Cours 4. Compression des données.

A. Désilles

19 avril 2010

Résumé

Résumé

Résumé

Résumé

Résumé

1. Rappels
   - Premier théorème de Shannon
   - Compression des données
   - Algorithmes statistiques.
2. Compression par codage de Huffman
3. Méthode de Shannon-Fano
4. Méthodes à dictionnaire
   - Méthode LZW
   - Quelques remarques sur les codes à dictionnaire
5. Autres méthodes
   - Codage RLE (Run-length Encoding)

Résumé

Vocabulaire de codage de source

- Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ et de distribution de probabilité $P_S = \{p_1, \ldots, p_n\}$
- Un code est un ensemble $\{m_1, m_2, \ldots, m_n\}$ de n mots codes correspondants chacun à un symbole de l'alphabet de la source : $\forall\ i = 1, \ldots, n,\ \ m_i = m(s_i)$.
- Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

$$\bar{L} = E[L] = \sum_{i=1}^{n} p_i l_i$$

- On dit qu'un code donné est **sans préfixe** ou **instantané** si aucun mot du code n'est un préfixe d'un autre.

Vocabulaire de codage de source

- Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ et de distribution de probabilité $P_S = \{p_1, \ldots, p_n\}$
- Un code est un ensemble $\{m_1, m_2, \ldots, m_n\}$ de n mots codes correspondants chacun à un symbole de l'alphabet de la source : $\forall\ i = 1, \ldots, n,\ \ m_i = m(s_i)$.
- Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

$$\bar{L} = E[L] = \sum_{i=1}^{n} p_i l_i$$

- On dit qu'un code donné est **sans préfixe** ou **instantané** si aucun mot du code n'est un préfixe d'un autre.

Vocabulaire de codage de source

- Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ et de distribution de probabilité $P_S = \{p_1, \ldots, p_n\}$
- Un code est un ensemble $\{m_1, m_2, \ldots, m_n\}$ de n mots codes correspondants chacun à un symbole de l'alphabet de la source : $\forall\ i = 1, \ldots, n,\ \ m_i = m(s_i)$.
- Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

$$\bar{L} = E[L] = \sum_{i=1}^{n} p_i l_i$$

- On dit qu'un code donné est **sans préfixe** ou **instantané** si aucun mot du code n'est un préfixe d'un autre.

Vocabulaire de codage de source

- Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ et de distribution de probabilité $P_S = \{p_1, \ldots, p_n\}$
- Un code est un ensemble $\{m_1, m_2, \ldots, m_n\}$ de n mots codes correspondants chacun à un symbole de l'alphabet de la source : $\forall\ i = 1, \ldots, n,\ \ m_i = m(s_i)$.
- Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

$$\bar{L} = E[L] = \sum_{i=1}^{n} p_i l_i$$

- On dit qu'un code donné est **sans préfixe** ou **instantané** si aucun mot du code n'est un préfixe d'un autre.

Théorème de la borne inférieure

Théorème

Soit un code déchiffrable $\{m_1, \ldots, m_n\}$ de longueurs de mots $\{l_1, \ldots, l_n\}$. Alors la longueur moyenne de mots de code vérifie :

$$\bar{L} = \sum_{i=1}^{n} p(s_i) l_i \geq \frac{H(S)}{\log_2(d)}$$

L'égalité n'est possible que si $\forall i = 1, \ldots, n, \quad p_i = d^{-l_i}$.

Le code dont la longueur moyenne de mots atteint la borne inférieure (s'il existe) s'appelle **absolument optimal**.

Théorème de la borne inférieure

Théorème

Soit un code déchiffrable $\{m_1, \ldots, m_n\}$ de longueurs de mots $\{l_1, \ldots, l_n\}$. Alors la longueur moyenne de mots de code vérifie :

$$\bar{L} = \sum_{i=1}^{n} p(s_i) l_i \geq \frac{H(S)}{\log_2(d)}$$

L'égalité n'est possible que si $\forall i = 1, \ldots, n, \quad p_i = d^{-l_i}$.

Le code dont la longueur moyenne de mots atteint la borne inférieure (s'il existe) s'appelle **absolument optimal**.

Premier théorème de Shannon

Théorème

Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, x_n\}$ de taille n et de distribution de probabilités $P_X = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Alors il existe un procédé de codage déchiffrable dont la longueur moyenne de mots de code est aussi voisine que l'on souhaite de la borne inférieure $\frac{H(S)}{\log_2(d)}$.

Compression vs codage

- La recherche de code dont la longueur moyenne de mots est minimale est initialement motivée par le besoin de transmettre le plus rapidement possible.
- Mais en réduisant la longueur des messages on réduit également la place qu'ils occupent en mémoire.
- Ainsi les techniques de codage optimal peuvent également servir à la compression des données.

