Compression de données en informatique

Techniques de Compression et utilisation d'outils Cryptographiques web et mobile

Birahime DIOUF, Docteur en Télécoms et Réseaux Enseignant chercheur

Chapitre 1 : Compression de données en informatique

Plan du chapitre :

- Généralités et enjeux actuels sur la compression
- Théorie de l'information et du codage
- Techniques de compression conservative (sans perte)
  - Codage de Shannon-Fano
  - Codage de Huffman
  - Codage par plages RLE
  - Codages par dictionnaire LZ77, LZ78 et LZW
- Applications des méthodes de compression sans perte

Introduction à la compression

C'est quoi la compression et pourquoi compresser ?

- L'étude de la transmission des informations entre une source et un destinataire via un canal de communication introduit un certain nombre de questions :
  - Le canal permet de transmettre des signaux (souvent un signal électrique binaire). Il s'agit alors de transformer le message à émettre en une séquence de signaux : c'est le *codage*. Le destinataire doit être capable de décoder la séquence de signaux reçus pour pouvoir lire le message émis.
  - Par souci d'efficacité, la séquence de signaux doit être la plus courte possible. Il s'agie alors d'utiliser des codages qui minimisent la taille de la séquence émise : c'est la *compression*.

- Cette première séquence du cours sera donc consacrée au compression des données qui devenu un sujet de plus en plus important avec le nombre croissant de transactions effectuées par le Web et les applications mobiles.

Introduction à la compression

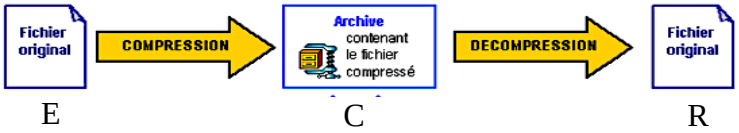
Intérêt et motivations

- Développements technologiques et exigences des utilisateurs ⟹ quantités de données plus importantes ;
  - Quels que soit le type d'info (sons, images fixes ou animées), volumes de données générés considérables ;
  - Problème d'exploitation optimale des voies de communication et des capacités de stockage est toujours resté un sujet d'actualité.
  - Stockage/exploitation sur disque dur et transport sur réseaux hauts débits ⟹ débits importants.
  - En plus la puissance des processeurs augmente plus vite que les capacités de stockage, et énormément plus vite que la bande passante des réseaux ⟹ déséquilibre entre le volume des données qu'il est possible de traiter, de stocker, et de transférer.
  - Compression présente des enjeux économiques -> aujourd'hui un grand nombre de procédés de compression.
  - Compression est utilisée majoritairement dans les applications informatiques et est une des conditions d'existence du multimédia.
- → Impératif de *réduire la taille des données*, c-à-d les compresser.

Généralités sur la compression

Définitions

- Un compresseur utilise un algorithme qui sert à optimiser les données
- Un décompresseur permet de reconstruire les données grâce à l'algorithme dual.



- Nombreux algorithmes de compressions, chacun avec sa particularité et surtout un type de données cible.
- La méthode de compression varie selon le type de données à compresser.
- On ne compressera pas de la même façon un texte, une image qu'un fichier audio...
  - algo de compression de texte exploite les répétitions du nb de caractères ou parties de phrases.
  - algo de compression d'images travaillera par exemple sur la différence entre 1 pixel et 1 autre.
- Pas de technique de compression universelle, pouvant compresser n'importe quel fichier.
- Logiciels spécialisés dans la compression/décompression :.
  - WinZip, WinRar, FilZip, QuickZip, FileRoller…
  - Différents formats de compression : .Zip, .Rar, .Tar, .Gz

Généralités sur la compression

Définitions, évaluation des algorithme de compression

- Différents algorithmes de compression sont choisis en fonction de plusieurs paramètres :
  - Le degré de réduction des données peut être évalué au moyen du quotient de compression :

$$Q = \frac{Taille\ avant\ comp}{Taille\ après\ comp}$$

  - Taux de compression, est l'inverse du quotient de compression

$$T = \frac{1}{Q} = \frac{Taille\ après\ comp}{Taille\ avant\ comp}$$ généralement exprimé en pourcentage

  - Le gain de compression est également exprimé en pourcentage.

$$G = 1 - T = \frac{Taille\ avant\ comp - Taille\ après\ comp}{Taille\ avant\ comp}$$

Les méthodes actuelles de compression permettent de gagner : 50% pour les fichiers texte, 80% pour les images fixes, 95% pour un film.

  - La vitesse de compression et de décompression.

