

# Compression d'images fixes (sans perte)

INF4710 Introduction aux technologies  
multimédia

# Plan

- ▶ Qu'est-ce qu'une image numérique
- ▶ Compression sans perte
  - « Autres »
  - Entropique
  - Avec dictionnaire

# Qu'est-ce qu'une image numérique

- ▶ Une image numérique est une matrice de pixels.
- ▶ Un pixel est un élément d'image (unité de surface). À chaque pixel est associé une couleur, habituellement décomposée en trois couleurs primaires (rouge (R), vert (G), bleu (B)).

# Qu'est-ce qu'une image numérique

Zoom d'une image. La matrice de pixels devient apparente.



Pour le blanc  
 $R=G=B=255$

# Qu'est-ce qu'une image numérique

## ► Une image dans MATLAB

```
>> image=imread('smallimage.jpg') % image est <15x16x3 uint8>

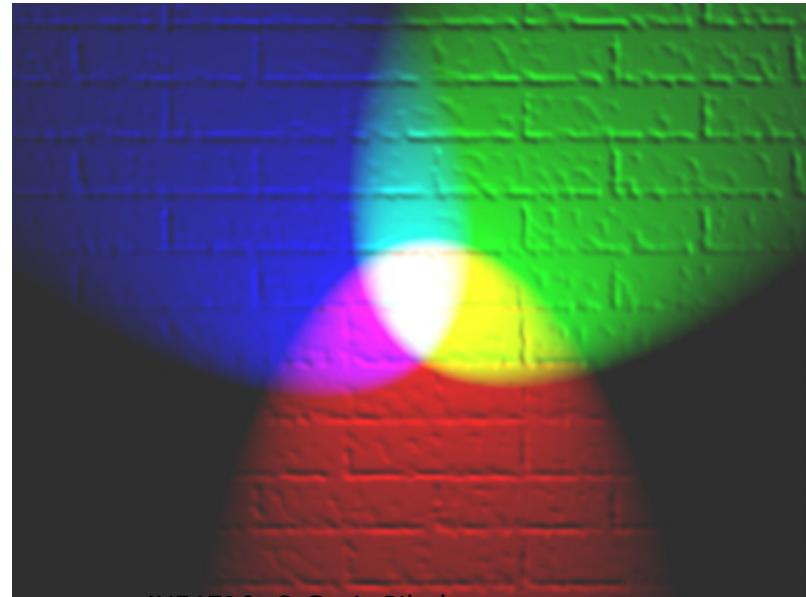
> image(:,:,1) =
Columns 1 through 13
 160 168 178 184 184 185 186 188 191 197 196 195 197 197 199 208
 128 126 126 123 123 128 139 146 154 151 149 152 157 152 152 161
 125 123 121 116 113 112 117 119 117 110 104 106 112 104 103 112
 152 153 154 152 149 144 141 139 148 147 141 134 134 131 131 138
 157 155 154 153 155 157 159 158 162 179 182 171 166 168 168 166
 139 136 136 136 139 142 144 142 164 195 210 198 193 196 197 191
 136 135 136 138 141 139 136 130 155 188 205 196 192 195 198 196
 148 145 143 144 146 148 147 144 156 182 194 187 183 186 193 199
 164 156 158 162 156 160 161 150 165 179 188 187 185 187 189 187
 194 186 189 188 183 183 187 175 173 182 188 185 184 186 190 189
 183 180 185 181 176 179 184 179 175 178 180 176 176 179 184 182
 167 172 175 170 165 167 172 170 168 170 169 167 168 172 174 172
 166 177 179 171 165 162 162 163 163 164 165 163 164 166 166 164
 157 173 179 170 165 160 158 157 157 158 160 161 162 160 159 156
 149 170 179 172 171 165 159 158 155 155 158 159 158 154 153 152

> image(:,:,2) =
Columns 1 through 13
 162 170 180 186 188 189 190 192 197 203 205 204 206 206 209 218
 130 131 128 127 127 134 143 152 160 160 158 163 166 163 162 173
...
```

# Qu'est-ce qu'une image numérique

## ▶ Modèle additif RGB

- Les couleurs rouge, vert, et bleu sont mélangées pour reproduire un grand nombre de couleurs.
- Chaque pixel est donc représenté par un triplet (R,G,B), où chaque élément du triplet est entre 0 et 255 inclusivement (unsigned char (8 bits)).



