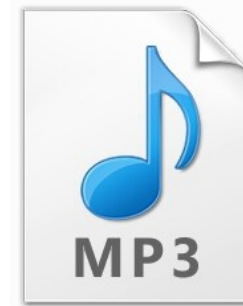
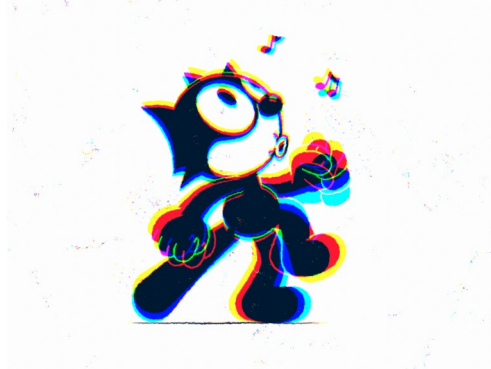
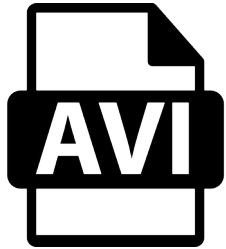


# Base de programmation

- BA1 Informatique  
Johan Depréter – [johan.depreter@heh.be](mailto:johan.depreter@heh.be)

- .AVI, MPEG, GIF, JPEG, FLAC, MP3...



Chapitre 10

- **Compression de données**

# Introduction

- Processus permettant de réduire la taille des données pour économiser de l'espace de stockage et améliorer les performances de transmission
- Données de + en + volumineuses
- Importance d'optimiser l'espace et l'échange de données

- Compression sans perte

Aucune perte des données d'origine, réécriture de façon concise. Pas de solution universelle

- Compression avec perte

Utilisée pour les données sonores ou visuelles où la perte d'informations est imperceptible pour l'œil humain

# Compression sans perte

- Possibilités de décompresser les données pour revenir à l'origine
- Exemples :
  - Compression de fichiers textes
  - Compression d'archives
  - Compression de fichiers de configuration

# Compression sans perte

- Basé sur la redondance, l'entropie
- Implémenté de manière statique ou sur base d'un dictionnaire

# Compression sans perte

- Quelques algorithmes :
  - Lempel-Ziv-Welch (LZW)
  - Huffman
  - Run Length Encoding (RLE)



- Très souvent utilisé avec les .GIF, les .PDF et les .TIFF
- Regroupement de symboles en chaînes
- Convertir chaînes en codes
- Codes moins de places que les chaînes

- Choix fréquents d'entrée possibles : 4096
- 0 à 255 bloqués pour les caractères uniques du fichier d'entrée
- Identification des séquences répétées dans les données et ajout à la table de codage

