Cours 4. Codage de Huffman. Application à la compression des données.

A. Désilles

27 avril 2009

Résumé

Résumé

1. Rappels
   - Premier théorème de Shannon
2. Construction de codes optimaux. Codage de Huffman.
   - Arbres et codes instantanés
   - Méthode de Huffman de construction de codes optimaux
3. Compression des données
   - Position du problème
   - Algorithmes statistiques.
   - Codage de Huffman adaptatif
   - Méthodes à disctionnaire

Résumé

1. Rappels
   - Premier théorème de Shannon
2. Construction de codes optimaux. Codage de Huffman.
   - Arbres et codes instantanés
   - Méthode de Huffman de construction de codes optimaux
3. Compression des données
   - Position du problème
   - Algorithmes statistiques.
   - Codage de Huffman adaptatif
   - Méthodes à disctionnaire

Résumé

Modèle d'une source d'information

Une source d'information X est décrite par un couple (Ω_X, P_X) où Ω_X est un alphabet fini et P_X est une distribution de probabilités sur Ω_X.

Exemple important : une source binaire

- $\Omega_X = \{0, 1\}$
- $P_X = \{p, 1-p\}$ avec $p = P_X(0)$

Modèle d'une source d'information

Une source d'information X est décrite par un couple (Ω_X, P_X) où Ω_X est un alphabet fini et P_X est une distribution de probabilités sur Ω_X.

Exemple important : une source binaire

- $\Omega_X = \{0, 1\}$
- $P_X = \{p, 1-p\}$ avec $p = P_X(0)$

Entropie d'une source. Définition.

Entropie d'une source

Soient $\Omega_X = \{x_1, \ldots, x_m\}$ l'alphabet fini d'une source et X la variable aléatoire associée t.q. $P[\omega_i] = p_i, \quad i = 1, \ldots, m$. On appelle **entropie** ou encore **quantité moyenne d'information** de la source la quantité

$$H(X) = H(p_1, p_2, \ldots, p_n) = E[h(x)] = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

L'unité de mesure de cette quantité est le "bit par symbole".

Un code

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 composés de symboles d'un autre alphabet, Ω_C t.q. $\forall\ i = 1, \ldots, n,\ m_i = m(s_i)$.
Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

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

Un code

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 composés de symboles d'un autre alphabet, Ω_C t.q. $\forall\ i = 1, \ldots, n,\ \ m_i = m(s_i)$.
Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

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

Un code

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 composés de symboles d'un autre alphabet, Ω_C t.q. $\forall\ i = 1, \ldots, n,\ \ m_i = m(s_i)$.
Soit $l_i = l(m_i)$ les longueurs des mots m_i du code. On définit alors la longueur moyenne du code par

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

Propriétés d'un code

- **Régularité** Un code $\{m_1, m_2, \ldots, m_n\}$ est dit régulier si tous les mots qui le composent sont distincts : $m_i \neq m_k, \quad \forall i \neq k$. Un code qui n'est pas régulier est dit **singulier ou irréversible**.
- **Déchiffrabilité** Un code régulier est dit déchiffrable (ou encore à décodage unique) si pour toute suite de mots de code $m^1 m^2 \ldots m^k$ il est possible de distinguer sans ambiguïté tous les mots et donc identifier les symboles $s^j, \ j = 1, \ldots, k$ composant le message.

Propriétés d'un code

- **Régularité** Un code $\{m_1, m_2, \ldots, m_n\}$ est dit régulier si tous les mots qui le composent sont distincts : $m_i \neq m_k, \quad \forall i \neq k$. Un code qui n'est pas régulier est dit **singulier ou irréversible**.
- **Déchiffrabilité** Un code régulier est dit déchiffrable (ou encore à décodage unique) si pour toute suite de mots de code $m^1 m^2 \ldots m^k$ il est possible de distinguer sans ambiguïté tous les mots et donc identifier les symboles $s^j, \ j = 1, \ldots, k$ composant le message.

Codes sans préfixe

On dit qu'un mot W est **un préfixe** d'un autre mot V s'il existe un mot U tel que $V = WU$. Autrement dit, le mot V commence par le mot W.

Un code instantané

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

Codes sans préfixe

On dit qu'un mot W est **un préfixe** d'un autre mot V s'il existe un mot U tel que $V = WU$. Autrement dit, le mot V commence par le mot W.

Un code instantané

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

Codes sans préfixe

On dit qu'un mot W est **un préfixe** d'un autre mot V s'il existe un mot U tel que $V = WU$. Autrement dit, le mot V commence par le mot W.

Un code instantané

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

Problème de codage sans bruit

Problème

Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Existe-t-il un code qui minimise la longueur moyenne de mots $\bar{L}$?