- Ex : *Un fichier original de 2000 caractères compressé en 800 caractères*
  - *présente un quotient de compression de 2,5.*
  - *un taux de compression de 40 %, et un gain de compression de 60 %.*

Généralités sur la compression

Classification des algorithmes de compression

- Plusieurs classifications mais le critère le plus pertinent basée sur la perte des données
- Signaux engendrés par des processus physiques contiennent généralement une certaine quantité d'info *redondante*, et qui peut gaspiller les ressources de communication.
- Pour une communication efficace, il faut supprimer cette redondance avant transmission
- ⟹ Compression sans perte ou non destructible ou réversible ou conservative :
  - les données originales peuvent être reconstruites après décompression sans perte d'information
- Pour les images, audio et vidéos, le taux de compression avec les techniques sans perte est souvent insuffisant.
- ⟹ Compression avec perte ou destructible ou irréversible ou non conservative :
  - élimine quelques infos en gardant un résultat le plus proche possible des données originales.
- On dit qu'une compression est :
  - symétrique si la même méthode utilisée pour compresser et décompresser l'info.
  - asymétrique : requiert plus de temps pour l'une des 2 opérations (en général, plus de temps pour la compression que pour la décompression

Généralités sur la compression

Classification des algorithmes de compression

	Compressions sans perte (codages)	Compressions avec perte
Exemples	• delta • codes à longueur variable : VLC préfixé, Shannon-Fano, Huffman • codage arithmétique • à base de dictionnaire : Lempel-Ziv (LZ77, LZW) • par décorrélation : Run-Length Encoding (RLE), codage prédictif sans perte	• par moyennage de blocs • par transformation linéaire optimale ou de Karhunen-Loeve (KLT) • par transformée en cosinus discrète (DCT) : JPEG • par transformée en ondelettes dicrète (DWT) : JPEG 2000 • par quantification (scalaire ou vectorielle) • par détection de motifs images redondants (fractale)
Remarques	• Taux de compression limité • Aucune perte d'information	• Bon taux de compression • Perte d'information

- JPEG, MP3, MPEG : exploitent les caractéristiques de l'audition et de la visibilité humaine et dégrade l'image ou le son d'une manière quasiment imperceptible
- MP3 a rendu possible le stockage et l'échange des fichiers audio par Internet..

Compression sans perte (codage)

Principe de la compression sans perte

- Reconstitution exacte de l'information après compression.
- 3 types d'algorithmes de compression sans perte :

1. Codage statistique
   - affecter des mots de code courts aux symboles ou (caractères) fréquents,
   - affecter des mots de code longs aux symboles (caractères) plus rares.
2. Substitution de séquences :
   - Comprime les séquences de caractères identiques.
3. Utilisation d'un dictionnaire :
   - Compression d'une séquence en exploitant les répétitions de sous-séquences qui peuvent exister.

Théorie de l'information et du codage

Mesure de l'information : entropie d'une source discrète

- Nous nous intéresserons aux sources discrètes sans mémoire. La sortie d'une telle source est une séquence de symboles tirées dans un alphabet fini $A = \{a_1, \ldots, a_n\}$.
- Chaque symbole est choisie aléatoirement d'après une loi de probabilité p. $p(a_i)$ est la probabilité pour que le symbole a_i soit choisie. On a $p(a_i) \in [0; 1]$ et $\sum_{a_i \in A} p(a_i) = 1$.
- L'information propre ou incertitude sur a_i est définie par :

$$\boldsymbol{I(a_i) = \log_2 \frac{1}{p(a_i)} = -\log_2 p(a_i)}$$

- La valeur moyenne (ou espérance) de l'information propre est appelée entropie de la source et est notée $H(A)$:

$$\boldsymbol{H(A) = -\sum_{a \in A} (p(a)\ \log_2 p(a))}$$

- L'entropie est max si les symboles sont équiprobables $(p(a) = \frac{1}{n}) \Longrightarrow \boldsymbol{H_{max} = \log_2 n}$.
- A partir de l'entropie, nous pouvons déterminer la redondance : $\boldsymbol{Red = H_{max} - H(A)}$.
- L'efficacité d'une source est donnée par : $\boldsymbol{\eta_A = \frac{H(A)}{H_{max}}}$.

Théorie de l'information et du codage

Mesure de l'information : entropie d'une source discrète

- **Exemple :** Soit la source A suivante :

symboles	a_1	a_2	a_3	a_4
probabilités	0,5	0,2	0,2	0,1

- $H(A) = -(0{,}5 \log_2 0{,}5 + 0{,}2 \log_2 0{,}2 + 0{,}2 \log_2 0{,}2 + 0{,}1 \log_2 0{,}1) = \mathbf{1,761 \boldsymbol{bits/symbole}}$.
- $H_{max} = \log_2 n = \log_2 4 = 2\ \boldsymbol{bits/symbole}$.
- $Red = H_{max} - H(A) = 2 - 1{,}761 = \mathbf{0,239\ \boldsymbol{bits/symbole}}$
- $\eta_A = \frac{H(A)}{H_{max}} = \frac{\mathbf{1,761}}{\mathbf{2}} = \mathbf{0,88}$

Théorie de l'information et du codage

Code et décodabilité d'un code

- Un ensemble de mots finis C sur un alphabet A est appelé code, s'il est uniquement décodable : chaque séquence codée ne peut être décodé que d'une seule manière.
- Ex : soit une source X à valeurs dans {R, V, B, J} et le codage C : R ⟶ 0
  V ⟶ 1
  B ⟶ 10
  J ⟶ 00
- C n'est pas décodable car par exemple, si on considère la séquence codée 0110, on peut la décoder en 0 1 1 0 (RVVR) ou 0 1 10 (RVB) ou 01 10 (JB), …
- Un code est dit non singulier si 2 symboles différents ont leur codes différents.
- Un code C est uniquement décodable si son code étendu tel que $C^*(x_1x_2 \dots x_n) = C(x_1)C(x_2) \dots C(x_n)$ est non singulier.
- Un code est instantanément décodable si chaque symbole $C(x_i)$ peut être décodé sans référence aux symboles suivants.