# LZW

BABAABAAA  
↑  
P=A  
C = empty

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA

LZW compression step 1

BABAABAAA  
↑  
P=B  
C = empty

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB

LZW compression step 2

BABAABAAA  
↑  
P=A  
C = empty

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA

LZW compression step 3

BABAABAAA  
↑  
P=A  
C = empty

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA

LZW compression step 4

BABAABAAA  
↑  
P=A  
C = A

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA

LZW compression step 5

BABAABAAA  
↑  
P=AA  
C = empty

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA
260	AA		

LZW compression step 6

# LZW

<66><65><256><257><65><260> Old = 65 S=A  
New = 66 C=A

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA

LZW compression step 1

<66><65><256><257><65><260> Old = 256 S=BA  
New = 256 C=B

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB

LZW compression step 2

<66><65><256><257><65><260> Old = 257 S=AB  
New = 257 C=A

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA

<66><65><256><257><65><260> Old = 65 S=A  
New = 66 C=A

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA

LZW compression step 4

<66><65><256><257><65><260> Old = 260 S=AA  
New = 260 C=A

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA
AA	260	AA

LZW compression step 5

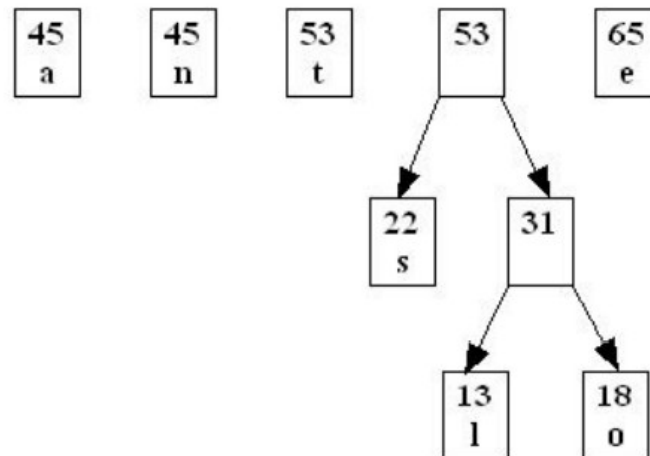
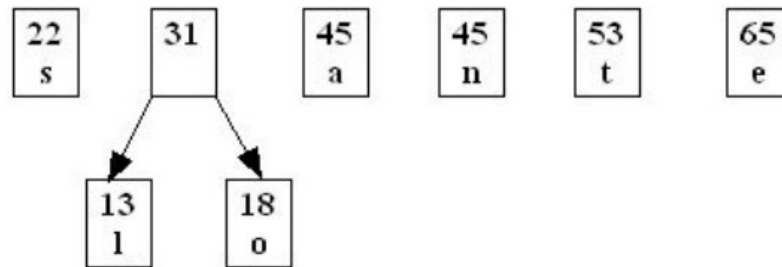
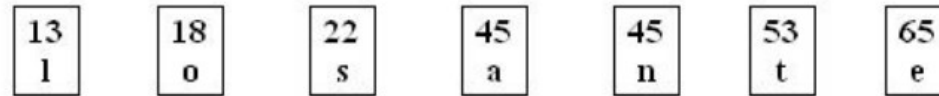
- Attribution de code en fonction de la fréquence d'apparition
  - Les symboles avec une basse fréquence sont codés sur beaucoup de bits et inversement pour les symboles à haute fréquence
- Génération d'un arbre binaire avec les branches étiquetées de 0 et de 1
- L'arbre et le message compressé doivent être envoyé pour pouvoir être décodé

# Codage de Huffman statique

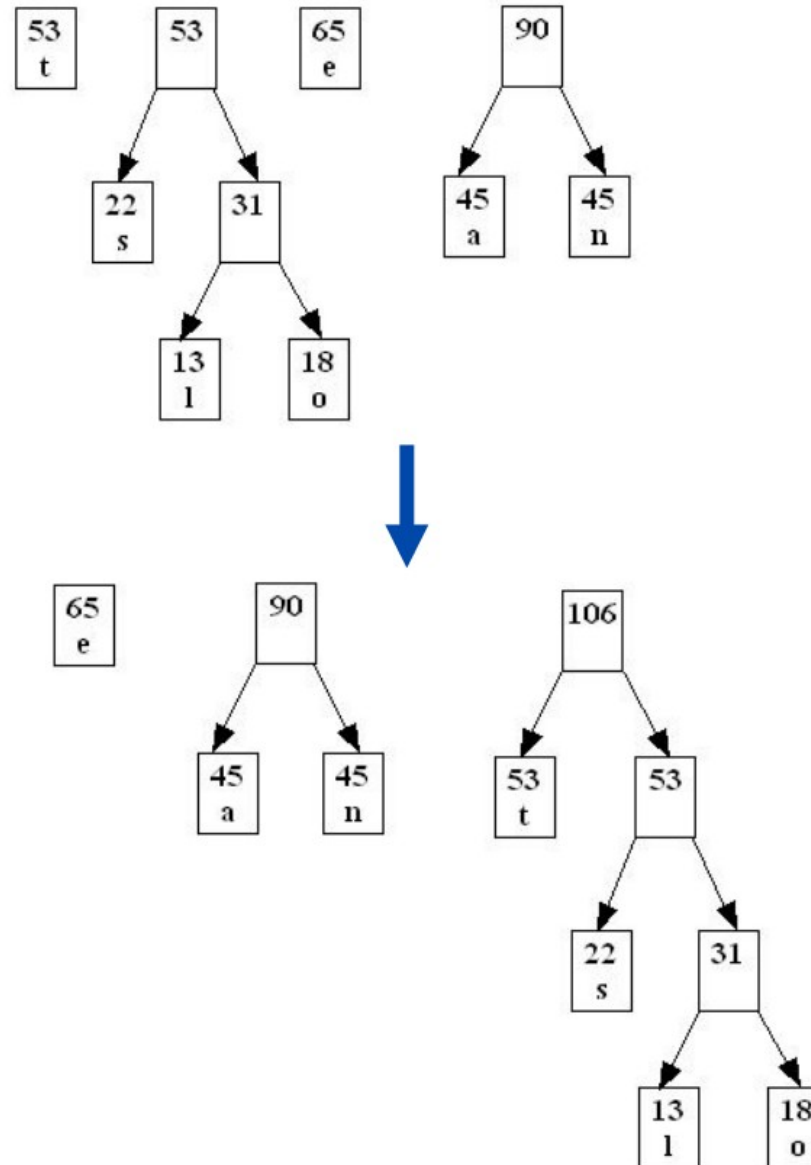
Char	a	e	l	n	o	s	t
Frequenc	4	6	1	4	1	2	5
e	5	5	3	5	8	2	3