Borne inférieure

Théorème

Soit une source S d'alphabet $\Omega_S = \{s_1, \ldots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \ldots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \ldots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. 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 :

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

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

Extension de source

Définition

Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, n\}$. On appelle extension d'ordre s de la source X la source $Y = (X_1, \ldots, X_s)$ où $X_i,\ i = 1, \ldots, s$ sont les variables aléatoires indépendantes et identiquement distribuées selon la distribution de X.

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 $\frac{H(S)}{\log(d)}$.

Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. **Arbre binaire complet de profondeur** n : tous les nœuds sauf les feuilles ont exactement 2 fils.
7. **Arbre binaire incomplet** : tous les nœuds ont 2 ou 0 fils.

Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. Arbre binaire complet de profondeur n : tous les nœuds sauf les feuilles ont exactement 2 fils.
7. Arbre binaire incomplet : tous les nœuds ont 2 ou 0 fils.

Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. **Arbre binaire complet de profondeur** n : tous les nœuds sauf les feuilles ont exactement 2 fils.
7. **Arbre binaire incomplet** : tous les nœuds ont 2 ou 0 fils.

Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. **Arbre binaire complet de profondeur** n : tous les nœuds sauf les feuilles ont **exactement** 2 fils.
7. **Arbre binaire incomplet** : tous les nœuds ont 2 ou 0 fils.

Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. **Arbre binaire complet de profondeur** n : tous les nœuds sauf les feuilles ont **exactement** 2 fils.
7. **Arbre binaire incomplet** : tous les nœuds ont 2 ou 0 fils.

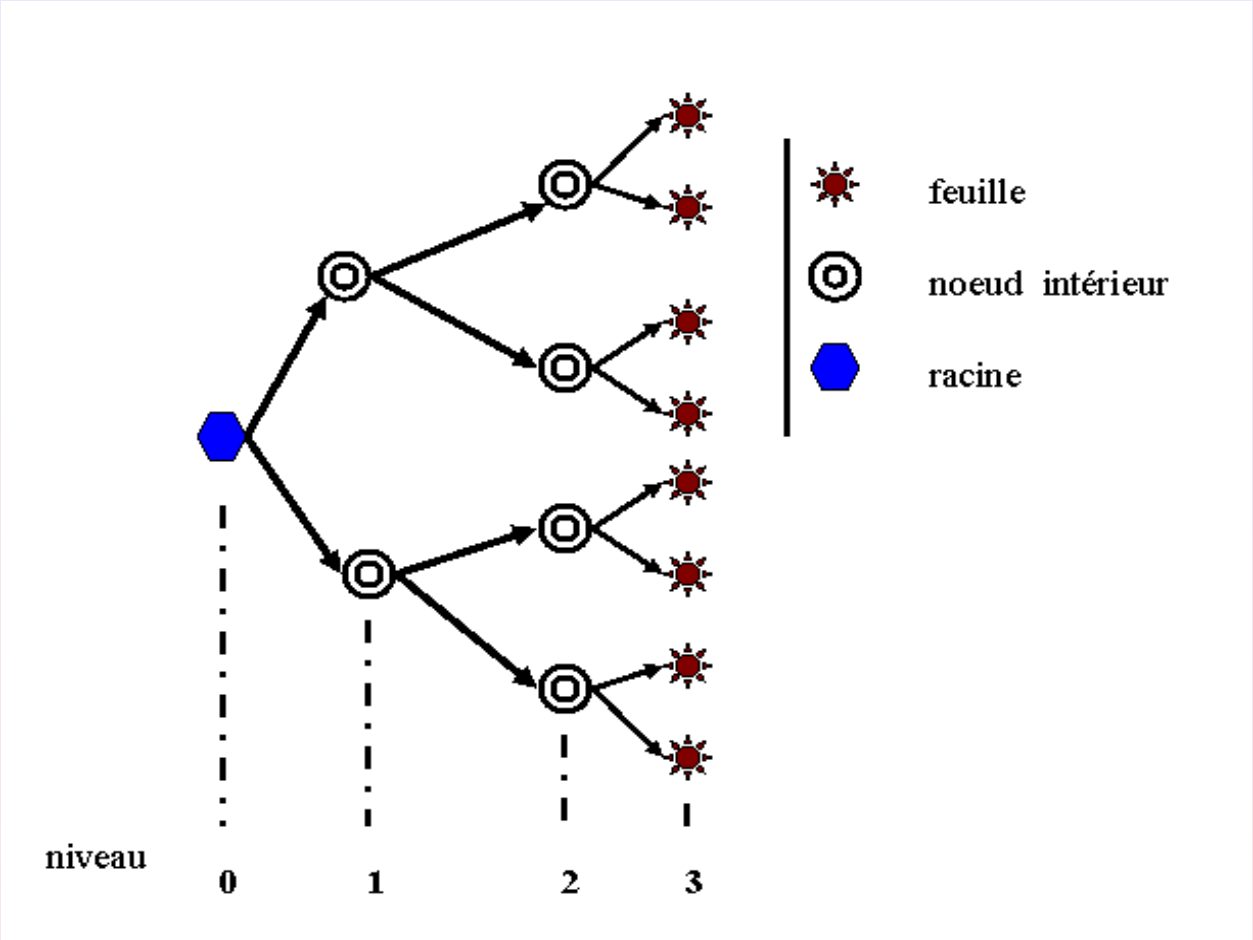
Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. **Arbre binaire complet de profondeur** n **:** tous les nœuds sauf les feuilles ont **exactement** 2 fils.
7. **Arbre binaire incomplet** : tous les nœuds ont 2 ou 0 fils.