Compression vs codage

- La recherche de code dont la longueur moyenne de mots est minimale est initialement motivée par le besoin de transmettre le plus rapidement possible.
- Mais en réduisant la longueur des messages on réduit également la place qu'ils occupent en mémoire.
- Ainsi les techniques de codage optimal peuvent également servir à la compression des données.

Compression vs codage

- La recherche de code dont la longueur moyenne de mots est minimale est initialement motivée par le besoin de transmettre le plus rapidement possible.
- Mais en réduisant la longueur des messages on réduit également la place qu'ils occupent en mémoire.
- Ainsi les techniques de codage optimal peuvent également servir à la compression des données.

Compression de données : position de problème

- Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit M le nombre de symboles dans T.
- Codage à longueur fixe, R : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte T tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(T) - l(U)}{l(T)}$$

Compression de données : position de problème

- Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit M le nombre de symboles dans T.
- Codage à longueur fixe, R : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte T tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(T) - l(U)}{l(T)}$$

Compression de données : position de problème

- Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit M le nombre de symboles dans T.
- Codage à longueur fixe, R : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte T tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(T) - l(U)}{l(T)}$$

Compression de données : position de problème

- Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit M le nombre de symboles dans T.
- Codage à longueur fixe, R : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte T tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(T) - l(U)}{l(T)}$$

Compression de données : position de problème

- Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit M le nombre de symboles dans T.
- Codage à longueur fixe, R : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte T tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(T) - l(U)}{l(T)}$$

Deux approches principales

Compression sans pertes. Il s'agit de méthodes inversibles, garantissant que le message initial peut être restauré intégralement à partir du compressé. Dans ce groupe de méthodes on retrouve en particulier, le format zip utilisé pour l'archivage et le format d'images GIF, par exemple. Le taux de compression est limité par l'entropie des données.

Compression avec pertes. Il s'agit de techniques basées sur les approximations des données initiales. Lors de la restauration le message obtenu n'est pas exactement le message initial. On retrouve dans ce groupe les formats Jpeg, MP3, MPEG.

Deux approches principales

Compression sans pertes. Il s'agit de méthodes inversibles, garantissant que le message initial peut être restauré intégralement à partir du compressé. Dans ce groupe de méthodes on retrouve en particulier, le format zip utilisé pour l'archivage et le format d'images GIF, par exemple. Le taux de compression est limité par l'entropie des données.

Compression avec pertes. Il s'agit de techniques basées sur les approximations des données initiales. Lors de la restauration le message obtenu n'est pas exactement le message initial. On retrouve dans ce groupe les formats Jpeg, MP3, MPEG.

Compression sans pertes : limites théoriques

- Le théorème de la borne inférieure établit la limite de compression sans pertes
- Le nombre minimal de bits par symbole en moyenne est **l'entropie des données**.
- Le premier théorème de Shannon établit la possibilité (théorique !) d'approcher la borne inférieure avec la précision voulue.

Compression sans pertes : limites théoriques

- Le théorème de la borne inférieure établit la limite de compression sans pertes
- Le nombre minimal de bits par symbole en moyenne est **l'entropie des données**.
- Le premier théorème de Shannon établit la possibilité (théorique !) d'approcher la borne inférieure avec la précision voulue.

Compression sans pertes : limites théoriques

- Le théorème de la borne inférieure établit la limite de compression sans pertes
- Le nombre minimal de bits par symbole en moyenne est **l'entropie des données**.
- Le premier théorème de Shannon établit la possibilité (théorique !) d'approcher la borne inférieure avec la précision voulue.

Compression sans pertes : deux familles d'algorithmes

Construction de la distribution de probabilités

- Pour tout symbole s_i de l'alphabet Ω soit m_i le nombre d'occurrences de ce symbole dans le texte T.
- On peut alors définir $f_i = \frac{m_i}{M}$ la fréquence du symbole s_i.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur Ω.

Construction de la distribution de probabilités

- Pour tout symbole s_i de l'alphabet Ω soit m_i le nombre d'occurrences de ce symbole dans le texte T.
- On peut alors définir $f_i = \dfrac{m_i}{M}$ la fréquence du symbole s_i.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur Ω.

Construction de la distribution de probabilités

- Pour tout symbole s_i de l'alphabet Ω soit m_i le nombre d'occurrences de ce symbole dans le texte T.
- On peut alors définir $f_i = \frac{m_i}{M}$ la fréquence du symbole s_i.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur Ω.

Construction de la distribution de probabilités

- Pour tout symbole s_i de l'alphabet Ω soit m_i le nombre d'occurrences de ce symbole dans le texte T.
- On peut alors définir $f_i = \frac{m_i}{M}$ la fréquence du symbole s_i.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur Ω.

Exemple

- Soit T="cette échelle est belle"
- Il y a $M = 20$ symboles
- Codage en ASCII (8 bits par symbole) il occupe alors $l(T) = 8 \cdot 20 = 160$ bits.

