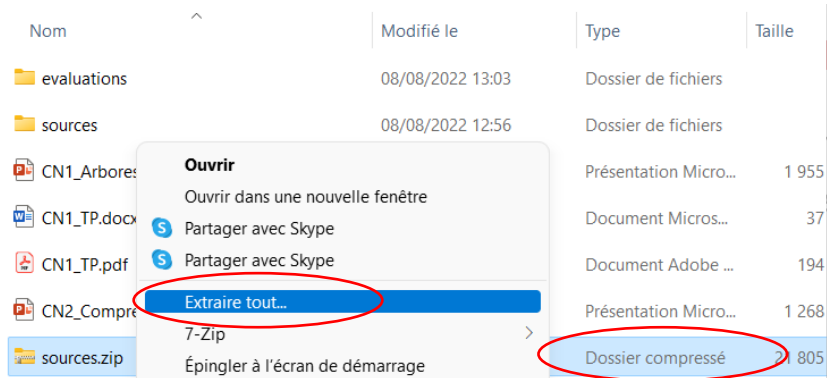
- Format normalisé et intégré à tous les systèmes d'exploitation
- Compression très rapide (en temps de compression)
- Performance de compression moyenne
- Algorithme couplant **dictionnaire** et compression de **Hoffman**

Format Zip à privilégier en général car prise en charge nativement par tout les systèmes.

Compression :



Décompression :



Format 7z



- Format **open-source**, mais nécessite un logiciel (7zip ou autre)
 - **N'est pas disponible par défaut sous windows !**
 - Logiciel de compression conseillé → 7zip : 7-zip.org/ **7ZIP**
- Compression **rapide** (en temps de compression)
- **Bonne performance de compression**
- Algorithme de type **dictionnaire**.

Format Rar



- Format **propriétaire** : nécessite le logiciel **Winrar**
 - **N'est pas disponible par défaut sous windows !**
- Permet le découpage multi-volume
 - Stockage d'une archive en plusieurs fichiers
- Compression assez **lente** (en temps de compression)
- Performance de compression :
 - **Meilleure** que **zip**
 - Mais **moins bonne** que **7z**

Format à éviter !

Formats tar, gz et bz2

- Format open-source, utilisé historiquement sous **Linux**
 - **N'est pas disponible par défaut sous windows !**
 - Logiciel de compression **7zip** conseillé sous Windows et Mac : 7-zip.org/
- Décorrélation de l'archive et de la compression :
 - **.tar** : archive
 - **.tar.gz** ou **tar.bz2** : compression de l'archive tar au format **gzip** ou **bz2**

Sous Mac ?



- Gestion **native** du format zip sous Mac
 - (compression et décompression)
 - => Pas de logiciel nécessaire
- Autres formats (7z, rar, ...) gérés par :
 - Keka : keka.io/fr/ 

Protection d'une archive par mot de passe

- Possibilité de limiter l'accès à archive avec mot de passe
 - Usage ? → Echange documents sensibles à destination de tiers
- 2 méthodes chiffrements :
 - ZipCrypto
 - Sécurité reconnue comme très faible
 - Décompression possible nativement sous Windows
 - Compression impossible sous windows
 - AES-256
 - Meilleure sécurité => **À privilégier si usage de chiffrement !**
 - **Nécessite un logiciel adapté sous Windows (7zip par exemple)**

Toujours se poser la question : **Compatibilité, facilité d'usage / Sécurité**

Formats de compression sans pertes

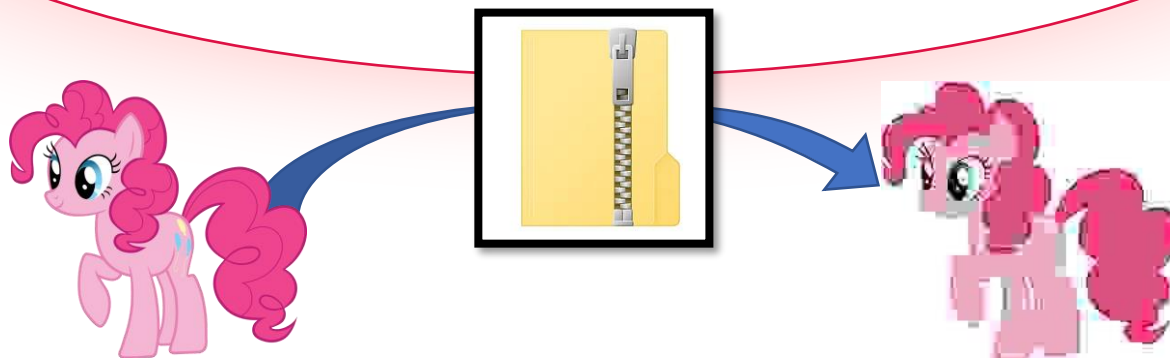
Sauf utilisation de la compression dans un contexte particulier

Privilégier le format Zip !!!