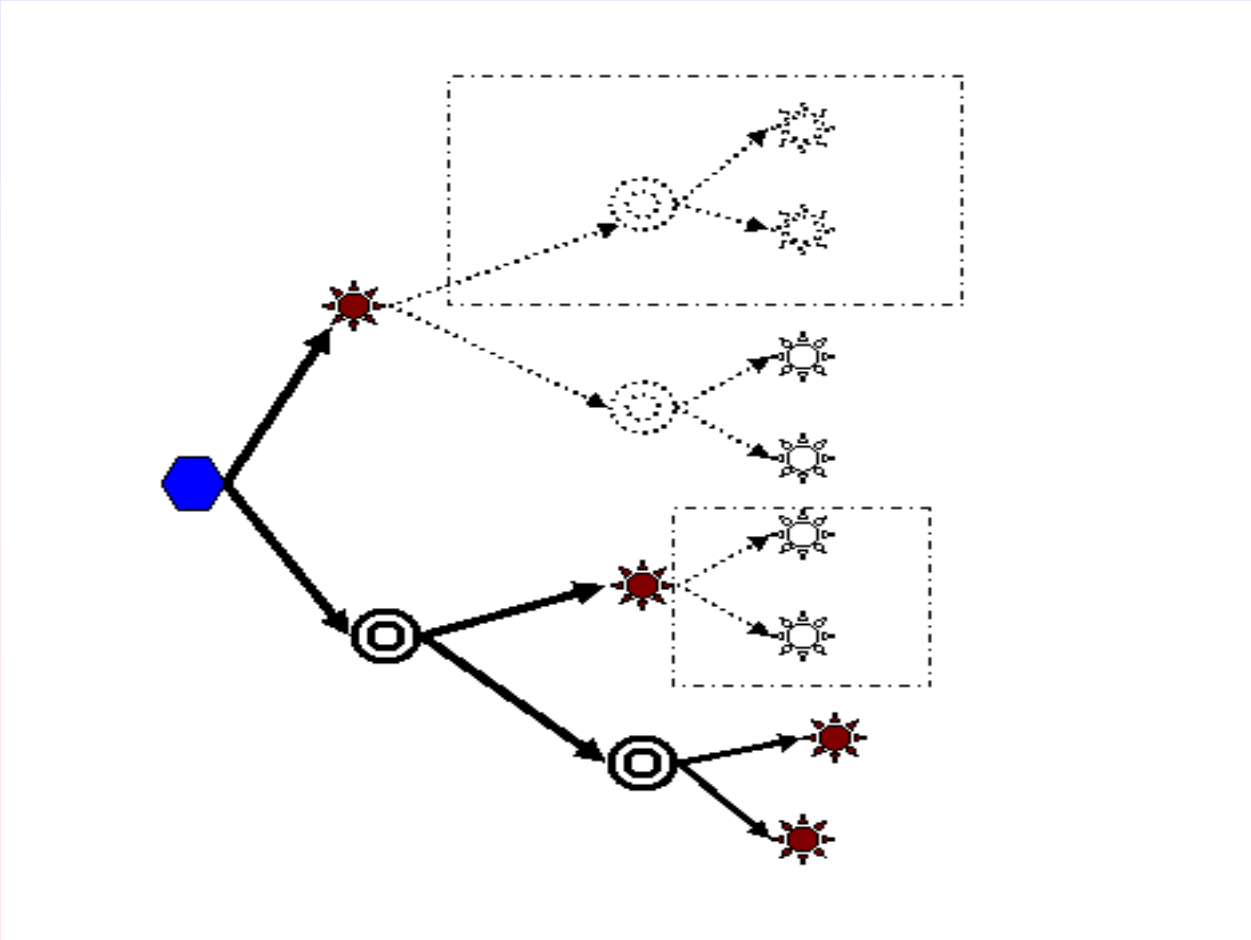
Arbres binaires

1. Un arbre binaire est un graphe orienté (N, R) où N est un ensemble de nœuds et $R \subset N \times N$ est un ensemble d'arcs.
2. Chaque nœud a au plus deux fils et chaque nœud sauf la racine a exactement un père.
3. Les nœuds qui n'ont pas de fils s'appellent feuilles de l'arbre.
4. On dit qu'un nœud est de **niveau** n si le chemin qui le relie à la racine est de longueur n.
5. On appelle **profondeur** d'un arbre la longueur du plus long chemin partant de la racine.
6. **Arbre binaire complet de profondeur** n **:** tous les nœuds sauf les feuilles ont **exactement** 2 fils.
7. **Arbre binaire incomplet** : tous les nœuds ont 2 ou 0 fils.