# Qu'est-ce qu'une image numérique

- ▶ Taille d'une image numérique
  - La matrice de pixels peut avoir une taille de 160x120, 320x240, ... 1024x728, 1900x1200, ...
  - Puisque chaque pixel d'une image est représenté par un triplet (R,G,B), 24 bits par triplet, une image 160 x 120, nécessite  $160 \times 120 \times 24 = 460800$  bits ou 56.25 kilooctets.
  - Une image de 320x240, nécessite 225 kilooctets.
  - Une image de 1024x768, nécessite 2304 kilooctets, 2.25 Mégaoctets.
  - Typiquement, on compresse les images...

# Compression d'images

- ▶ Il existe deux grandes familles de méthodes de compression:
  - Compression sans perte:
    - Autre
    - Codage entropique
    - Dictionnaire
  - Compression avec perte
    - Quantification
    - DCT (Transformée en cosinus discrète)
    - Ondelettes

# Compression sans perte: Autre

- ▶ Codage par plage (run-length encoding: RLE)
  - Appliqué sur des images en noir et blanc principalement (exemple: format BMP, JPEG pour codage des coefficients);
  - On compte le nombre de symboles (exemple: pixels blancs et noirs) consécutifs;
  - Le compteur est typiquement sur 8 bits.
  - Pas nécessairement appliqué sur 2 symboles, mais plus efficace quand il y a moins de symboles.

# Compression sans perte: Autre

- ▶ Codage par plage (run-length encoding: RLE)
- ▶ Exemple:
  - Soit le code binaire suivant à compresser:
  - 1111**010011110010111110011111**
  - Trouver le code RLE (sur 4 bits)
  - Soit le code binaire suivant à compresser:
  - 0000**11110000000000001111**
  - Trouver le code RLE (sur 4 bits)

# Compression sans perte: Autre

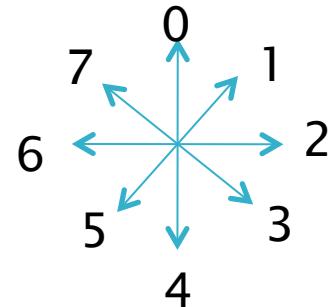
- ▶ Codage par plage (run-length encoding: RLE)
- ▶ Exemple:
  - Soit le code binaire suivant à compresser:
  - 1111**0000**1111**0000**1111**0000**1111
  - Trouver le code RLE (pour des codes 1 bit, et un compteur de 4 bits)
  - Soit le code binaire suivant à compresser:
  - 0000**1111**00000000**00000000**1111
  - Trouver le code RLE (pour des codes 1 bit, et un compteur de 4 bits)

# Taux de compression

- ▶ Il existe deux définitions du taux de compression:
  - Taille du message compressé/Taille du message initiale (définition peu courante)
  - $1 - (\text{Taille du message compressé}/\text{Taille du message initiale})$  (*space saving*)
- ▶ Il y a aussi le quotient de compression (*compression ratio*)
  - Taille du message initial/Taille du message compressé  
(inverse de la première définition du taux de compression)

# Compression sans perte: Autre

- ▶ Codage en chaîne (Chain code)
  - Appliqué sur les images en noir et blanc
  - Codage séparé de chaque composante connectée
  - Construction du code
    1. Un pixel sur la frontière d'une composante connectée est encodé par sa coordonnée ( $x,y$ );
    2. Ensuite, chaque pixel de la composante connectée est encodé par la direction empruntée pour aller du pixel précédent vers le pixel suivant à l'aide, par exemple, du code de Freeman.



# Compression sans perte: Autre

- ▶ Codage en chaîne (Chain code)
- ▶ Exemple:
  - Soit l'image suivante
  - 000000000
  - 001100010
  - 001110010
  - 000000000
  - Trouver le code en chaîne avec le code Freeman.