- Créer l'arbre
- Trouver le code de chaque symbole

# Codage de Huffman

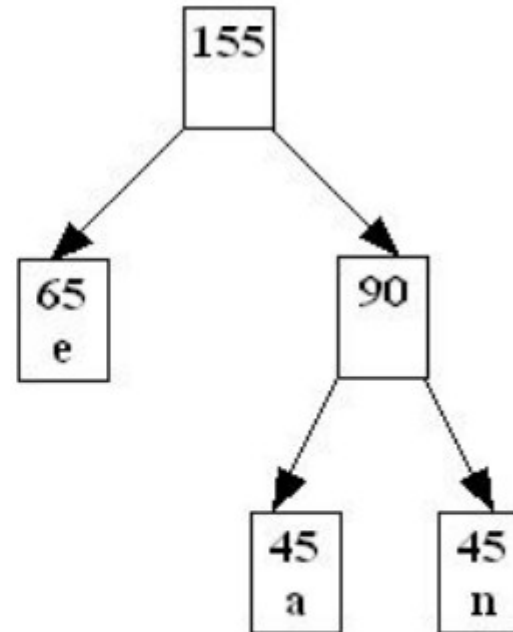
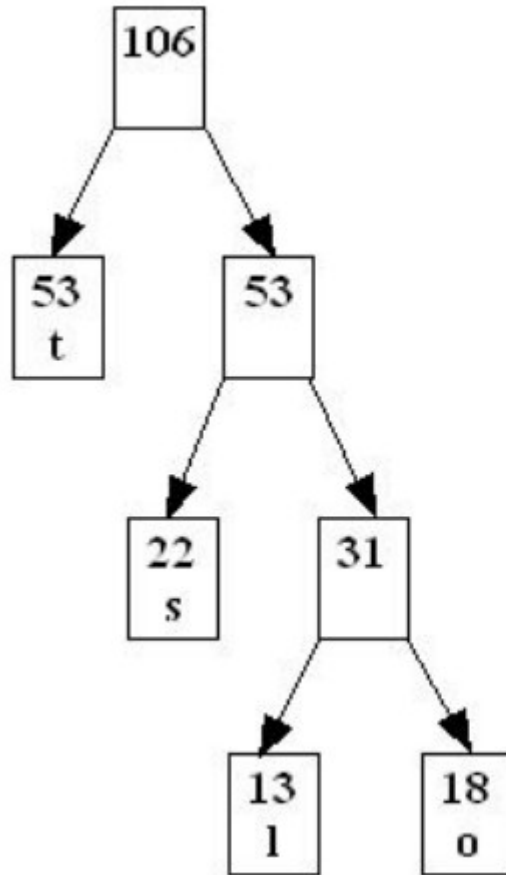