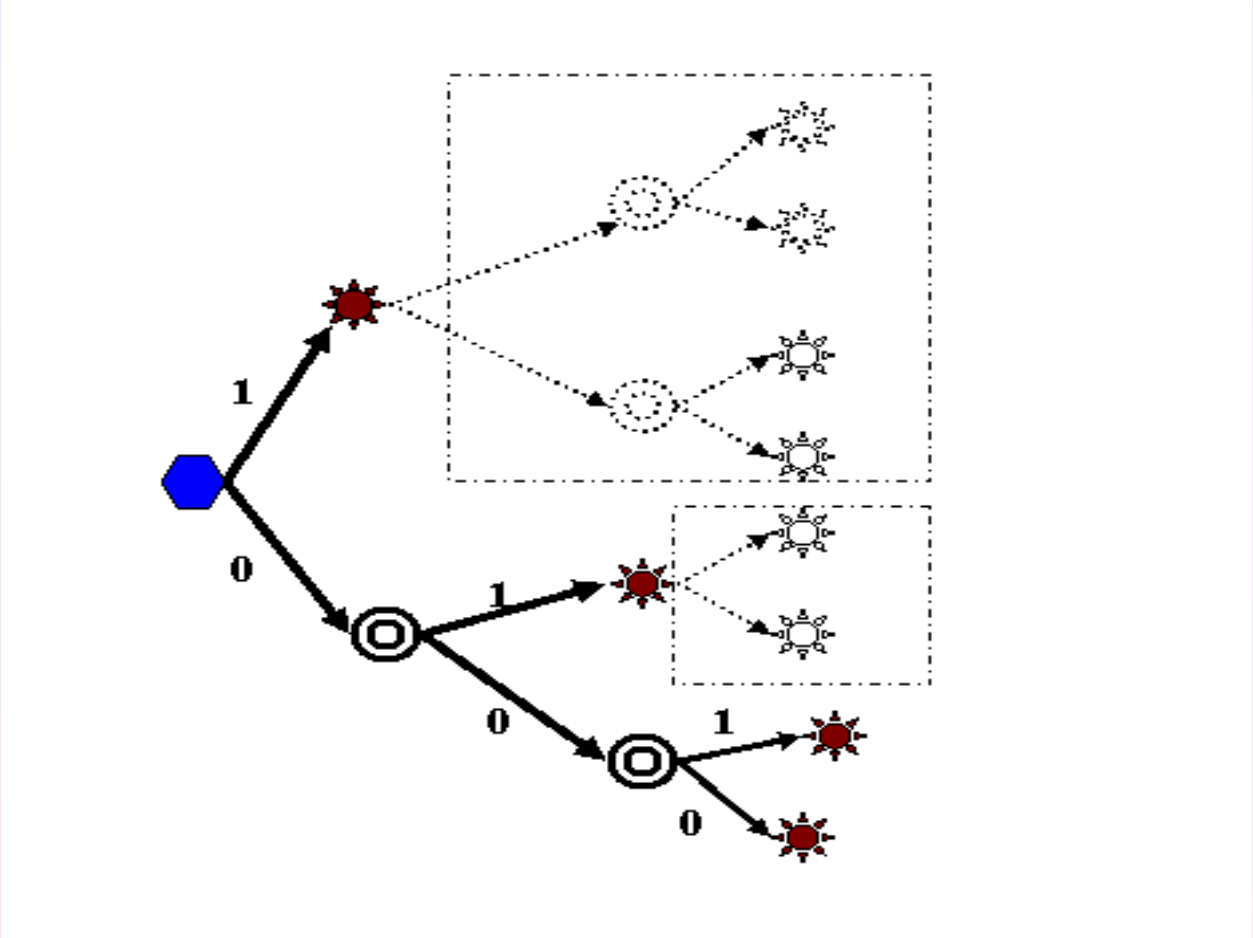
Exemple : arbre complet



Exemple : arbre incomplet



Exemple : associer un code à un arbre incimplet



Le code associé est $\{1, 01, 001, 000\}$.