Exemple

- Soit T="cette échelle est belle"
- Il y a $M = 20$ symboles
- Codage en ASCII (8 bits par symbole) il occupe alors $l(T) = 8 \cdot 20 = 160$ bits.

Exemple

- Soit T="cette échelle est belle"
- Il y a $M = 20$ symboles
- Codage en ASCII (8 bits par symbole) il occupe alors $l(T) = 8 \cdot 20 = 160$ bits.

T="cette echelle est belle".

s_i							
m_i							
f_i							

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e						
m_i	8						
f_i	0.4						

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l					
m_i	8	4					
f_i	0.4	0.2					

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l	t				
m_i	8	4	3				
f_i	0.4	0.2	0.15				

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l	t	c			
m_i	8	4	3	2			
f_i	0.4	0.2	0.15	0.1			

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l	t	c	b		
m_i	8	4	3	2	1		
f_i	0.4	0.2	0.15	0.1	0.05		

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l	t	c	b	h	
m_i	8	4	3	2	1	1	
f_i	0.4	0.2	0.15	0.1	0.05	0.05	

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l	t	c	b	h	s
m_i	8	4	3	2	1	1	1
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

T="cette echelle est belle".

s_i	e	l	t	c	b	h	s
m_i	8	4	3	2	1	1	1
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

L'alphabet initial est ainsi $\Omega_2 = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \simeq 2.38$.

e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

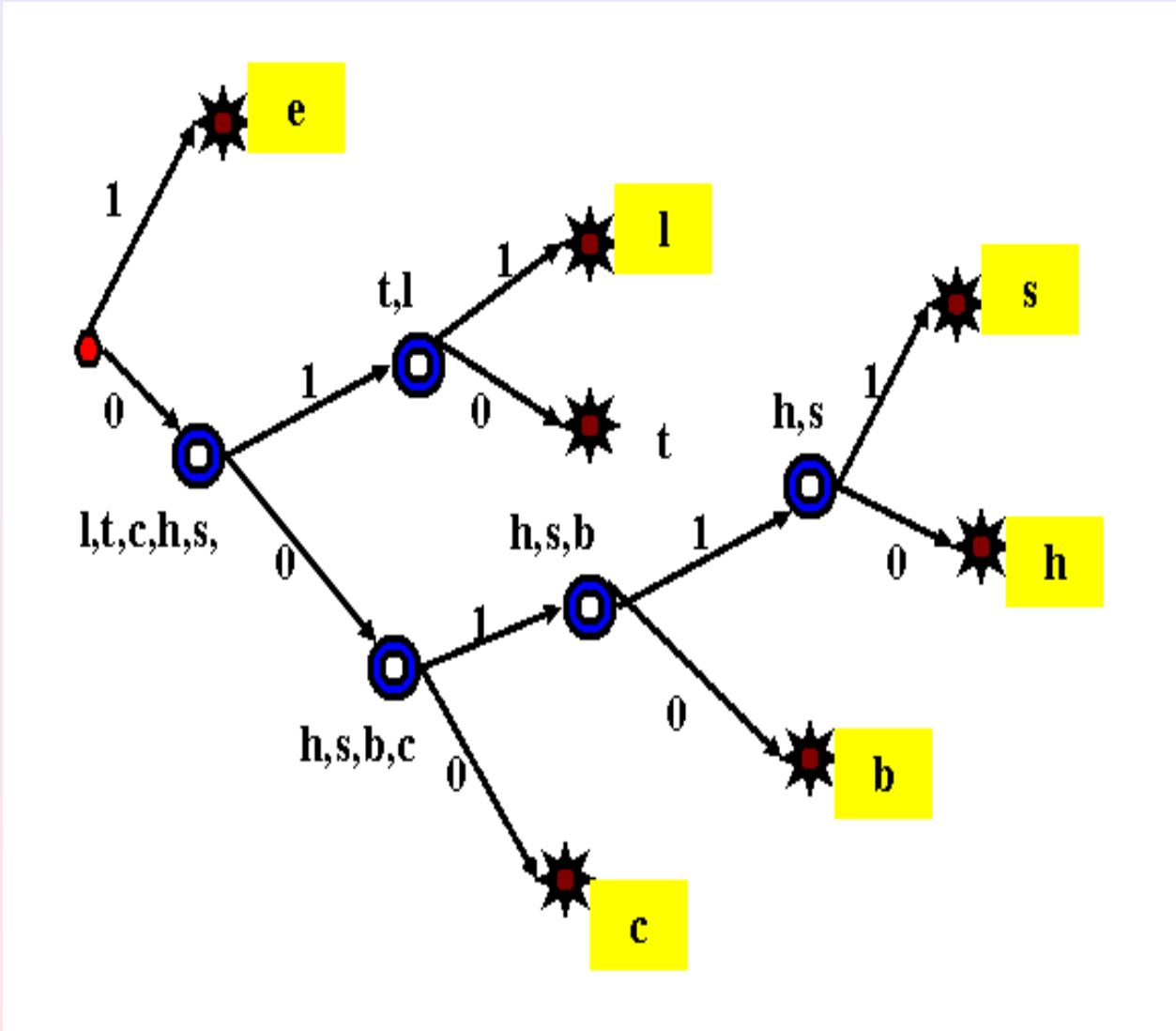
e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

Construction du code de Huffman.