En logiciel :

- **Privilégiez 7zip**
- **Fuyez winrar !**

Compression avec pertes



Dans quels cas ?

- Quand les données exactes ne sont pas critiques :
 - Ex : Pour des données de type texte on évite
 - « Paul » très différent de « Pul ».
- En général quand il s'agit de vidéo, d'image ou de son
 - Car une musique, une vidéo ou une image légèrement moins qualitative n'est pas dramatique.

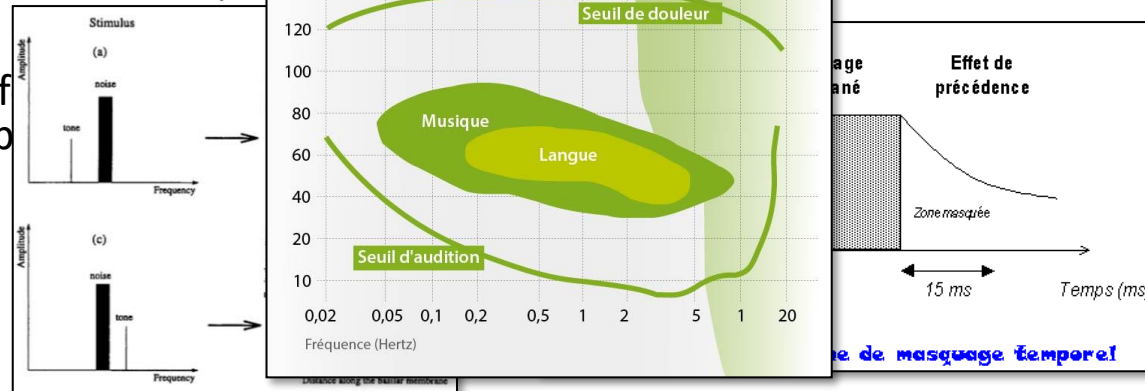
Comment compresser ? Idée générale

- Idée générale : Il faut réduire ou supprimer les données là ou elles ne sont pas (ou peu) perceptible par les sens de l'être humain.
- Exemple :
 - Pour le son : certaines **fréquences** sont peu perceptibles.
 - Pour l'image : certaines **variations de couleurs** sont peu perceptible (notamment quand elles sont rapide)
 - Pour la vidéo : idem que pour l'image mais avec en plus un aspect temporel

Le son

- Principe :
 - N'échantillonner que pour retenir les **fréquences audibles** pour l'être humain (20Hz -> 20Khz environ).
 - Supprimer les fréquences inaudibles.
 - Profiter de différences de sensibilité de l'oreille à un ensemble de filtres
 - exemple : l'effet de masquage temporel

- Une fréquence comp...