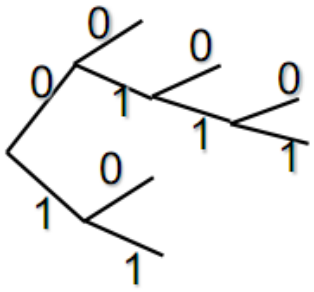
Théorie de l'information et du codage

Code et décodabilité d'un code

- Un ensemble de mots de même longueur est un code. On dit que c'est un code de longueur fixe.

Un code à longueur fixe est à déchiffrage unique.

- Pour être à décodage unique et instantané, un code à longueur variable doit remplir la *condition du préfixe* : aucun mot de code ne doit être le début d'un autre mot de code. Un code est dit *préfixe* s'il remplit cette condition.
- Un code préfixe peut être représenté par un arbre k-aire si k est la taille de l'alphabet du code.
- Les mots de code sont les suites de 0 et 1 sur les branches allant de la racine jusqu'aux feuilles de l'arbre.
- **Exemple** :
  - 00
  - 010
  - 0110
  - 0111
  - 10
  - 11



Théorie de l'information et du codage

Performances d'un code

- *Quelles performances peut on attendre d'un codage de source ?*
- Les performances d'un code peuvent être estimées par le nombre moyen de bits, (longueur moyenne d'un code) $\bar{n}$, nécessaires pour représenter un symbole de la source :

$$\bar{n} = \sum_i p(a_i) \cdot n_{a_i} \qquad n_{a_i} = \text{nombre de bits du symbole } a_i$$

- **Exemple :** soit une source X à 4 symboles, 2 codes C1 et C2, et 2 lois de probabilité :

Symboles	Code C1	Code C2	Loi 1	Loi 2
a_1	00	0	1/4	0.4
a_2	01	10	1/4	0.3
a_3	10	110	1/4	0.2
a_4	11	111	1/4	0.1

  - pour le Code C1 :
    - on trouve $\bar{n} = 2$ dans les 2 cas (loi 1 ou 2)
  - pour le Code C2 :
    - avec la Loi 1 : $\bar{n} = (1 + 2 + 3 + 3) * \frac{1}{4} = 2.25.$
    - avec la Loi 2 : $\bar{n} = 1 * 0.4 + 2 * 0.3 + 3 * 0.2 + 3 * 0.1 = 1.9.$
- Le meilleur code dépend de la loi de la source.
- Peut on trouver un code meilleur que C2 pour la source X de loi 2 ?

Théorie de l'information et du codage

Codage avec un code de longueur fixe

- Si une source a pour cardinal n, il est possible de la coder avec un code de longueur fixe m tel que : $\boldsymbol{log_2\, n \leq m < 1 + log_2\, n.}$
- L'efficacité d'un code de longueur m est égale à $\eta = \frac{H(A)}{m} \leq 1$. $\eta = 1$ sssi $m = \log_2 n = H(A)$, le cardinal de la source est une puissance de 2.

- **Exemple** : Soit une source dont l'alphabet est $A = \{0, 1, \ldots, 9\}$ munie de la loi de probabilité uniforme. On code cette source par une code en longueur fixe $m = 4$.

symboles	0	1	2	3	4	5	6	7	8	9
mots de code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

- L'efficacité du code est $H(A)/4 = \log_2 10\ /4 \approx 0{,}83$. Ce code n'est pas optimal.

- **Remarque** :
  - Si $H(A) < log_2\, n$, *il est impossible de s'approcher d'un codage optimal avec une longueur fixe*

Théorie de l'information et du codage

Codage avec un code de longueur variable : codage entropique

- Pour les codes optimaux, la longueur moyenne est égale à l'entropie de la source :

$$\bar{n} = \sum_x p(a_i) \cdot n_{a_i} = -\sum_x p(a_i) \cdot log_2\, p(a_i) = H(X)$$

- Dans ce cas : $n_{a_i} = -\, log_2\, p(a_i)$
- **Exemple** : Soit une source : $X = \{1,2,3,4\}$, alphabet binaire $\{0,1\}$

P(x=1)=1/2	$n_1 = -\, log_2(1/2) = 1$	C(1)=0
P(x=2)=1/4	$n_2 = -\, log_2(1/2) = 2$	C(2)=10
P(x=3)=1/8	$n_3 = -\, log_2(1/8) = 3$	C(3)=110
P(x=4)=1/8	$n_4 = -\, log_2(1/8) = 3$	C(4)=111