En utilisant la méthode de Huffman (sans détails ici) on obtient l'arbre de code suivant



Le code qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	e	l	t	c	b	h	s
w_i	1	011	010	000	0010	00110	00111
l_i	1	3	3	3	4	5	5

La longueur moyenne de mots de code est $\bar{L} = 2.45$.

Résultat

Le message codé sera (sans les espaces)

$$U = 000101001011000001101011011110011101000101010110111$$

et on a $l(U) = 49$ Ainsi le taux de compression est

$$t = \frac{160 - 49}{160} \simeq 0.70$$

Algorithme de Shannon-Fano

1. On élimine de l'alphabet Ω les symboles dont les fréquences sont nulles et on trie la table dans l'ordre décroissant des fréquences.
2. On divise la table de fréquences en deux parties de telle sorte que les sommes des fréquences de ces parties soient aussi proches que possible. On attribue 0 à la première partie et 1 à la seconde.
3. On applique ensuite ce principe des divisions à chacune des parties recursivement jusqu'à l'obtention de parties à un seul symbole.

Algorithme de Shannon-Fano

1. On élimine de l'alphabet Ω les symboles dont les fréquences sont nulles et on trie la table dans l'ordre décroissant des fréquences.
2. On divise la table de fréquences en deux parties de telle sorte que les sommes des fréquences de ces parties soient aussi proches que possible. On attribue 0 à la première partie et 1 à la seconde.
3. On applique ensuite ce principe des divisions à chacune des parties recursivement jusqu'à l'obtention de parties à un seul symbole.

Algorithme de Shannon-Fano

1. On élimine de l'alphabet Ω les symboles dont les fréquences sont nulles et on trie la table dans l'ordre décroissant des fréquences.
2. On divise la table de fréquences en deux parties de telle sorte que les sommes des fréquences de ces parties soient aussi proches que possible. On attribue 0 à la première partie et 1 à la seconde.
3. On applique ensuite ce principe des divisions à chacune des parties recursivement jusqu'à l'obtention de parties à un seul symbole.

Exemple

Soit T="cette échelle est belle".

s_i	e	l	t	c	b	h	s
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

Première partition :
$\Omega_1 = \{e, l\}$, $P(\Omega_1) = 0.4 + 0.2 = 0.6$ et $\Omega_2 = \{t, c, b, h, s\}$, $P(\Omega_2) = 0.4$.

$P_1 =$

s_i	e	l
f_i	4/6	2/6

$P_2 =$

s_i	t	c	b	h	s
f_i	3/8	1/4	1/8	1/8	1/8

Exemple

Soit T="cette échelle est belle".

s_i	e	l	t	c	b	h	s
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

Première partition :
$\Omega_1 = \{e, l\}$, $P(\Omega_1) = 0.4 + 0.2 = 0.6$ et $\Omega_2 = \{t, c, b, h, s\}$, $P(\Omega_2) = 0.4$.

$P_1 =$

s_i	e	l
f_i	4/6	2/6

$P_2 =$

s_i	t	c	b	h	s
f_i	3/8	1/4	1/8	1/8	1/8

Exemple

Soit T="cette échelle est belle".

s_i	e	l	t	c	b	h	s
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

Première partition :
$\Omega_1 = \{e, l\}$, $P(\Omega_1) = 0.4 + 0.2 = 0.6$ et $\Omega_2 = \{t, c, b, h, s\}$, $P(\Omega_2) = 0.4$.