Condition nécessaire d'optimalité

Lemme

Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, x_n\}$ et de distribution de probabilités $p_1 \geq p_2 \geq \cdots \geq p_n$. Soit $C = \{m_1, \ldots, m_n\}$ un code instantané associé à la source de longueurs de mots $\{l_1, \ldots, l_n\}$.
Si C est optimal pour la source X dans la **classe de codes instantanés** alors il vérifie les propriétés suivantes :

1. Les symboles les plus probables ont les longueurs de mots de code plus petites :
$$p_i \geq p_j \quad \Rightarrow \quad l_i \leq l_j$$
2. Les deux derniers symboles ont les longueurs de mots de code égales : $l_n = l_{n-1}$

Condition nécessaire d'optimalité

Lemme

Soit une source X d'alphabet $\Omega_X = \{x_1, \ldots, x_n\}$ et de distribution de probabilités $p_1 \geq p_2 \geq \cdots \geq p_n$. Soit $C = \{m_1, \ldots, m_n\}$ un code instantané associé à la source de longueurs de mots $\{l_1, \ldots, l_n\}$.
Si C est optimal pour la source X dans la **classe de codes instantanés** alors il vérifie les propriétés suivantes :

1. Les symboles les plus probables ont les longueurs de mots de code plus petites :
$$p_i \geq p_j \quad \Rightarrow \quad l_i \leq l_j$$
2. Les deux derniers symboles ont les longueurs de mots de code égales : $l_n = l_{n-1}$

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : principe

1. Une méthode récursive
2. Construit l'arbre de codage à partir des feuilles
3. A l'étape n il y a n symboles dans l'alphabet, triés dans l'ordre décroissant des probabilités
4. On choisit deux symboles les moins probables x_n et x_{n-1}.
5. On les regroupe pour former un seul symbole $w_{n,n-1}$
6. On additionne les probabilités : $P(w_{n,n-1}) = P(x_n) + P(x_{n-1})$
7. On obtient ainsi un nouvel alphabet de $n-1$ symboles, dont $w_{n,n-1}$, trié encore selon l'ordre décroissant des probabilités

Méthode de Huffman : construction de l'arbre de code

1. On applique le principe récursivement jusqu'à ce qu'il ne reste que deux symboles.
2. On leur attribue alors les préfixes $\{0, 1\}$ respectivement.
3. Soit C_{n-1} le code associé à la source de $n-1$ symboles dont le dernier est $w_{n,n-1}$
4. Les mots du code C_n associés aux symboles x_n et x_{n-1} sont construits en ajoutant au mot code du symbole $w_{n,n-1}$ respectivement 0 et 1.

Exemple

Etape 1									
x_1	0.3								
x_2	0.25								
x_3	0.2								
x_4	0.1								
x_5	0.1								
x_6	0.05								

Exemple

Etape 1									
x_1	0.3								
x_2	0.25								
x_3	0.2								
x_4	0.1								
x_5	0.1								
x_6	0.05								

Exemple

Etape 1		Etape 2							
x_1	0.3								
x_2	0.25								
x_3	0.2								
x_4	0.1	$x_{5,6}$	0.15						
x_5	0.1								
x_6	0.05								

Exemple

Etape 1		Etape 2							
x_1	0.3	x_1	0.3						
x_2	0.25	x_2	0.25						
x_3	0.2	x_3	0.2						
x_4	0.1	$x_{5,6}$	0.15						
x_5	0.1	x_4	0.1						
x_6	0.05								

Exemple

Etape 1		Etape 2		Etape 3					
x_1	0.3	x_1	0.3						
x_2	0.25	x_2	0.25						
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25				
x_4	0.1	$x_{5,6}$	0.15						
x_5	0.1	x_4	0.1						
x_6	0.05								

Exemple

Etape 1		Etape 2		Etape 3					
x_1	0.3	x_1	0.3	x_1	0.3				
x_2	0.25	x_2	0.25	x_2	0.25				
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25				
x_4	0.1	$x_{5,6}$	0.15	x_3	0.2				
x_5	0.1	x_4	0.1						
x_6	0.05								