Ce code est optimal

- En pratique, on peut négliger la contrainte n_{a_i} entier, et il est donc souvent impossible d'atteindre l'optimum. Cependant, il est possible de s'en approcher.
- Premier théorème de Shannon :
  - *Pour toute source discrète sans mémoire, il existe un codage permettant de coder la source et dont l'efficacité est arbitrairement proche de 1.*

Techniques de compression sans perte

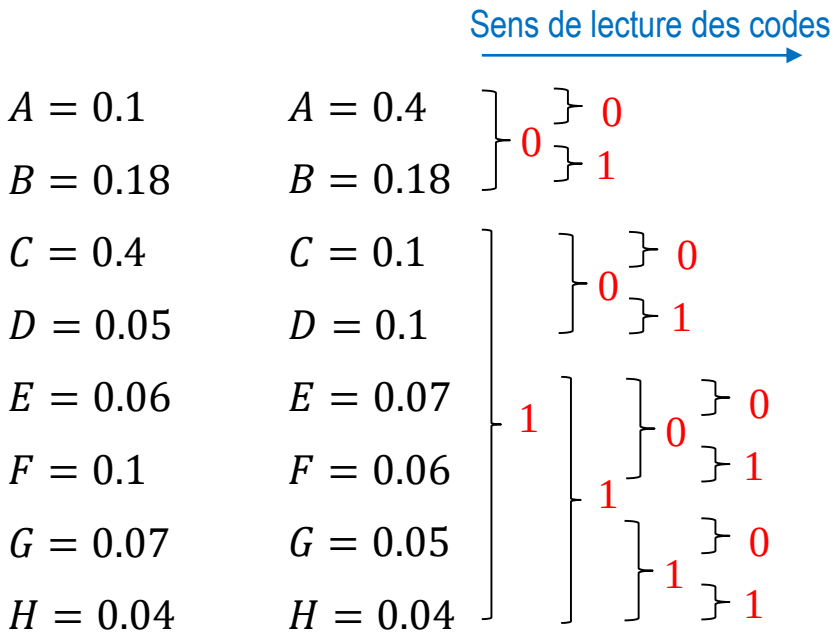
Codage de Shannon-Fano

- Algorithme :
  1. Déterminer les probabilités de chacun des symboles.
  2. Ordonner les symboles de la source dans l'ordre décroissant des probabilités.
  3. Diviser l'ensemble des symboles en 2 sous-groupes ayant une différence de cumule de probabilités minimale.
  4. Assigner un '0' pour le premier sous-groupe et un '1' pour le second sous-groupes.
  5. Répéter la 3ème étape en subdivisant les sous-groupes jusqu'à ce que chaque sous-groupe ne contienne qu'un seul symbole de la source (donc un code distinct).

Techniques de compression sans perte

Codage de Shannon-Fano

- **Exemple :**
  - Source S avec 8 symboles A, B, …, H
  - P(A) = 0.1, P(B) = 0.18, P(C) = 0.4. P(D) = 0.05, P(E) = 0.06, P(F) = 0.1, P(G) = 0.07 et P(H) = 0.04.
  - L'entropie de cette source sans mémoire est *H(S)* = 2,55 bits par symbole



Code :

C : 00
B : 01
A : 100
F : 101
G : 1100
E : 1101
D : 1110
H : 1111

$$\bar{n} = \sum_x p(x) \cdot n_x = 2.64$$

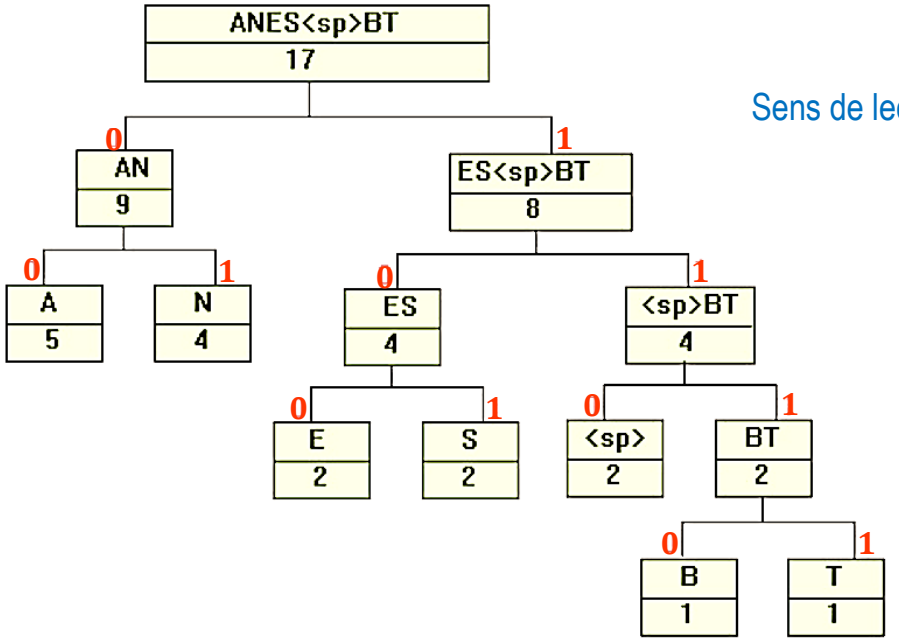
L'efficacité du codage est de :