$P_1 =$

s_i	e	l
f_i	4/6	2/6

$P_2 =$

s_i	t	c	b	h	s
f_i	3/8	1/4	1/8	1/8	1/8

Arbre de codage

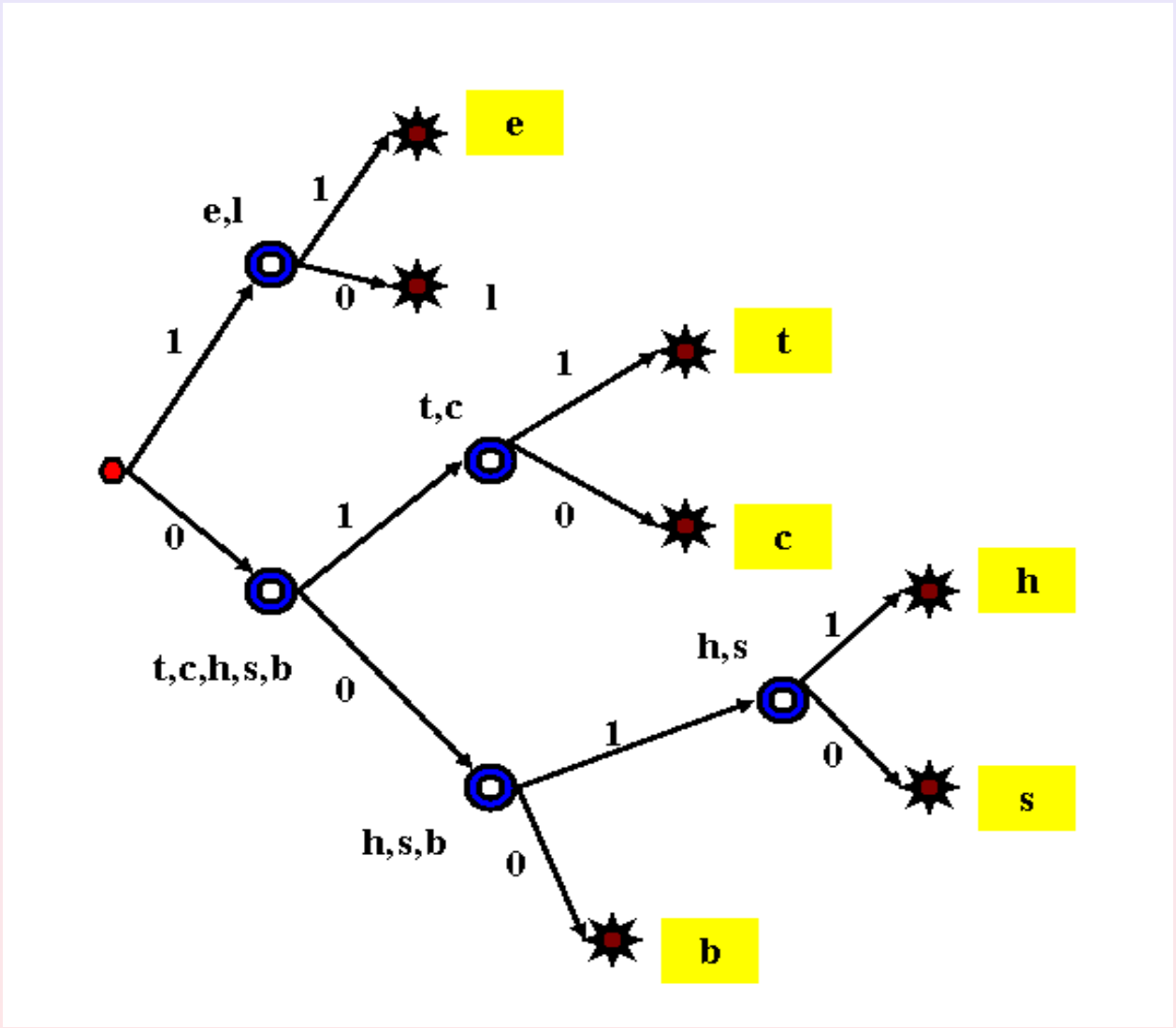


Table de code

Le code qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	e	l	t	c	b	h	s
w_i	11	10	011	010	000	0011	0010
l_i	2	2	3	3	3	4	4

La longueur moyenne de mots de code est $\bar{L} = 2.5$, on a trouvé $\bar{L} = 2.45$ avec le codage de Huffman.

Table de code

Le code qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	e	l	t	c	b	h	s
w_i	11	10	011	010	000	0011	0010
l_i	2	2	3	3	3	4	4

La longueur moyenne de mots de code est $\bar{L} = 2.5$, on a trouvé $\bar{L} = 2.45$ avec le codage de Huffman.

Le message codé sera (sans les espaces)

$$U = 0101101101111110100011111010111100100110001110101 1$$

et on a $l(U) = 50$ Ainsi le taux de compression est

$$t = \frac{160 - 50}{160} \simeq 0.69$$

Compression par codage de Huffman : Avantages

1. Compression sans pertes
2. Méthodes applicables à tout type de données numériques (images, son, texte)
3. L'arbre de code n'occupe pas beaucoup de mémoire

Compression par codage de Huffman : Avantages

1. Compression sans pertes
2. Méthodes applicables à tout type de données numériques (images, son, texte)
3. L'arbre de code n'occupe pas beaucoup de mémoire

Compression par codage de Huffman : Avantages

1. Compression sans pertes
2. Méthodes applicables à tout type de données numériques (images, son, texte)
3. L'arbre de code n'occupe pas beaucoup de mémoire

Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

Méthodes à dictionnaire. Principe

- Le principe commun consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire.
- Le dictionnaire est construit à partir du texte lui même
- Il contient toutes les séquences déjà rencontrées

Méthodes à dictionnaire. Principe

- Le principe commun consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire.
- Le dictionnaire est construit à partir du texte lui même
- Il contient toutes les séquences déjà rencontrées

Méthodes à dictionnaire. Principe

- Le principe commun consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire.
- Le dictionnaire est construit à partir du texte lui même
- Il contient toutes les séquences déjà rencontrées

LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : (i, c).
- i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- c est le dernier caractère de la séquence codée.

LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : (i, c).
- i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- c est le dernier caractère de la séquence codée.

LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : (i, c).
- i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- c est le dernier caractère de la séquence codée.

LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : (i, c).
- i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- c est le dernier caractère de la séquence codée.

LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : (i, c).
- i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- c est le dernier caractère de la séquence codée.

LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : (i, c).
- i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- c est le dernier caractère de la séquence codée.

LZ78. Méthode

- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence s est de la forme

$$s = s_d c$$

où c est le dernier caractère lu et s_d est la dernière séquence rencontrée qui appartient au dictionnaire.

LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
			14	CH	(10,H)
			15	ELL	(8,L)
			16	E␣	(1,␣)
			17	ETE	(11,E)
			18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4,B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3,␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
			15	ELL	(8,L)
			16	E␣	(1,␣)
			17	ETE	(11,E)
			18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4,B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3,␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(0,L)	15	ELL	(8,L)
			16	E␣	(1,␣)
			17	ETE	(11,E)
			18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4,B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3,␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(0,L)	15	ELL	(8,L)
3	LE	(2,E)	16	E␣	(1,␣)
			17	ETE	(11,E)
			18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4,B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3,␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(0,L)	15	ELL	(8,L)
3	LE	(2,E)	16	E␣	(1,␣)
4	␣	(0,␣)	17	ETE	(11,E)
			18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4,B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3,␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(0,L)	15	ELL	(8,L)
3	LE	(2,E)	16	E␣	(1,␣)
4	␣	(0,␣)	17	ETE	(11,E)
5	ES	(1,S)	18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4,B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3,␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

indice	Dictionnaire	Code	indice	Dictionnaire	Code
0	null		13	␣E	(4,E)
1	E	(0,E)	14	CH	(10,H)
2	L	(0,L)	15	ELL	(8,L)
3	LE	(2,E)	16	E␣	(1,␣)
4	␣	(0,␣)	17	ETE	(11,E)
5	ES	(1,S)	18	R	(0,R)
6	T	(0,T)	19	N	(0,N)
7	␣B	(4, B)	20	ELLE	(15,E)
8	EL	(1,L)	21	.	(0,.)
9	LE␣	(3, ␣)	22	␣ES	(13,S)
10	C	(0,C)	23	T␣	(6,␣)
11	ET	(1,T)	24	RE	(18,E)

LZ78 : Exemple

Le code définitif obtenu sera

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L,$$

$$1\sqcup, 11E, 0R, 0N, 15E, 0., 13S, 6\sqcup, 18E, 20.)$$

On obtient un code de 50 caractères au lieu de 56.

LZ78 : Exemple

Le code définitif obtenu sera

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L,$$

$$1\sqcup, 11E, 0R, 0N, 15E, 0., 13S, 6\sqcup, 18E, 20.)$$

On obtient un code de 50 caractères au lieu de 56.

Décodage

A chaque couple de code lu le décodeur va en effet effectuer deux actions :

1. Restituer la séquence qu'il représente en concaténant le mot dictionnaire indiqué par l'adresse du couple et le caractère contenu dans le couple ;
2. Enregistrer la nouvelle séquence dans le dictionnaire.

Décodage

A chaque couple de code lu le décodeur va en effet effectuer deux actions :

1. Restituer la séquence qu'il représente en concaténant le mot dictionnaire indiqué par l'adresse du couple et le caractère contenu dans le couple ;
2. Enregistrer la nouvelle séquence dans le dictionnaire.

Décodage

Voici le décodage du code de notre premier exemple :

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme (m_i, c) où m_i est un mot du dictionnaire à la position i et c est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors m_i par i, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère c

LZW : exemple

Voici ce qu'on obtient en appliquant cette méthode à notre exemple 'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec les 256 caractères ASCII, numérotés de 0 à 255.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"EL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 256$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 257$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LE". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 258$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'E'.

LZW : exemple

Voici ce qu'on obtient en appliquant cette méthode à notre exemple 'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec les 256 caractères ASCII, numérotés de 0 à 255.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"EL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 256$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 257$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LE". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 258$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'E'.

LZW : exemple

Voici ce qu'on obtient en appliquant cette méthode à notre exemple 'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec les 256 caractères ASCII, numérotés de 0 à 255.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"EL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 256$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 257$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LE". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 258$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'E'.