# Codage de Huffman

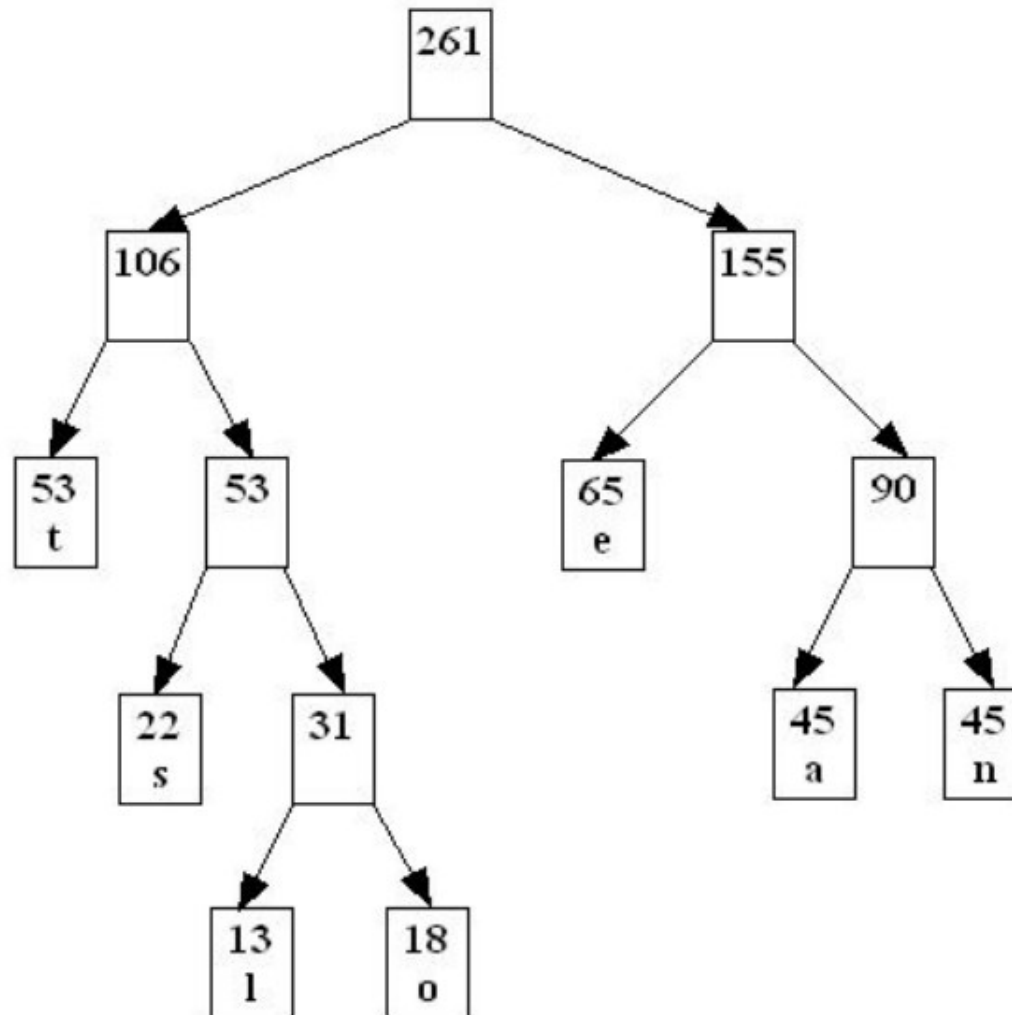




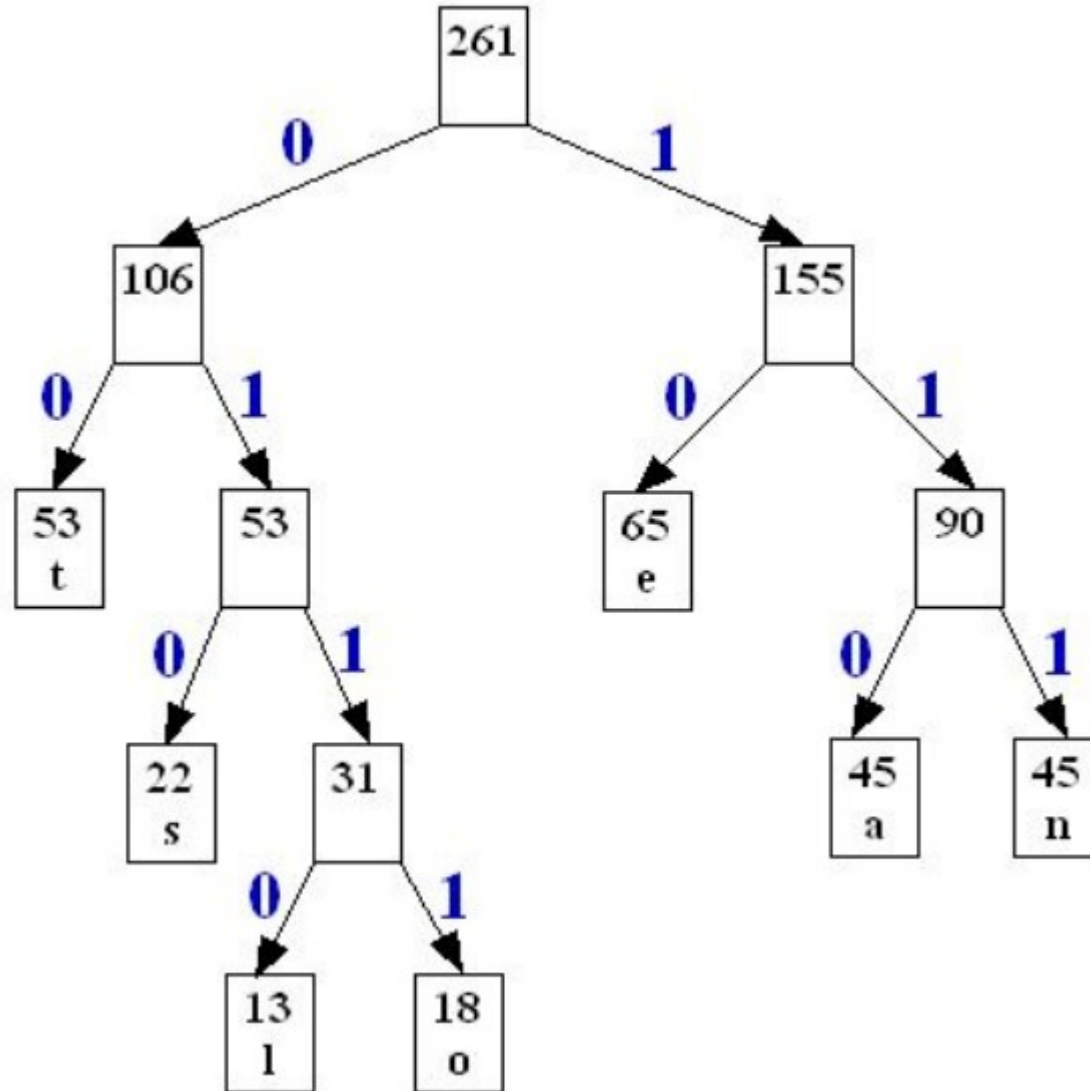
# Codage de Huffman



# Codage de Huffman



# Codage de Huffman



# Codage de Huffman statique

a : 110

e : 10

l : 0110

n : 111

o : 0111

s : 010

t : 00

- Avec les infos d'avant on peut par exemple encoder  
« tales »

00110011010010

- Et pour décoder il suffit de vérifier, à partir de la racine, bit  
par bit jusqu'à obtenir une feuille :

0110011101000

lost



# Exercices !



- En utilisant l'arbre d'Huffman réalisé durant la séance, décoder les séries de bits suivants, si c'est possible. Si ce n'est pas possible, expliquer où le decodage plante.

10100010111010001000010011

001001101111000

001001000

00100

- Encoder « states » et « notes »



# Exercices

● Réaliser l'arbre d'huffman pour les fréquences suivantes :

E : 102

A : 64

C : 35