Exemple

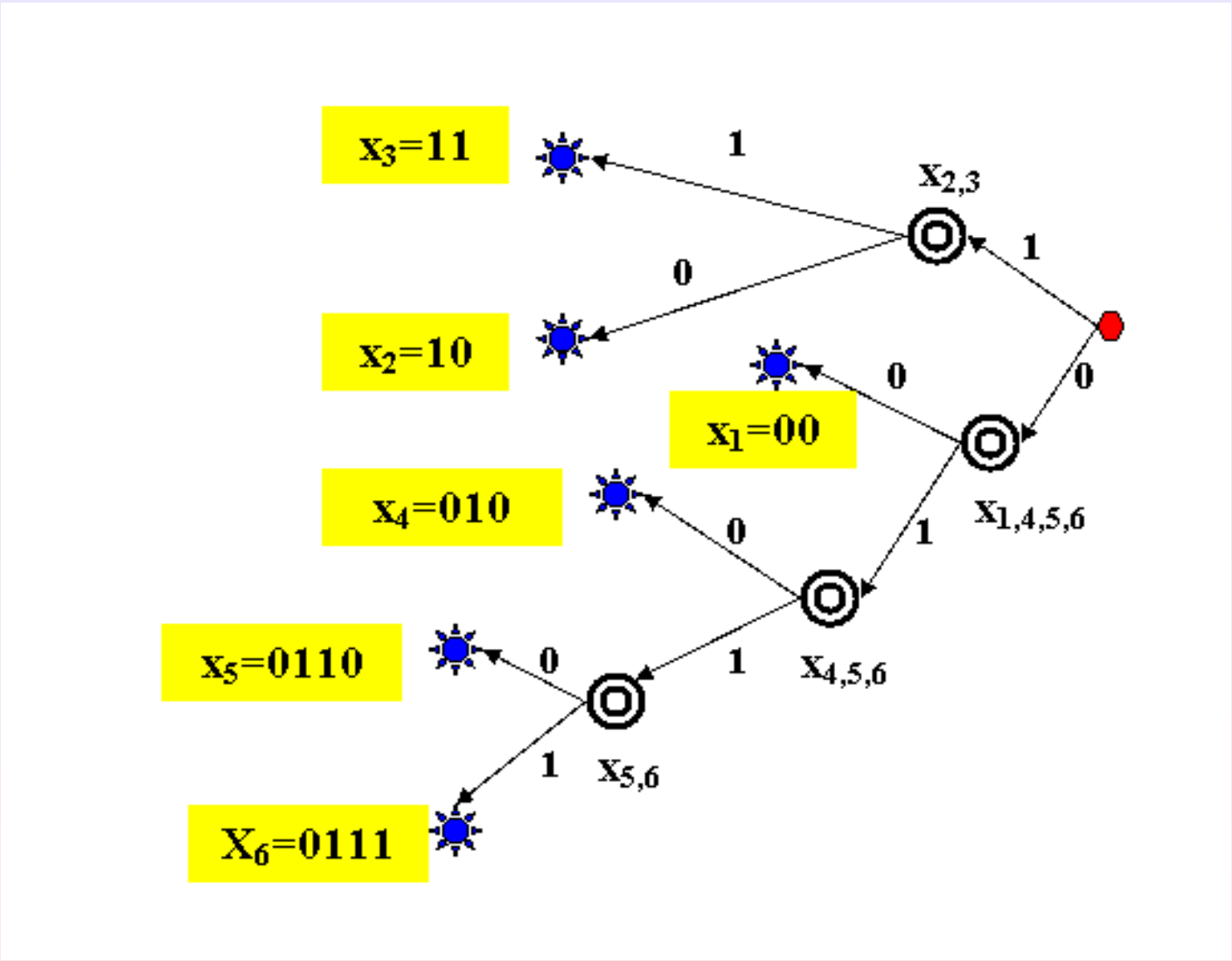
Etape 1		Etape 2		Etape 3		Etape 4			
x_1	0.3	x_1	0.3	x_1	0.3	$x_{3,4,5,6}$	0.45		
x_2	0.25	x_2	0.25	x_2	0.25				
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25				
x_4	0.1	$x_{5,6}$	0.15	x_3	0.2				
x_5	0.1	x_4	0.1						
x_6	0.05								

Exemple

Etape 1		Etape 2		Etape 3		Etape 4			
x_1	0.3	x_1	0.3	x_1	0.3	$x_{3,4,5,6}$	0.45		
x_2	0.25	x_2	0.25	x_2	0.25	x_1	0.3		
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25	x_2	0.25		
x_4	0.1	$x_{5,6}$	0.15	x_3	0.2				
x_5	0.1	x_4	0.1						
x_6	0.05								

Exemple

Etape 1		Etape 2		Etape 3		Etape 4		Etape 5	
x_1	0.3	x_1	0.3	x_1	0.3	$x_{3,4,5,6}$	0.45	$x_{3,4,5,6}$	0.45
x_2	0.25	x_2	0.25	x_2	0.25	x_1	0.3	$x_{1,2}$	0.55
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25	x_2	0.25		
x_4	0.1	$x_{5,6}$	0.15	x_3	0.2				
x_5	0.1	x_4	0.1						
x_6	0.05								


x3=11
1
x2,3
1
0
x2=10
0
0
x1=00
x4=010
0
1
x1,4,5,6
x5=0110
0
1
x4,5,6
1
x5,6
X6=0111

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 longuer 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

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

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 longuer 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

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

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 longuer 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

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

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 longuer 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

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

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 longuer 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

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

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 : 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 $f_i = \dfrac{m_i}{M}$ 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 $f_i = \dfrac{m_i}{M}$ 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.

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 $H(T) \simeq 2.38$.

Construction du code de Huffman.

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			c	2						
h	1	b	1								
s	1										

Construction du code de Huffman.

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.

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.

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.