η = 2,55/2,64 = 0,966 soit 96,6%

Techniques de compression sans perte

Codage de Shannon-Fano

- **Exemple** :
- Soit le message à transmettre : «BANANES ET ANANAS»
  - Les différents caractères sont : BANES<sp>T
  - Construction de l'arbre de Fano



Sens de lecture des codes

s	f(s)	p(s)
A	5	0.294
N	4	0.235
E	2	0.118
S	2	0.118
<space>	2	0.118
B	1	0.059
T	1	0.059
Total	**17**	

A	00
N	01
E	100
S	101
< space >	110
B	1110
T	1111

Message est codé par : 11000010001100101110100111110000100010010 1

Techniques de compression sans perte

Codage de Huffman

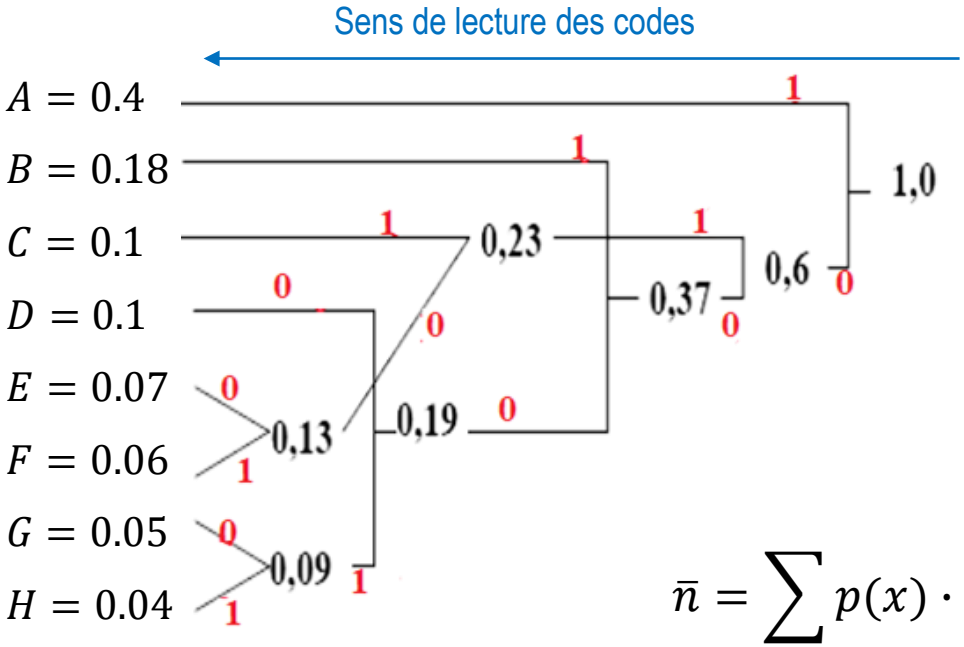
- Code plus efficace que le code Fano, décrit ici pour le cas binaire mais généralisable à d'autres alphabets.
- Le code de Huffman est optimal et il est basé sur 2 observations. Dans le code optimal :
  - on assigne moins de bits aux symboles les plus fréquents et plus de bits au symboles les moins fréquents.
  - les 2 moins fréquents symboles ont la même longueur.

- Algorithme :
  1. Déterminer les probabilités de chacun des symboles.
  2. Ordonner les symboles de la source dans l'ordre décroissant des probabilités.
  3. Regrouper les 2 symboles les moins probables en un seul nouveau en additionnant leurs probabilités et les distinguer avec '0' et '1'.
  4. Ordonner à nouveau les symboles restants avec le nouveau symbole créé.
  5. Reprendre la 3ème étape jusqu'à ce que toutes les symboles soient regroupés.

Techniques de compression sans perte

Codage de Huffman

- Code plus efficace que le code Fano, décrit ici pour le cas binaire mais généralisable à d'autres alphabets.
- **Exemple** : On considère la source de l'exemple précédant



Code :

C : 1
B : 001
A : 011
F : 0000
G : 0100
E : 0101
D : 00010
H : 00011

$$\bar{n} = \sum_x p(x) \cdot n_x = 2.64$$

L'efficacité du codage est de : η = 2,55/2,64 = 0,966 soit 96,6%

Techniques de compression sans perte

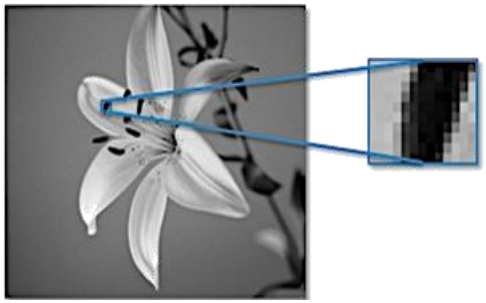
Codage des répétitions RLE (Run-Length Encoding) : codage par plages

- C'est un algorithme de compression extrêmement répandu utilisé dans le but de réduire le taux de données graphiques redondantes.
- **Principe** :
  - coder un premier élément donnant le nombre de répétitions d'une valeur
  - puis le compléter par la valeur à répéter (ou l'inverse).
- **Exemple** : "AAAAAHHHHHHHHHHHHHH"
  - La compression RLE donne "5A14H".
  - Le gain de compression est ainsi de (19-5)/19 soit environ 73,7%.
- **Autre exemple** : "BBBBBBBZZZAAAAAARBB"
  - La compression RLE donne " 7B3Z6A1R2B".