G : 12

S : 48

T : 35

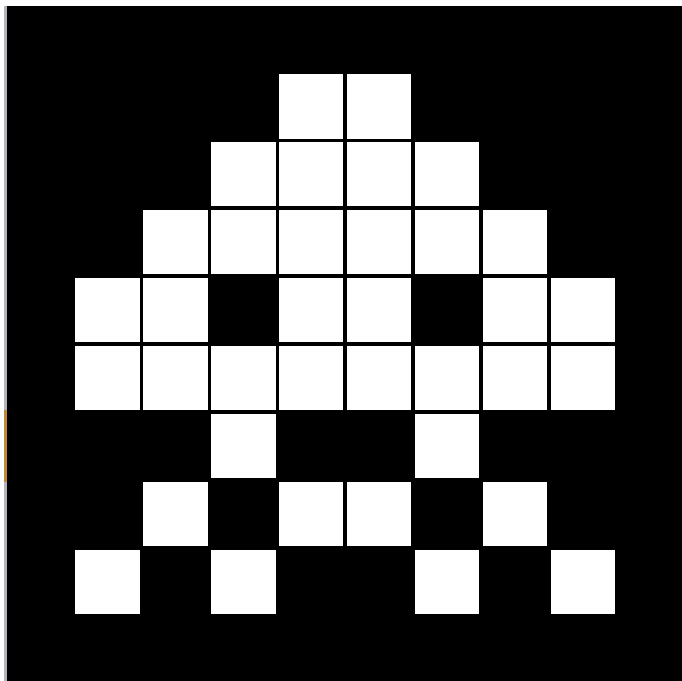
H : 9

- Compresser et décompresser la chaîne de caractères suivantes selon LZW :

TOBEORNOTTOBEORTOBEORNOTTO

# RLE

- BBBBHHDDXXXXKKKKWWZZZZ
- Va devenir : B4H2D2X4K4W2Z3

	10 1
	4 1 2 0 4 1
	3 1 4 0 3 1
	2 1 6 0 2 1
	1 1 2 0 1 1 2 0 1 1 2 0 1 1
	1 1 8 0 1 1
	3 1 1 0 2 1 1 0 3 1
	2 1 1 0 1 1 2 0 1 1 1 0 2 1
	1 1 1 0 1 1 1 0 2 1 1 0 1 1 1 0 1 1
	10 1

# Compression avec perte

- Les données répétitives sont remplacées par des codes plus court, afin de réduire la perte d'informations
- Les données compressées ne peuvent pas être décompressées pour récupérer exactement les données d'origine.

# Les algorithmes de compression

- JPEG (Joint Photographic Expert Group)  
Utilisé dans la compression d'image.
- MPEG (Moving Picture Experts Group)  
Utilisé dans la compression vidéo.
- MP3

# JPEG