age
ané

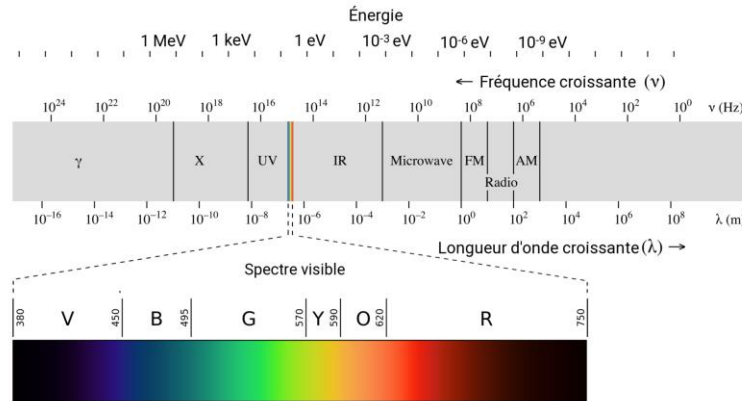
Effet de
précédence

de

de masquage temporel

L'image

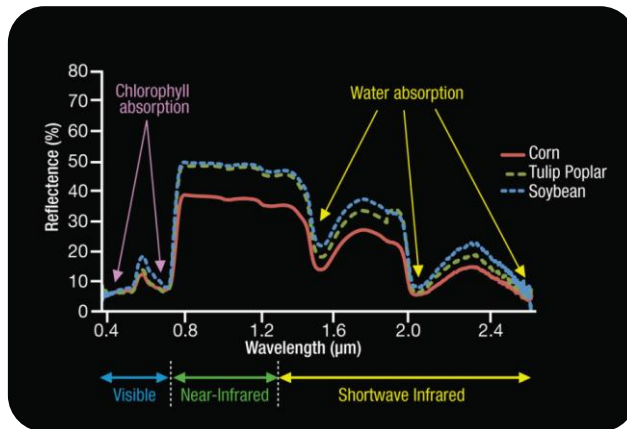
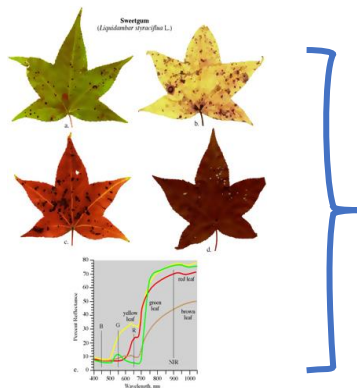
- Rappel sur la lumière :



- La **lumière visible** est constituée d'une infinité de longueur d'onde, perçues comme des **couleurs** par l'œil humain.

L'image

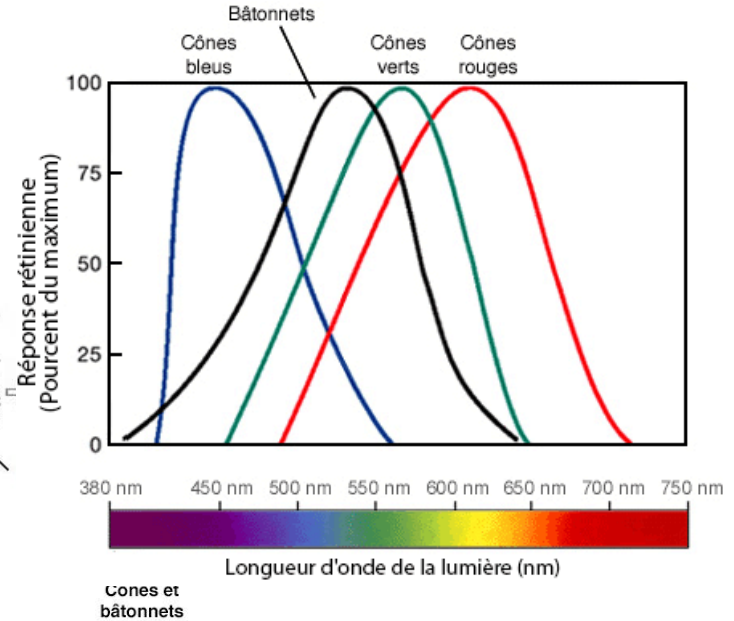
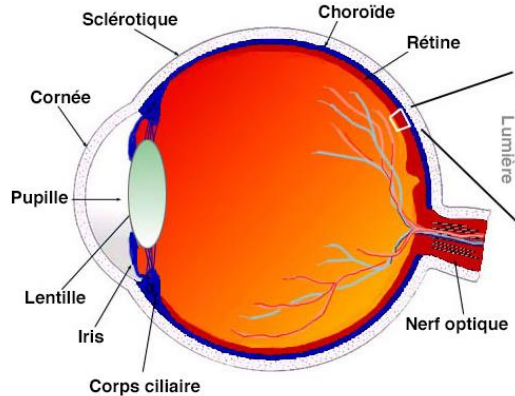
- Si on voulait reproduire l'ensemble de signaux provenant d'une feuille d'arbre, par exemple : il faudrait une infinité d'émetteurs différents ! Et pour seulement un pixel de la feuille !



- Contrairement aux ondes mécaniques du son : on ne sait pas fabriquer de système petit et peu cher émettant toutes les fréquences lumineuses.