LZW : exemple

Voici ce qu'on obtient en appliquant cette méthode à notre exemple 'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec les 256 caractères ASCII, numérotés de 0 à 255.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"EL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 256$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LL". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 257$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'L'.
- On lit le caractère "L". Cette séquence est présente dans le dictionnaire, $i = 76$. On lit le caractère suivant :"LE". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 258$. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'E'.

LZW : exemple

- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"E␣". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 259$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère '␣'.
- On lit le caractère "␣". Cette séquence est présente dans le dictionnaire, $i = 32$. On lit le caractère suivant :"␣E". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 260$. On encode '␣' en le remplaçant par 32 et on reprend la lecture au caractère 'E'.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"*ES*". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 261$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'S'.
- etc

LZW : exemple

- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"E␣". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 259$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère '␣'.
- On lit le caractère "␣". Cette séquence est présente dans le dictionnaire, $i = 32$. On lit le caractère suivant :"␣E". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 260$. On encode '␣' en le remplaçant par 32 et on reprend la lecture au caractère 'E'.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"*ES*". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 261$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'S'.
- etc

LZW : exemple

- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"E␣". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 259$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère '␣'.
- On lit le caractère "␣". Cette séquence est présente dans le dictionnaire, $i = 32$. On lit le caractère suivant :"␣E". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 260$. On encode '␣' en le remplaçant par 32 et on reprend la lecture au caractère 'E'.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"*ES*". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 261$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'S'.
- etc

LZW : exemple

- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"E␣". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 259$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère '␣'.
- On lit le caractère "␣". Cette séquence est présente dans le dictionnaire, $i = 32$. On lit le caractère suivant :"␣E". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 260$. On encode '␣' en le remplaçant par 32 et on reprend la lecture au caractère 'E'.
- On lit le caractère "E". Cette séquence est présente dans le dictionnaire, $i = 69$. On lit le caractère suivant :"*ES*". Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse $i = 261$. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'S'.
- etc

LZW : exemple

indice	Dictionnaire	Code	indice	Dictionnaire	Code
256	EL	69	269	ET	69
257	LL	76	270	TT	84
258	LE	76	271	TE	84
259	E␣	69	272	E␣E	259
260	␣E	32	272	EC	69
261	ES	69	273	CH	67
262	ST	83	274	HE	72
263	T␣	84	275	ELL	256
264	␣B	32	276	LE␣	258
265	BE	66	277	␣ET	260
266	LLE	257	278	TER	271
267	E␣C	259	279	RN	82
268	CE	67	280	NE	78
			281	ELLE	275

LZW : exemple

Le code obtenu pour la phrase sera

$$69, 76, 76, 69, 32, 69, 83, 84, 32, 66, 257, 259, 67, 69, 84, 84, 259, 69, 67, 72, 256,$$

Nous avons 28 nombres entre 0 et 512 qui peuvent être codés sur 9 bits. On a ainsi 252 bits de code. le texte initial a 38 caractères ASCII, codés sur 8 bits, donc 304 bits. Bien évidemment, le taux de compression est meilleur pour un texte grand. Le décodage se fait, comme dans le cas de LZ78 de manière inverse, en restituant le disctionnaire au fur et mesure de la lecture du code.

LZW : exemple

Le code obtenu pour la phrase sera

$$69, 76, 76, 69, 32, 69, 83, 84, 32, 66, 257, 259, 67, 69, 84, 84, 259, 69, 67, 72, 256, \ldots$$

Nous avons 28 nombres entre 0 et 512 qui peuvent être codés sur 9 bits. On a ainsi 252 bits de code. le texte initial a 38 caractères ASCII, codés sur 8 bits, donc 304 bits. Bien évidemment, le taux de compression est meilleur pour un texte grand. Le décodage se fait, comme dans le cas de LZ78 de manière inverse, en restituant le disctionnaire au fur et mesure de la lecture du code.

LZW : exemple

Le code obtenu pour la phrase sera

$$69, 76, 76, 69, 32, 69, 83, 84, 32, 66, 257, 259, 67, 69, 84, 84, 259, 69, 67, 72, 256, \ldots$$

Nous avons 28 nombres entre 0 et 512 qui peuvent être codés sur 9 bits. On a ainsi 252 bits de code. le texte initial a 38 caractères ASCII, codés sur 8 bits, donc 304 bits. Bien évidemment, le taux de compression est meilleur pour un texte grand. Le décodage se fait, comme dans le cas de LZ78 de manière inverse, en restituant le disctionnaire au fur et mesure de la lecture du code.

Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

Remarques. Applications des codes à dictionnaire

- **La commande compress** sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots (codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille (codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.
- **GZIP, PKZIP, WINZIP** utilisent une variante de l'algorithme LZ77
- **Formats d'images GIF, PNG** utilisent également une variante de codage par dictionnaire.