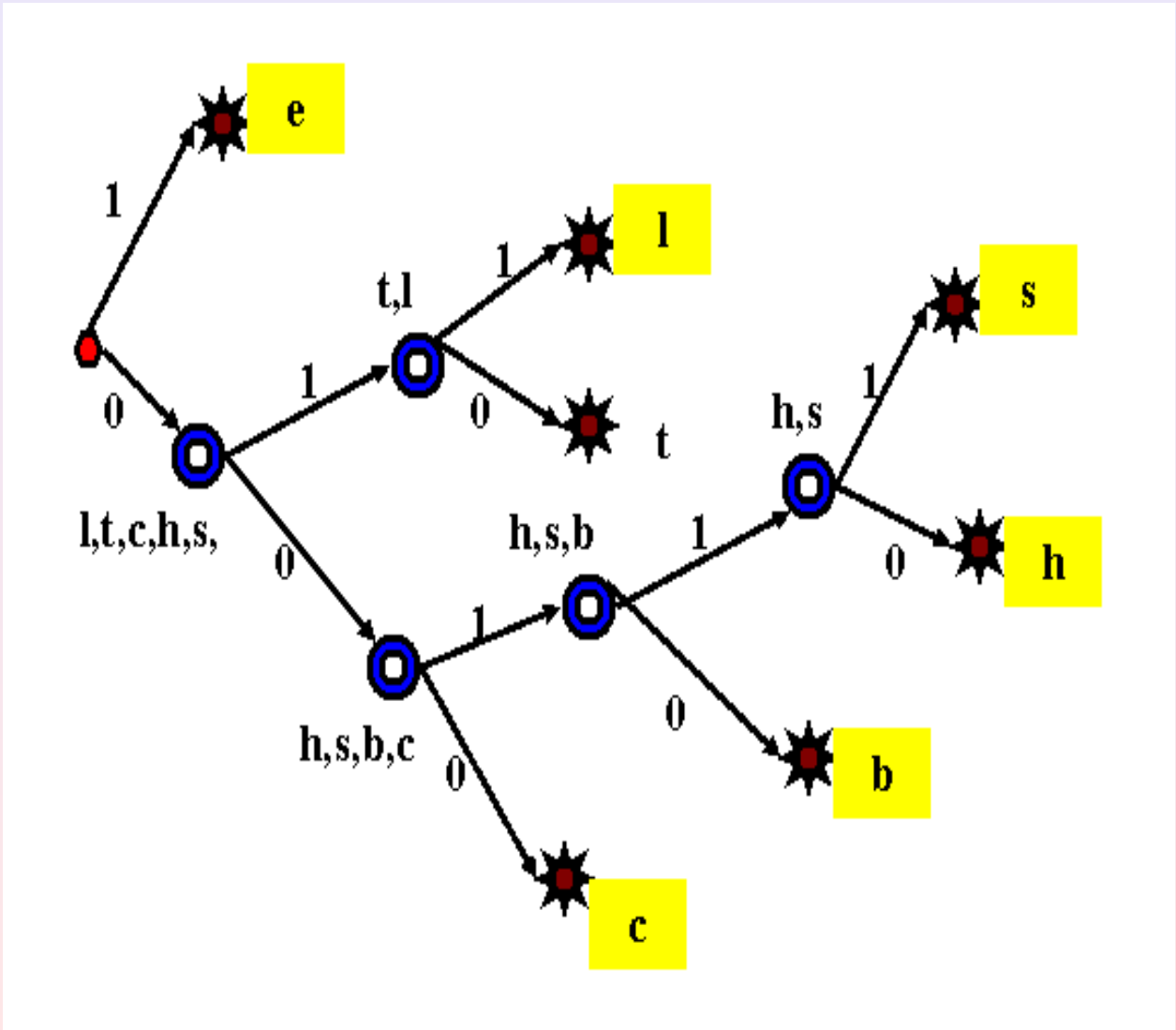
e	8	e	8	e	8	e	8	e	8	hsbclt	12
l	4	l	4	l	4	hsbc	5	lt	7	e	8
t	3	t	3	t	3	l	4	hsbc	5		
c	2	c	2	hsb	3	t	3				
b	1	hs	2	c	2						
h	1	b	1								
s	1										

Construction du code de Huffman.

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 $\bar{L} = 2.45$.

Le message codé sera (sans les espaces)

$$U = 000101001011000001101011011110011101000101011011 1$$

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

$$t = \frac{49}{160} \simeq 0.305$$

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

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

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

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

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

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

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

Codage de Huffman adaptatif : principe

1. L'arbre de codage est mis à jour à chaque lecture de caractère dans le texte T
2. Chaque caractère lu est encodé en utilisant l'arbre précédant
3. Le décodeur construit l'arbre en même temps
4. Il n'est pas nécessaire de communiquer l'arbre de codage
5. utilisable en temps réel

Codage de Huffman adaptatif : principe

1. L'arbre de codage est mis à jour à chaque lecture de caractère dans le texte T
2. Chaque caractère lu est encodé en utilisant l'arbre précédant
3. Le décodeur construit l'arbre en même temps
4. Il n'est pas nécessaire de communiquer l'arbre de codage
5. utilisable en temps réel