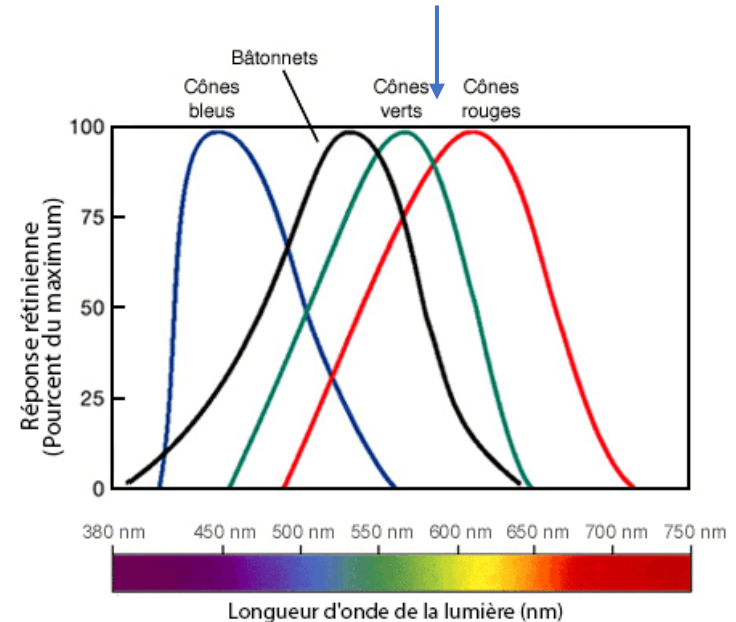
L'image

- Par chance :
 - L'œil humain ne perçoit que des plages de fréquences via ses bâtonnets et ses cônes.
 - Centrée, en ξ et le **bleu**.
 - Conséquence capable de faire un mélange de t véritable cou



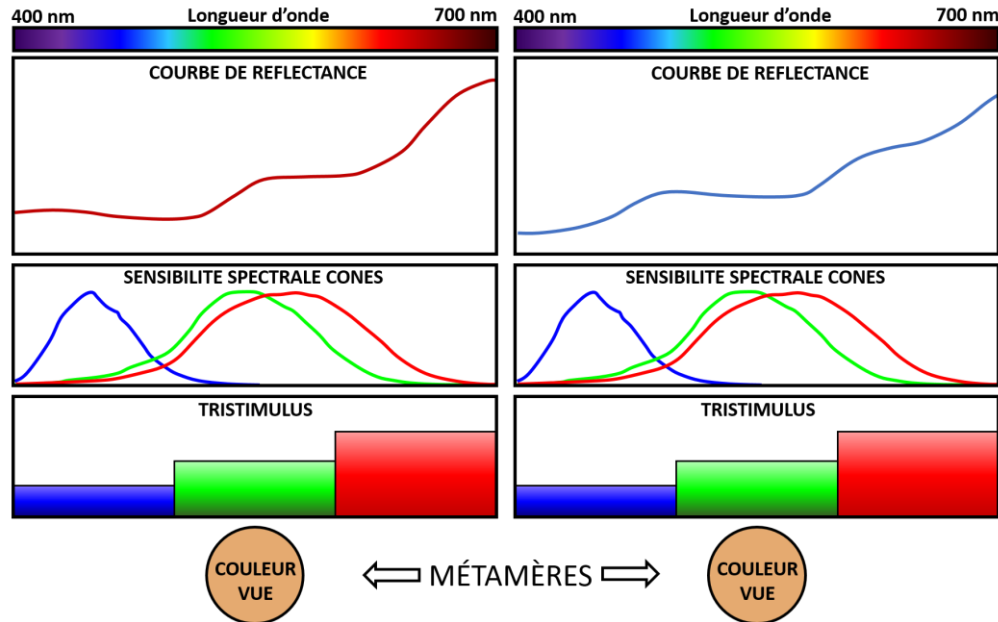
L'image

- Démonstration :
 - On peut observer sur les courbes que si j'excite :
 - Le récepteur du **rouge**
 - Le récepteur du **vert**
 - Le seul endroit correspondant est le **jaune**
 - De fait : impossible de distinguer un objet émettant un peu de rouge et un peu de vert d'un objet émettant vraiment du « jaune ».



L'image

- On parle de couleur **métamère**



L'image

- Note : parfois, certaines couleurs perçue n'existe pas vraiment « physiquement » du point de vue des longueur d'onde.
- Exemple : **Le blanc**.



L'image

- **Conséquences :**
 - Inutile de tenter de reproduire le spectre complet : il suffit de tromper suffisamment les trois capteurs de l'œil humain.
 - Pour ce faire : On va composer chaque **pixel** des écrans de trois émetteurs de longueur d'onde dans le **rouge**, le **vert** et le **bleu**.
 - Les émetteurs peuvent être très divers (LCD, OLED etc.)
 - Voir le chapitre sur le matériel.
 - Vu que nous n'émettrons que trois longueur d'onde, inutile de stocker plus d'information. On stockera trois couleurs par pixel d'une image.