# Compression sans perte: Autre

## ▶ Codage prédictif

- Plutôt qu'encoder le signal lui-même, on code l'erreur de reconstruction du signal suite à une prédiction.
- L'erreur  $e_n$  pour le  $n^{\text{ième}}$  symbole est:

$$e_n = x_n - \hat{x}_n$$

- où  $x_n$  est un symbole à coder et  $\hat{x}_n$  est le symbole prédit.
- Le nombre de bits requis pour le codage dépend de la qualité de la prédiction.

# Compression sans perte: Autre

## ▶ Codage prédictif

▶ Soit les signaux suivants à compresser:

- Exemple 1: 56 55 52 54 65 54
- Exemple 2: 100 50 97 52 98 51

◦ et les prédicteurs suivants:

$$\hat{x}_n = x_{n-1}$$

$$\hat{x}_n = 0.1x_{n-1} + 0.9x_{n-2}$$

◦ Trouver les codes prédictifs

# Compression sans perte: Autre

## ▶ Codage prédictif

- Pour décoder le signal, on fait

$$x_n = \hat{x}_n + e_n$$

- De façon générale,  $\hat{x}_n$  est donné par

$$\hat{x}_n = \text{round}\left(\sum_{i=1}^m \alpha_i x_{n-i}\right)$$

- Il s'agit de trouver les meilleurs  $\alpha_i$  possible pour minimiser l'erreur de prédiction. Dans ce but, on peut analyser le signal.

# Compression sans perte: Autre

- ▶ Codage prédictif
  - Peut être fait en 1D ou 2D pour exploiter la corrélation entre les pixels.
  - N'est pas optimal. Donne un quotient de compression de 2:1 ou 3:1.
  - Souvent suivi d'un codage entropique.
- ▶ Voir exemple MATLAB
  - *CodageDPCM.m*
  - *RGB.jpg*

# Compression sans perte: Codage entropique

- ▶ Une méthode de codage entropique compressse les données à l'aide de statistiques sur les données à compresser.
- ▶ Le codage entropique est issue de la théorie de l'information.

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
  - Code à longueur variable;
  - Applications: JPEG, MPEG, MP3, ZIP, gzip
  - Code déterminé à partir d'une estimation des probabilités d'apparition des symboles dans la source;
  - Un code plus court est assigné aux symboles les plus fréquents;
  - Chaque code n'est pas préfixe à un autre (e.g. 10,110);
  - Les codes sont déterminés à l'aide d'un arbre.

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
  - Construction du code
    1. On doit d'abord calculer le nombre d'occurrences de chaque symbole;
    2. On construit l'arbre des codes selon le principe suivant:
      - Un poids est attribué à chaque symbole en fonction de son nombre d'occurrences;
      - On associe à chaque fois les deux nœuds de plus faibles poids pour donner un nœud ayant un poids équivalent à la somme des nœuds groupés.
      - On associe ensuite 0 aux branches de droite et 1 aux branches de gauche. On parcourt l'arbre jusqu'à chaque feuille pour obtenir les codes binaires de compression.

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
- ▶ Exemple:
  - Soit le code binaire (4 bits) suivant à compresser:
  - 0101**01110110010111110101**0111
  - Trouver le code de Huffman

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
- ▶ Propriété:
  - L'espérance de la longueur d'un code  $C$  (nombre moyen de bits pour coder les symboles  $x$ ) est

$$L(C) = \sum_{x \in \Omega} p(x) \cdot l(x)$$

où  $l(x)$  est la longueur du mot code  $C(x)$  associé au symbole source  $x$ ,  $p(x)$  est la probabilité d'un symbole  $x$ , et  $\Omega$  est l'ensemble des symboles sources.

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
- ▶ Optimalité
  - L'entropie de Shannon est la mesure du plus court code possible d'obtenir en théorie. L'entropie est définie comme étant:

$$H(x) = - \sum_{p(x)>0} p(x) \log_2 p(x)$$

# Compression sans perte: Codage entropique

- ▶ L'entropie peut être expliquée comme le nombre moyen de questions requises pour identifier un symbole en procédant par dichotomie.
  - Exemple #1: Si on a 4 symboles équiprobables (A,B,C,D). Pour identifier un symbole inconnu (supposons que c'est **A**).
    - Par dichotomie, on demandera, est-ce que le symbole est A ou B. Si oui, c'est A ou B. Sinon, c'est C ou D. Dans l'exemple, la réponse est oui.
    - Ensuite, on demandera, est-ce que le symbole est A. Si oui, c'est A, sinon c'est B. Dans notre cas, c'est oui, donc **A**.