Techniques de compression sans perte

Application sur des images : rappels sur les images numériques

- Une image est une représentation visuelle voire mentale de quelque chose de naturelle, artificielle ou psychique.
- Image réelle est obtenue à partir d'un signal continu bidimensionnel : par ex un appareil photo ou une caméra, …
- Image numérique est définie comme un signal fini bidimensionnel échantillonné à valeurs quantifiées dans un certain espace de représentation. Elle est constituée de points (pixels).



179	198	204	208	208
182	202	206	208	208
183	203	206	207	208
184	202	205	207	207
184	201	203	205	206
182	199	200	203	205
180	195	197	200	203
177	192	193	198	202
175	189	191	196	201
170	190	192	189	192
167	185	189	188	191

- Différents types et représentation d'images
  - Images binaires (noir et blanc)

  - Images à nivaux de gris

  - Images couleur

0 (noir) 255 (blanc)

Techniques de compression sans perte

Application sur des images : Format d'images

- Formats d'images non compressées :
  - bmp (Bitmap) :
    - Pas de compression = pas de perte de qualité
    - Taille fichiers grande, impossibles à afficher sur internet pour une connexion bas débit
  - tiff (Tag(ged) Image File Format) :
    - Formats destination des professionnels (imprimeurs, publicitaires...).
    - Reconnu sur tout type de système d'exploitation : Windows, Linux, Mac.
    - Image de très bonne qualité, mais taille volumineuse, même si elle est inferieure à celle de BMP.
- Formats d'images compressées :

➢ Compression avec perte :

- jpeg (Joint Photographic Expert Group)
- gif (Graphics Interchange Format)

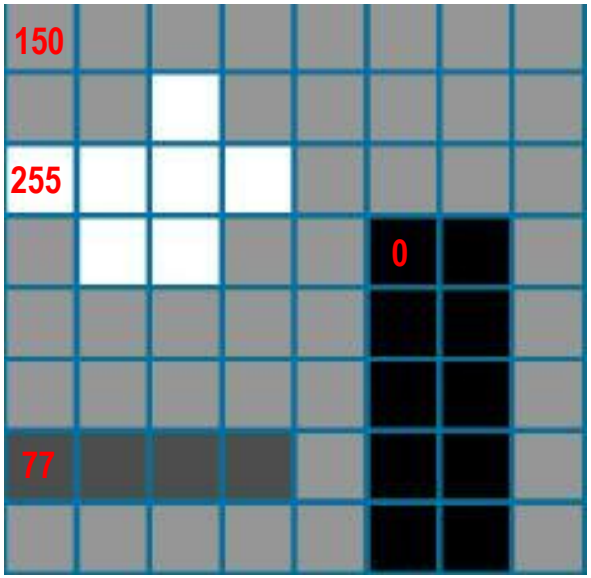
➢ Compression sans perte :

- png (Portable Network Graphics)

Techniques de compression sans perte

Application sur des images : codage RLE

- On considère l'image en niveau de gris ci-dessous :
  - Les valeurs des niveaux de gris sont marquées sur l'images



- On considère l'image en niveau de gris ci-dessous :
- Chaque niveau de gris est code sur 1 octet
- Taille de l'image : 8 x 8 x 1 octet = 64 octets
- RLE : on analyse les pixels consécutifs

-> un octet pour indiquer le niveau de gris et un octet pour indiquer le nombre de pixels ayant cette valeur

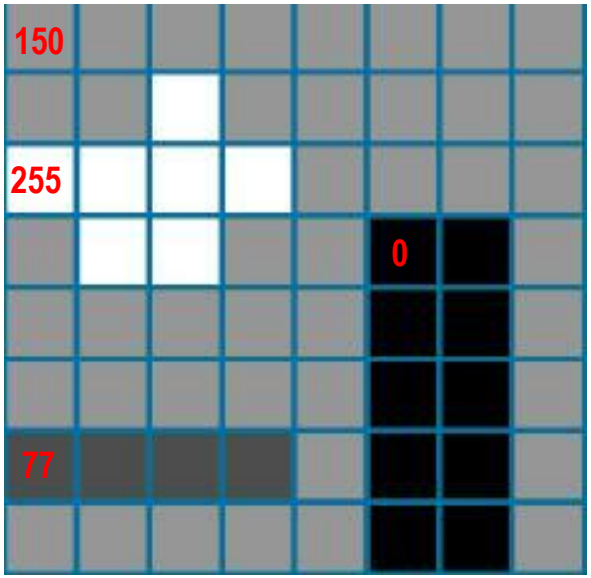
150 – 10 – 255 – 1 – 150 – 5 – 255 – 4 – 150 – 5 – 255 – 2 – 150 – 2 – 0 – 2 – 150 – 6 – 0 – 2 – 150 – 6 – 0 – 2 – 150 – 1 – 77 – 4 – 150 – 1 – 0 – 2 – 150 – 6 – 0 – 2 – 150 – 1