Codage de Huffman adaptatif : principe

1. L'arbre de codage est mis à jour à chaque lecture de caractère dans le texte T
2. Chaque caractère lu est encodé en utilisant l'arbre précédant
3. Le décodeur construit l'arbre en même temps
4. Il n'est pas nécessaire de communiquer l'arbre de codage
5. utilisable en temps réel

Codage de Huffman adaptatif : principe

1. L'arbre de codage est mis à jour à chaque lecture de caractère dans le texte T
2. Chaque caractère lu est encodé en utilisant l'arbre précédant
3. Le décodeur construit l'arbre en même temps
4. Il n'est pas nécessaire de communiquer l'arbre de codage
5. utilisable en temps réel

Codage de Huffman adaptatif : principe

1. L'arbre de codage est mis à jour à chaque lecture de caractère dans le texte T
2. Chaque caractère lu est encodé en utilisant l'arbre précédant
3. Le décodeur construit l'arbre en même temps
4. Il n'est pas nécessaire de communiquer l'arbre de codage
5. utilisable en temps réel

Codage de Huffman adaptatif : conventions

1. L'alphabet de la source est supposé connu à priori
2. Au départ, tous les caractères sont considérés équiprobables
3. On initialise l'alphabet avec le nombre d'occurrences 1 pour chaque caractère
4. Les caractères sont ordonnés dans l'ordre lexico-graphique
5. On construit alors l'arbre initial de Huffman

Codage de Huffman adaptatif : conventions

1. L'alphabet de la source est supposé connu à priori
2. Au départ, tous les caractères sont considérés équiprobables
3. On initialise l'alphabet avec le nombre d'occurrences 1 pour chaque caractère
4. Les caractères sont ordonnés dans l'ordre lexico-graphique
5. On construit alors l'arbre initial de Huffman

Codage de Huffman adaptatif : conventions

1. L'alphabet de la source est supposé connu à priori
2. Au départ, tous les caractères sont considérés équiprobables
3. On initialise l'alphabet avec le nombre d'occurrences 1 pour chaque caractère
4. Les caractères sont ordonnés dans l'ordre lexico-graphique
5. On construit alors l'arbre initial de Huffman

Codage de Huffman adaptatif : conventions

1. L'alphabet de la source est supposé connu à priori
2. Au départ, tous les caractères sont considérés équiprobables
3. On initialise l'alphabet avec le nombre d'occurrences 1 pour chaque caractère
4. Les caractères sont ordonnés dans l'ordre lexico-graphique
5. On construit alors l'arbre initial de Huffman

Codage de Huffman adaptatif : conventions

1. L'alphabet de la source est supposé connu à priori
2. Au départ, tous les caractères sont considérés équiprobables
3. On initialise l'alphabet avec le nombre d'occurrences 1 pour chaque caractère
4. Les caractères sont ordonnés dans l'ordre lexico-graphique
5. On construit alors l'arbre initial de Huffman

Codage de Huffman adaptatif : mise à jour de l'arbre de codage

1. Chaque caractère lu est d'abord encodé en utilisant l'arbre de codage en cours
2. Son compteur d'occurrences est incrémenté de 1
3. On reconstruit l'arbre

Codage de Huffman adaptatif : mise à jour de l'arbre de codage

1. Chaque caractère lu est d'abord encodé en utilisant l'arbre de codage en cours
2. Son compteur d'occurrences est incrémenté de 1
3. On reconstruit l'arbre

Codage de Huffman adaptatif : mise à jour de l'arbre de codage

1. Chaque caractère lu est d'abord encodé en utilisant l'arbre de codage en cours
2. Son compteur d'occurrences est incrémenté de 1
3. On reconstruit l'arbre

Codage de Huffman adaptatif : décodage

1. Le décodeur construit l'arbre initial identique
2. Il reçoit les bits jusqu'à reconnaître un mot de code dans l'arbre en cours
3. Dès qu'un symbole est reconnu, il est décodé et 'arbre est mis à jour

Codage de Huffman adaptatif : décodage

1. Le décodeur construit l'arbre initial identique
2. Il reçoit les bits jusqu'à reconnaître un mot de code dans l'arbre en cours
3. Dès qu'un symbole est reconnu, il est décodé et 'arbre est mis à jour

Codage de Huffman adaptatif : décodage

1. Le décodeur construit l'arbre initial identique
2. Il reçoit les bits jusqu'à reconnaître un mot de code dans l'arbre en cours
3. Dès qu'un symbole est reconnu, il est décodé et 'arbre est mis à jour

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

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

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.
- 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

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.
- 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

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.
- 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

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.
- 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

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.
- 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

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.
- 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

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.
- 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

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)

LZ 78 : Exemple

Le code définitif obtenu sera

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

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

LZ 78 : Exemple

Le code définitif obtenu sera

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

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