# Compression sans perte: Codage entropique

## ▶ L'entropie ...

- Il a donc fallu deux questions, donc 2 bits. 2 bits, c'est  $\log_2(4)$ , où 4 est le nombre de symboles. Les symboles étant équiprobables, il faudra toujours deux bits, donc en moyenne, on aura  $(2+2+2+2)/4$ , soit 2 bits par symbole.

# Compression sans perte: Codage entropique

## ▶ L'entropie ...

- Exemple #2: Si on a 4 symboles (A,B,C,D) avec les probabilités suivantes: A: 1/2, B: 1/4, C: 1/8, D: 1/8.
- Pour identifier un symbole inconnu (supposons que c'est A).
- Par dichotomie, on demandera, est-ce que le symbole est A. Si oui, c'est A. Sinon, B, C ou D. Dans l'exemple, la réponse est oui.
- Un bit est suffisant pour identifier A.

# Compression sans perte: Codage entropique

## ▶ L'entropie ...

- Exemple #2: Si on a 4 symboles (A,B,C,D) avec les probabilités suivantes: A: 1/2, B: 1/4, C: 1/8, D: 1/8.
- Pour identifier un symbole inconnu (supposons que c'est maintenant B).
- Par dichotomie, on demandera, est-ce que le symbole est A. Si oui, c'est A. Sinon, B, C ou D. Dans l'exemple, la réponse est non.
- On demandera alors, est-ce que c'est B. Si oui, c'est B, sinon, c'est C ou D. Dans l'exemple, la réponse est oui. Donc, on cherchait B.

# Compression sans perte: Codage entropique

## ▶ L'entropie ...

- Exemple #2: Si on a 4 symboles (A,B,C,D) avec les probabilités suivantes: A: 1/2, B: 1/4, C: 1/8, D: 1/8.
- Il faut deux bits pour identifier B. Pour C et D, il faudrait poser une troisième question.
- Donc, 1 bit pour A, 2 bits pour B, et 3 bits pour C et D. Il faut en moyenne en tenant compte des probabilités:  $(1*0.5+2*0.25+3*.0125+3*0.125)= 1.75$  bits
- Si on applique

$$H(x) = - \sum_{p(x)>0} p(x) \log_2 p(x)$$

- On trouve aussi 1.75

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
- ▶ Pour se rapprocher de l'optimalité, le codage de Huffman existe en plusieurs versions:
  - Statique
  - Semi-adaptatif
  - Adaptatif
  - etc.
- ▶ Exemple MATLAB:
  - *codageHuffman.m*

# Compression sans perte: Codage entropique

- ▶ Codage de Huffman
  - Comment écrire la table? Deux façons courantes:
    1. Enregistrer la fréquence de chaque caractère + le caractère
    2. Enregistrer l'arbre. Utiliser une méthode de recherche dans un arbre. Exemple: Parcours préfixe: [https://fr.wikipedia.org/wiki/Arbre\\_binaire#Parcours\\_pr.%C3%A9fixe](https://fr.wikipedia.org/wiki/Arbre_binaire#Parcours_pr.%C3%A9fixe)
- Illustration:  
[https://en.wikipedia.org/wiki/Tree\\_traversal#/media/File:Sorted\\_binary\\_tree\\_preorder.svg](https://en.wikipedia.org/wiki/Tree_traversal#/media/File:Sorted_binary_tree_preorder.svg)

On met des 1 binaires pour les nœuds qui ne sont pas des feuilles.

# Compression sans perte: Codage entropique

- ▶ Codage arithmétique
  - Code à longueur variable;
  - Les symboles les plus utilisés sont codés avec moins de bits;
  - Code le message complet, et non symbole par symbole. Code le message par un nombre flottant.

# Compression sans perte: Codage entropique

- ▶ Codage arithmétique
  - Construction du code:
    1. On calcule la fréquence des différents symboles à coder.
    2. On construit des intervalles à l'aide des fréquences.
    3. Pour chaque symbole à coder:
      - On vérifie à quel intervalle il appartient.
      - À l'aide des bornes de cet intervalle, on construit de nouveaux intervalles entre ces bornes.
    4. Le code attribué au message est un nombre flottant dans l'intervalle final.