- Taille de l'image après compression : 38 octets
- Adapte aux images simples

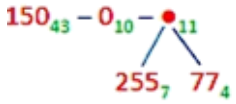
Techniques de compression sans perte

Application sur des images : codage de Huffman

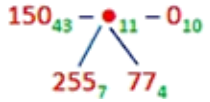
- On considère l'image en niveau de gris ci-dessous :
  - Les valeurs des niveaux de gris sont marquées sur l'images



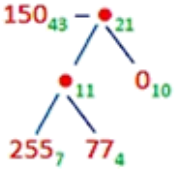
- Trier les différentes valeurs par ordre décroissant de fréquence d'apparition $150_{43} - 0_{10} - 255_{7} - 77_{4}$
- Fusionner les 2 poids minimaux

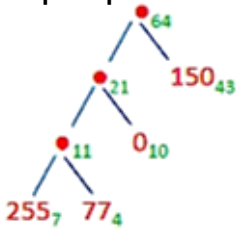


- Réordonner

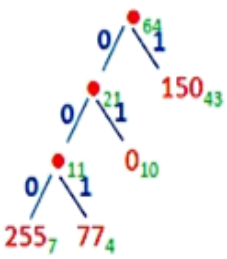


- Recommencer l'étape 2 autant de fois que possible





- Affecter des valeurs binaires



1	150
01	0
001	77
000	255

→ Niveau de gris codé sur 1 bit au lieu de 8
→ Niveau de gris codé sur 2 bits au lieu de 8
→ Niveau de gris codé sur 3 bits au lieu de 8
→ Niveau de gris codé sur 3 bits au lieu de 8

Taille de l'image : 96 bits (12 octets)

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Principe de la méthode par dictionnaire :
- Basé sur le fait qu'une séquence de caractères peut apparaître plusieurs fois dans un fichier ⇒ compresse une séquence en exploitant les répétitions de sous-séquences.
- Chacun mot répété (déjà rencontré) est remplacé, dans les données compressées, par sa seule adresse dans une structure de données appelée dictionnaire.
- Pour produire un procédé capable de s'adapter à une grande variété de besoins, il est nécessaire de définir un dictionnaire spécifique aux données à traiter.
- Le dictionnaire est construit dynamiquement d'après les motifs rencontrés.
- LZ77 : introduit en 1977 par Abraham Lempel et Jacob Ziv.
  - utilise un horizon pour trouver les concordances qui aideront à la compression.
- Amélioration avec LZ78 qui utilise un dictionnaire.
  - Pour représenter une séquence répétée on utilise des tuples sous forme <index, &> où index représente l'index de la séquence répétée dans le dictionnaire et & le caractère qui la suit.
- En 1984, Terry Welch modifia LZ78 pour l'appliquer dans les contrôleurs de disques durs.
  - élimine l'utilisation des tuples et garder seulement les index des séquences.

Techniques de compression sans perte

Compression par dictionnaire : LZW

- LZW se distingue des méthodes statistiques pour plusieurs raisons :
  - Algorithme LZW : une amélioration de LZ77 et LZ78
  - LZW est un algorithme d'apprentissage, puisque les séquences répétitives de symboles sont dans un premier temps détectées puis compressées seulement lors de leurs prochaines occurrences.
  - Le fichier comprimé ne stocke pas le dictionnaire qui est reconstruit lors de la décompression.
  - On émet à la place des séquences, leurs adresses dans le dictionnaire. La sortie de l'algorithme sera un fichier composé de symboles et d'indices.
  - blocs de tailles variables → codes de taille fixe (plus petite).
  - Compresseur universel.
  - LZW est nettement plus performant en moyenne que les algorithmes statistiques puisqu'il permet d'obtenir des gains plus élevés sur la majorité des fichiers.

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Description de LZW
- La structure de données la plus importante est ce dictionnaire.
  - On attribue une case numérotée à chaque séquence.
  - Le dictionnaire est pré-initialisé avec tous les symboles de l'alphabet du signal à coder.

Dictionnaire utilisé par LZW pour représenter les séquences rencontrées

Indice ou adresse	Chaîne mémorisé	commentaires
0, …, N	N éléments prédéfinis	La table est initialisée avec un certain nombre de symboles prédéfinis, par ex les lettres de l'alphabet ou les nuances de gris des pixels, ou couleurs de palette
N+1, ….	Les séquences qu'on a rencontrées au cours de la compression	On ajoute toutes les séquences qu'on rencontre

- Pour écrire l'algorithme, on définit 3 fonctions sur le dictionnaire (Dico) :
  - contient (séq) renvoie vrai si le dictionnaire contient cette séquence,
  - position (séq) renvoie l'indice de cette séquence ou -1 si elle est absente du dictionnaire,
  - ajouter (séq) ajoute la séquence dans la 1ère position libre, sauf si elle fait déjà partie du dico.