L'image

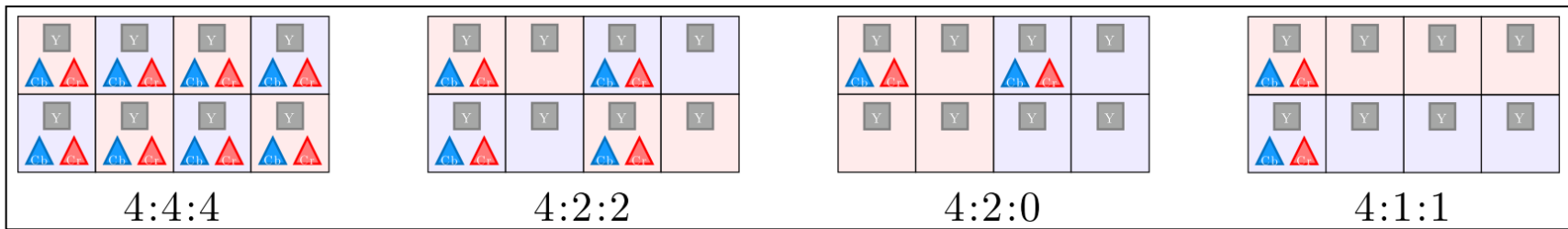
- Une image numérique est donc composée de **pixels (petit carré de couleur)** dont chaque pixels contient une information sur :
 - Son niveau de Rouge
 - Son niveau de Vert
 - Son niveau Bleu
 - Et éventuellement un canal qui indique à quel point la couleur doit être considérée comme **transparente** quand elle se superpose à d'autres.
 - On parle de **canal Alpha** pour cette transparence.
 - On parle de **RVB / RVBA** ou **RGB / RGBA** pour l'ensemble de ce système.

L'image

- Principe :
 - L'œil humain est **peu sensible aux variations de couleurs**, mais beaucoup à la luminosité, donc :
 - On change la représentation de l'image :
 - De **Rouge – Vert – Bleu** vers un système **Luma – Chrominance bleu – Chrominance rouge**
 - Où Luma est la luminosité d'ensemble et les chrominances les informations de couleurs.
 - On retrouvera le vert par un calcul car **Luma = R + B + V**
 - On parle de **RVB/RGB** pour le premier système et de **YUV** ou encore de **YCbCr** pour le second.
- Une fois l'image transformée, on garde le maximum d'information sur Luma, mais pas sur les chrominances.

L'image

- Quelques modes existants, exemple avec un rectangle de 8*2 pixels :



100 % de l'information
Chaque pixels contient
Luma, Cb et Cr

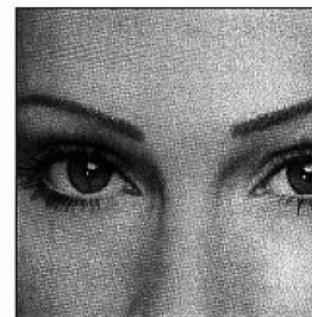
100% de l'information
de luminosité, un pixel
sur deux d'information
chromatique

100% de l'information
de luminosité, un pixel
d'info. Chromatique
compte pour les 4
environnants

100% de l'information
de luminosité, un pixel
d'info. Chromatique
compte pour les 4 de la
même ligne

L'image

- Par la suite, un peu comme pour le son :
 - On décompose en fréquences
 - Dans le domaine spatial, une fréquence est une variation (de luminosité, de couleurs etc.)
 - On identifie les fréquences spatiales peu perceptible par l'être humain :
 - L'être humain perçoit peu les petites variations mais beaucoup les grosses variations.
 - On filtre les zones peu perçues pour compresser les données



a. Original image



b. With 10:1 compression



c. With 45:1 compression

FIGURE 27-15
Example of JPEG distortion. Figure (a) shows the original image, while (b) and (c) shows restored images using compression ratios of 10:1 and 45:1, respectively. The high compression ratio used in (c) results in each 8×8 pixel group being represented by less than 12 bits.

component

component

L'image

- Une fois ces étapes réalisées :
 - Comme pour le son, on applique ensuite des algorithmes de compression sans pertes
 - Exemple pour le **format JPEG** : **Codage par plage (RLE)** puis **codage de Huffman** !
- Principaux format de compression d'image avec pertes :
 - **JPEG** (Le principal, très largement)
 - **WebP** (plus rare, plus performant, en cours de déploiement)
 - **AVIF/HEIC/HEIF** (plus rare, mais très performant)