Remarques. Applications des codes à dictionnaire

- **La commande compress** sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots (codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille (codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.
- **GZIP, PKZIP, WINZIP** utilisent une variante de l'algorithme LZ77
- **Formats d'images GIF, PNG** utilisent également une variante de codage par dictionnaire.

Remarques. Applications des codes à dictionnaire

- **La commande compress** sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots (codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille (codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.
- **GZIP, PKZIP, WINZIP** utilisent une variante de l'algorithme LZ77
- **Formats d'images GIF, PNG** utilisent également une variante de codage par dictionnaire.

Codage RLE : principe

- Cette méthode est utilisée par les formats BITMAP et GIFF.
- Sa popularité est surtout due à la simplicité de sa mise en pratique.
- Elle s'applique surtout à la compression des images

Codage RLE : principe

- Cette méthode est utilisée par les formats BITMAP et GIFF.
- Sa popularité est surtout due à la simplicité de sa mise en pratique.
- Elle s'applique surtout à la compression des images

Codage RLE : principe

- Cette méthode est utilisée par les formats BITMAP et GIFF.
- Sa popularité est surtout due à la simplicité de sa mise en pratique.
- Elle s'applique surtout à la compression des images

Compression RLE : exemple sur un texte

- On recherche dans le texte à compresser des séquences de caractères répétés
- on les remplace par des couples (N, c) où N est le nombre de répétitions et c est le caractère à répéter.
- Par exemple, la séquence

$$eeeeeeee$$

sera codée par $(8e)$.

Compression RLE : exemple sur un texte

- On recherche dans le texte à compresser des séquences de caractères répétés
- on les remplace par des couples (N, c) où N est le nombre de répétitions et c est le caractère à répéter.
- Par exemple, la séquence

$$eeeeeeee$$

sera codée par $(8e)$.

Compression RLE : exemple sur un texte

- On recherche dans le texte à compresser des séquences de caractères répétés
- on les remplace par des couples (N, c) où N est le nombre de répétitions et c est le caractère à répéter.
- Par exemple, la séquence

$$eeeeeeee$$

sera codée par $(8e)$.

RLE : exemple

- Alors le texte *tttaaattaarrrrbbb* sera codé comme : $3t3a2t2a4r3b$.
- Si chaque nombre et chaque caractère sont codés sur 8 bits on obtient ainsi un code de $12 \times 8 = 96$ bits au lieu de 136 bits pour le texte initial en ASCII.
- Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétition de caractères.
- Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères et l'augmente pour un seul caractère.
- Pour les images, cette méthode est au contraire très intéressante. Car dans les zones, même petites, de couleur constante plusieurs pixels se succèdent avec les mêmes valeurs.

RLE : exemple

- Alors le texte *tttaaattaarrrrbbb* sera codé comme : $3t3a2t2a4r3b$.
- Si chaque nombre et chaque caractère sont codés sur 8 bits on obtient ainsi un code de $12 \times 8 = 96$ bits au lieu de 136 bits pour le texte initial en ASCII.
- Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétition de caractères.
- Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères et l'augmente pour un seul caractère.
- Pour les images, cette méthode est au contraire très intéressante. Car dans les zones, même petites, de couleur constante plusieurs pixels se succèdent avec les mêmes valeurs.

RLE : exemple

- Alors le texte *tttaaattaarrrrbbb* sera codé comme : $3t3a2t2a4r3b$.
- Si chaque nombre et chaque caractère sont codés sur 8 bits on obtient ainsi un code de $12 \times 8 = 96$ bits au lieu de 136 bits pour le texte initial en ASCII.
- Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétition de caractères.
- Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères et l'augmente pour un seul caractère.
- Pour les images, cette méthode est au contraire très intéressante. Car dans les zones, même petites, de couleur constante plusieurs pixels se succèdent avec les mêmes valeurs.

RLE : exemple

- Alors le texte *tttaaattaarrrrbbb* sera codé comme : $3t3a2t2a4r3b$.
- Si chaque nombre et chaque caractère sont codés sur 8 bits on obtient ainsi un code de $12 \times 8 = 96$ bits au lieu de 136 bits pour le texte initial en ASCII.
- Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétition de caractères.
- Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères et l'augmente pour un seul caractère.
- Pour les images, cette méthode est au contraire très intéressante. Car dans les zones, même petites, de couleur constante plusieurs pixels se succèdent avec les mêmes valeurs.

RLE : exemple

- Alors le texte *tttaaattaarrrrbbb* sera codé comme : $3t3a2t2a4r3b$.
- Si chaque nombre et chaque caractère sont codés sur 8 bits on obtient ainsi un code de $12 \times 8 = 96$ bits au lieu de 136 bits pour le texte initial en ASCII.
- Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétition de caractères.
- Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères et l'augmente pour un seul caractère.
- Pour les images, cette méthode est au contraire très intéressante. Car dans les zones, même petites, de couleur constante plusieurs pixels se succèdent avec les mêmes valeurs.