# Compression sans perte: Codage entropique

- ▶ Codage arithmétique
- ▶ Exemple:
  - Soit le code binaire (4 bits) suivant à compresser:
  - 0101**01110110010111110101**0111
  - Trouver le code arithmétique pour les trois premiers symboles.
- ▶ Exemples MATLAB:
  - *CodageArithmetique.m*
  - *DecodageArithmetique.m*

# Compression sans perte: Codage entropique

## ▶ Codage arithmétique

- Comment choisir le nombre flottant?
  - Il faut choisir le nombre flottant dans l'intervalle final qui requiert le moins de bits possible.
- Comment savoir si on a terminé le décodage?
  - Si le message contient un caractère de fin de message, pas de problème, on décode jusqu'à ce qu'on trouve ce caractère.
  - Sinon, on peut spécifier, par exemple, le nombre de symboles à décoder.

# Compression sans perte: Codage entropique

- ▶ Codage arithmétique
- ▶ Propriété et optimalité:
  - Si on néglige la table de fréquence des symboles, l'espérance de la longueur d'un code  $C$  (nombre moyen de bits pour coder les symboles  $x$ ) tend vers

$$L(C) = - \sum_{p(x)>0} p(x) \log_2 p(x)$$

ce qui correspond à l'entropie.

# Compression sans perte: Codage par dictionnaire

- ▶ Consiste à remplacer un symbole par un code dans un dictionnaire.
- ▶ Aussi appelé codage par substitution.
- ▶ N'est pas basé sur la fréquence des symboles.

# Compression sans perte: Codage par dictionnaire

- ▶ Codage par paire d'octets (Byte pair encoding)
  - Remplace une paire d'octets par un octet qui n'apparaît pas dans les données;
  - Fait récursivement jusqu'à ce qu'il n'y ait plus de paires d'octets qui se répètent au moins deux fois.
  - Commence par la paire d'octets la plus fréquente.
  - Application: surtout pour le texte

# Compression sans perte: Codage par dictionnaire

- ▶ Codage par paire d'octets (Byte pair encoding)
- ▶ Exemple:
  - Soit le code binaire (4 bits) suivant à compresser:
  - 0101**01110110**0101111101010111
  - Trouver le code par paire d'octets (sur 4 bits)
  - Soit le code binaire (4 bits) suivant à compresser:
  - 0000**111100000000000000001111**
  - Trouver le code par paire d'octets (sur 4 bits)

# Compression sans perte: Codage avec dictionnaire

## ▶ Codage Lempel-Ziv (LZ77)

- Utiliser dans l'algorithme DEFLATE, et dans le système de fichier NTFS.
- Utilise une fenêtre glissante (Typiquement 2, 4, ou 32 kilooctets); Cet historique constitue le dictionnaire.
- Les motifs déjà rencontrés dans la fenêtre glissante sont remplacés par une référence à leur première apparition (typiquement une paire (distance, longueur)).
- Sinon, le premier symbole du motif est retourné tel quel.

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage Lempel-Ziv (LZ77)
- ▶ Exemple:
  - Soit le code binaire (4 bits) suivant à compresser:
  - 0101**01110110010111110101**0111**01011111**
  - Trouver le code LZ77 des groupes de 4 bits avec un historique de 16 bits.

# Codage Lempel-Ziv (LZ77)

## ▶ Un peu plus de détails

- Les motifs pas encore codés peuvent être inclus dans le codage.
- Exemple:
  - accabracadabrar|rarradea
  - Code: (3,5) !
- Attention! Il existe plusieurs variantes pour le code. La position peut être exprimée par rapport au motif à coder, ou par rapport au début de la fenêtre glissante.

# Codage Lempel-Ziv (LZ77)

## ▶ Un peu plus de détails

- En pratique, il faut pouvoir distinguer les paires (distance, longueur) des symboles non codés du message. Il existe plusieurs variantes:
  - On peut donc utiliser un octet comme code spécial pour indiquer les paires (distance, longueur). Il faut donc remplacer cette octet qui apparaît dans le message à coder par l'octet du code spécial et une paire (distance, longueur) particulière.
  - On peut ajouter un bit qui indique si ce qui suit est un symbole ou une paire (distance, longueur).