L'image

- Logiciels recommandés :
 - Visionneurs :
 - Par défaut chaque système permet de visualiser des formats sans pertes (bmp, png) et avec pertes (jpeg).
 - Pour des formats plus exotiques, il faudra installer d'autres logiciels :
 - **Xnview** (windows, mac, Linux)
 - Manipulateur d'image :
 - Par défaut : chaque système dispose d'un manipulateur sommaire (**Paint** sous Windows)
 - Plus avancés : **GIMP** (gratuit) ou Photoshop (payant)

L'image

- Questions ouvertes avancées :

Qu'est-ce que le format vectoriel ? →



Un poney vectoriel

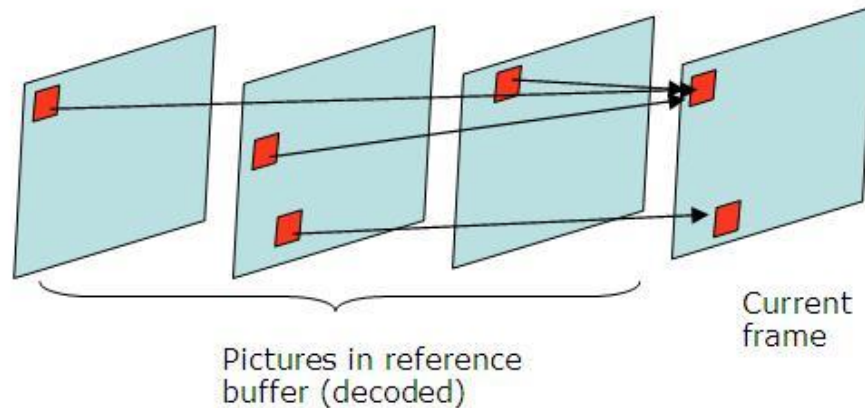
- Quelles sont les conséquences sur la taille du fichier résultant et sur sa compressibilité ?
- Quelles sont les conséquences sur l'agrandissement du fichier ?
- Principaux formats et logiciels?
 - SVG (gratuit, très courant) : **INKSCAPE**.
 - AI (propriétaire adobe, existe dans le monde pro en parallèle de SVG) : **ADOBE ILLUSTRATOR**.
 - EVF : Format Windows. Bien pris en charge sur Word etc.

La vidéo

- La vidéo est une suite d'image
- Très proche du principe des images pour la compression
- Mais on y ajoute un **aspect temporel** :
 - par exemple : sur un plan fixe ou un avion passe au loin, **inutile de recoder toutes l'image**. Il suffit de dire que c'est la même qu'auparavant + l'avion.
 - On va parler :
 - **D'image de référence** pour l'image du plan fixe.
 - **D'image inter-frame** pour l'avion qui se déplace.

La vidéo

- Dans la réalité c'est plus complexe : on peut le faire sur des bouts d'image et sur plusieurs images de références :



- Une fois qu'on retrouve une zone dans le passé on se contente de coder **son mouvement**, et **non son contenu**.

La vidéo

- Principaux formats de codage :
 - MPEG2
 - MPEG4/H264
 - HEVC/H265
- Principaux formats conteneurs :
 - AVI
 - MP4
 - MKV



La vidéo

- Avoir des informations sur une vidéo ? (type de son, codage de la vidéo, sous titres inclus etc.)
 - Sous Windows : Clic-droit -> propriétés.
 - Logiciel **MediaInfo** (Windows, Linux, Mac).
- Lire une vidéo :
 - Lecteur système (souvent mauvais)
 - VLC (compatible avec tout, portable, mais lourd)
 - MPC-HC / MPC-BE (léger, mais doit être un peu paramétré)

Les précautions à prendre !

- Si je dois travailler sur un fichier (une image par exemple) que je risque d'enregistrer de nombreuses fois.

Vaut-il mieux :

- Choisir un codage sans pertes ?
- Choisir un codage avec pertes ?





FLIF
-Q100
PSNR: inf
502601 bytes



WebP
q:100
PSNR: inf
526366 bytes



BPG
q:0
PSNR: inf
642041 bytes



JPEG
q:100
PSNR: 50.8851
581904 bytes

0 generations (original)

Les précautions à prendre !

- Si je veux inclure des images ou graphiques dans un gros document pour un rapport :
 - Quels formats choisir ?
 - Réponse :
 - JPEG
 - SVG