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Algorithme de compression LZW

```
initialiser le Dico avec tous les symboles possibles
(ex : tous les codes ascii)
p ← vide
Tant Que (c ← lire 1 caractère) faire
    sequence ←  p + c
    si sequence dans Dico alors
        p ←  sequence
    Sinon
        Coder p
        Ajouter sequence dans Dico
        p ← c
    fin (si)
fin (TQ)
Coder p
```

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Compression
- Ex : "ABBBC" à encoder en LZW un dictionnaire initial réduit aux 5 premières lettres :
  - dico[0] = 'A', dico[1] = 'B', dico[2] = 'C', dico[3] = 'D', dico[4] = 'E'.
- Voici les étapes du codage :

étape	préc. au début du TQ	caractère lu : c	séquence	dans dico ?	décision	dico	code	préc. final
1	vide	A	A	oui				A
2	A	B	AB	non	ajout	dico[5] = AB	0/3	B
3	B	B	BB	non	ajout	dico[6] = BB	1/3	B
4	B	B	BB	oui				BB
5	BB	C	BBC	non	ajout	dico[7] = BBC	6/3	C
6	C	eof					2/3	

- On doit coder 0 1 6 et 2 chacun sur 3 bits. Cela donne donc la séquence "000001110010"
- $\Longrightarrow$ 12 bits pour coder le message au lieu de 5*8 = 40 bits sans compression (asccii).

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Algorithme de décompression LZW

```
initialiser le Dico avec tous les symboles possibles
(exactement ceux du codeur)
p ← vide
Tant Que (code ← lire 1 code)
   si (code hors du Dico) alors
        courante ← p + p[0] (1er caractère de p)
    sinon
        courante ← Dico[code]
    Fin (si)
    afficher courante
    c ← 1er caractère de courante (celui de gauche)
    nouv ← p + c
    Si (nouv non dans Dico) alors
        Ajouter nouv
    Fin(Si)
    p ← courante
Fin (Tant Que)
```

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Décompression
- Voici les étapes du décodage de résultat précédent :
- Séquence "000001110010" à décoder

étape	précédente	taille du dico => nbits	code	test et courante	c	nouvelle et dico
1	vide	5 entrées => code 3 bits	0/3	ok : A	A	vide+A, dico[0] = A
2	A	5 => lire 3 bits	1/3	ok : B	B	dico[5] = AB
3	B	6 => 3 bits	6/3	hors dico : BB	B	dico[6] = BB
4	BB	7 entrées => 3 bits	2/3	C	A	dico[7] = CA
5	C		eof			

- On trouve : "ABBBC".

Techniques de compression sans perte

Compression par dictionnaire : LZW

- Avantages et inconvénients
- Avantages :
  - Méthode excellente lorsque les données comportent des répétitions variées comme texte en français ou en anglais.
  - Le dictionnaire est reconstruit chez le récepteur son l'envoyer.
  - Taux de compression d'au moins 50%.
- Inconvénients :
  - Introduit des éléments supplémentaires dans le dictionnaire qui peuvent être peu utilisés.
  - Le dictionnaire peut être dans l'état plein.
  - La recherche dans le dictionnaire consomme du temps.

Techniques de compression sans perte

Applications des techniques de compressions sans perte

- Nombreuses applications :
  - programmes exécutables, documents texte, code source
  - Format deflate basé sur la compression sans perte combinée LZ avec Huffman, utilisé par formats d'archive de fichier ZIP, GNU outil gzip et formats de fichier image PNG images.
  - LZW : utilisé par formats de fichier d'image GIF et compress d'Unix.
  - LZMA : rapport très élevé de compression, utilisé par 7zip.
  - Format bzip2 - Combine Burrows-Wheeler transformée (BWT) avec RLE et codage de Huffman. Taux de compression est meilleur que celui du format GZIP.
  - LZSS est utilisé par WinRAR avec le codage de Huffman
  - Unix pack utilitaire utilise Huffman
  - Audio Lossless formats sont le plus souvent utilisés à des fins d'archivage ou de production.
  - Prédiction par concordance partielle (PPM) : optimisée pour comprimer le texte brut
  - Souvent utilisées comme composant dans les technologies de compression avec perte.

Techniques de compression sans perte

Applications des techniques de compressions sans perte

- Huffman : algorithme est utilisé pour la compression de fichiers dans la commande *compact* d'UNIX.

lettre	fréquence en anglais	code binaire
E	12.31	011
T	9.59	001
A	8.05	1111
O	7.94	1110
N	7.19	1100
I	7.18	1011
S	6.59	1001
R	6.03	0101
H	5.14	00001
L	4.03	11011
D	3.65	10101
C	3.20	10001
U	3.10	10000
F	2.28	00010

lettre	fréquence en anglais	code binaire
P	2.28	00011
M	2.25	01001
Y	1.88	110100
B	1.62	101001
G	1.61	101000
V	0.93	000000
W	0.25	110101
K	0.52	0000010
Q	2.29	00000110
X	0.20	000001111
Z	0.09	0000011100
J	0.10	0000011101