# Codage Lempel-Ziv (LZ77)

- Ou dans la version originale (pas nécessairement ce qui est le plus efficace): Les motifs déjà rencontrés dans l'historique de la fenêtre glissante sont remplacés par une référence à leur première apparition (typiquement un triplet (distance, longueur, symbole)), où symbole est le symbole suivant le motif.
- Sinon, si le symbole n'existe pas dans l'historique, on le remplace par (0,0, symbole), où le symbole qui n'est pas trouvé.

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage Lempel–Ziv–Welch (LZW)
  - Le code est de longueur variable;
  - Applications: GIF, TIFF, PDF
  - Un dictionnaire est créé en fonction des séquences d'octets rencontrés dans les données à coder.
  - À chaque étape de la compression, les octets sont groupés en séquences jusqu'à ce que le prochain octet donne une séquence qui n'est pas dans le dictionnaire. Le message est codé à l'aide du code dans le dictionnaire sans le dernier octet, et la nouvelle séquence avec le dernier octet est placée dans le dictionnaire avec un nouveau code.

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage Lempel–Ziv–Welch (LZW)
  - La longueur du code en bit est déterminée en fonction du nombre d'entrée dans le dictionnaire. La longueur du code change au fur et à mesure que le dictionnaire est allongé.
  - Le message codé est sauvegardé avec le dictionnaire initial seulement. Au décodage, le dictionnaire est allongé de la même façon qu'au codage;
  - Le dictionnaire initial contient uniquement les codes pour les symboles uniques.

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage Lempel–Ziv–Welch (LZW)
  - Construction du code
    1. Construire le dictionnaire initial;
    2. Trouver la plus longue chaîne de symboles W dans le dictionnaire qui correspond aux symboles à coder;
    3. Coder W avec le code du dictionnaire correspondant et enlever W de l'entrée;
    4. Ajouter la concaténation de W et du symbole suivant dans le dictionnaire;
    5. Aller à l'étape 2.

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage Lempel–Ziv–Welch (LZW)
- ▶ Exemple:
  - Soit le code suivant à compresser:
  - AABCABACABAB# (5 symboles: A,B,C,D + #(fin))
  - Trouver le code LZW.

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage Lempel–Ziv–Welch (LZW)
- ▶ Propriété et optimalité:
  - N'est pas optimal, car le dictionnaire est construit au fur et à mesure.
- ▶ Exemple MATLAB
  - *CodageLZW.m*

# Compression sans perte: Codage avec dictionnaire

- ▶ Exemple: Format GIF (Graphics Interchange Format)
  - Lancer en 1987
  - Utilise la compression LZW
  - Utilise une table de couleur (limite de 256 couleurs différentes)
  - Deux versions 87a et 89a. 89a ajoute les arrière-plans transparents, étiquettes textes dans les images.

# Compression sans perte: Codage avec dictionnaire

## ▶ Format GIF: propriétés

- En théorie, GIF peut afficher  $256^3 = 16\ 777\ 216$  couleurs. 8 bits pour chaque canal RGB.
- Dans la pratique, chaque image GIF possède une palette de couleurs de 256 couleurs maximum.
- Chaque couleur de la palette est codée sur 3 octets (R, G, B): elle est donc choisie parmi plus de 16 millions de couleurs.

# Compression sans perte: Codage avec dictionnaire

- ▶ Format GIF: propriétés
  - Chaque pixel de l'image est codé par le numéro de la couleur dans l'index (palette).
  - L'ensemble des pixels est ensuite codé par l'algorithme LZW, donc compression sans perte.
  - Transmission entrelacée possible.

# Compression sans perte: Codage avec dictionnaire

## ▶ Structure d'un fichier GIF

byte#	hexadecimal (hex)	text or value	Meaning
0:	47 49 46		
	38 39 61	GIF89a	Header
6:	03 00	3	Logical Screen Descriptor - logical screen width in pixels
8:	05 00	5	- logical screen height in pixels
A:	F7		- GCT follows for 256 colors with resolution 3 x 8 bits/primary
B:	00	0	- background color #0
C:	00		- default pixel aspect ratio
		R    G    B	Global Color Table
D:	00 00 00	0    0    0	- color #0 black
10:	80 00 00	128    0    0	- color #1
:			:
85:	00 00 00	0    0    0	- color #10 black
:			:
30A:	FF FF FF	255    255    255	- color #255 white
30D:	21 F9		Graphic Control Extension
30F:	04		- 4 bytes of GCE data follow
310:	01		- there is a transparent background color
311:	00 00		- delay for animation: not used
313:	10	16	- color #16 is transparent
314:	00		- end
315:	2C		Image Descriptor
316:	00 00 00 00	(0,0)	- NW corner position of image in canvas
31A:	03 00 05 00	(3,5)	- image width and height in pixels
31E:	00		- end
31F:	08	8	LZW minimum code size
320:	0B	11	11 bytes LZW encoded image data follow
321:	00 51 FC 1B 28 70 A0 C1	83 01 01	
32C:	00		- end
32D:	3B		GIF file terminator

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage DEFLATE
  - Codage à longueur variable;
  - Applications: zip, gzip, PNG
  - Combinaison de LZ77 et Huffman
  - Trois modes de compression:
    - Non compressé
    - LZ77 + Huffman statique
    - LZ77 + Huffman dynamique

# Compression sans perte: Codage avec dictionnaire

- ▶ Codage DEFLATE
  - Construction du code
    1. Élimination des duplicates pour les chaînes de symboles (LZ77).
    2. Remplacer les symboles par des symboles choisis en fonction de leur fréquence d'utilisation (Huffman)

# Compression sans perte: Codage avec dictionnaire

## ▶ Codage DEFLATE

- Format des données: les données sont groupées en blocs (exemple: de 32 Kilooctets) précédés d'un entête de 3 bits:
  - 1 bit: marqueur de fin de bloc. 1 si le bloc est le dernier bloc de données. 0 sinon.
  - 2 bits: mode de codage utilisé dans le bloc: 00: non-compressé, 01: LZ77 + Huffman statique, 10: LZ77 + Huffman dynamique (le plus courant), 11: pas utilisé.

# Compression sans perte: Codage avec dictionnaire

## ▶ Codage DEFLATE

- Mode sans compression: Aucune compression n'est appliquée sur les données;
- Compression LZ77 + Huffman statique: Les arbres de Huffman sont définis à l'avance dans les spécifications de DEFLATE.
- Compression LZ77 + Huffman dynamique: Les arbres de Huffman sont créés lors de l'encodage. L'arbre est enregistré avec les données.

# Compression sans perte: Codage avec dictionnaire

- ▶ Exemple: Format PNG (Portable Network Graphics)
  - Lancer en 1996 suite à un brevet sur LZW en 1995. GIF n'était plus un format libre.
  - Pas limité à 256 couleurs. Supporte RGB 24 bits, et RGBA 32 bits. Mais ne supporte pas les animations (GIF, oui).
  - Conçu pour le transfert d'images sur internet

# Compression sans perte: Codage avec dictionnaire

- ▶ Exemple: Format PNG
  - Utilise deux étapes de compression
    1. Codage prédictif (Moins de niveaux de couleurs (symboles) sont requis et donc cela favorise la compression)
    2. DEFLATE

# Compression sans perte: Codage avec dictionnaire

## ▶ Structure d'un fichier PNG

- Signature de 8 octets: 89 50 4E 47 0D 0A 1A 0A (50 4E 47 = code ASCII de PNG)
- Groupes de données séparés en 4 parties:
  - Longueur L du groupe de données (sur 4 octets)
  - Type du groupe (sur 4 octets)
  - Les données (sur L octets)
  - Contrôle de redondance cyclique (CRC) (sur 4 octets)
  - Certains groupes sont obligatoires, d'autres optionnels.

# Compression sans perte: Codage avec dictionnaire

- ▶ Structure d'un fichier PNG
  - Groupes de données obligatoires (Type:)
    - IHDR: Entête
    - PLTE: liste des couleurs (obligatoire pour couleur indexée)
    - IDAT: Données de l'image. Celles-ci peuvent être divisées en plusieurs groupes
    - IEND: fin de l'image