Cours/TD 3 Codage Huffman

1.1 Codage Huffman

Definition. *Soit une source discrete et sans mémoire produisant N symboles d'un alphabet $a_0, a_1, \cdot, a_{N-1}$ selon la distribution de probabilité $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$.*
Un code binaire préfixe C est optimal $\iff$

La longueur moyenne $\bar{l} = p_0 l_0 + p_1 l_1 + \cdots + p_{N-1} l_{N-1}$ de ses mots est minimale dans l'ensemble de **tous** *les codes binaires* **préfixes** *associés à la source et à la distribution de probabilité considérées.*

Algorithme de Huffman

Initialisation Liste des noeuds candidats : chaque symbole à coder est un nœud pondéré dont le poids est sa probabilité.

Tant que nous avons au moins deux nœuds dans la liste des noeuds candidats

- On identifie deux nœuds de poids le plus faible de la liste actuelle noeuds candidats. Nous les remplaçons dans la liste par un père dont le poids sera la somme des poids de ses successeurs.

L'algorithme de Huffman construit récursivement un arbre binaire pondéré avec la somme des poids égale à 1, appelé arbre avec des probabilités. À chaque étape, on trouve un nœud antécédent (un père) à deux nœuds (les fils) pris d'une liste de candidats. À chaque étape la somme des poids restera égale à 1.

L'algorithme de Huffman produit un code binaire préfixe optimal

Exemple. *Considérons une source discrète sans mémoire qui a produit "aabbbcddef" sur l'alphabet $\{a, b, c, d, e, f\}$.*

Après une analysé des fréquences nous avons $\mathbf{p} = (\ p_0, p_1, \cdots, p_7\ )$ avec $p_a = p_d = \frac{2}{10}$, $p_b = \frac{3}{10}$, $p_c = p_e = p_f = \frac{1}{10}$. On a donc :

Initialisation

b	3/10
a	2/10
d	2/10
c	1/10
e	1/10
f	1/10

Étape 2 On choisit 'e' et 'f' parmi les symboles de plus faible poids. La liste misé à jour est :

b	3/10
a	2/10
d	2/10
c	1/10
(ef)	2/10

Étape 3 On prend 'c' et on choisit '(e,f)' parmi les symboles de plus faible poids. La liste misé à jour est :

b	3/10
a	2/10
d	2/10
$((ef)c)$	3/10

Étape 4 On prend 'a' et 'd' en tant que symboles de plus faible poids. La liste misé à jour est :

b	3/10
(ad)	4/10
$((ef)c)$	3/10

Étape 5 On prend 'b' and '(c (e f))' en tant que symboles de plus faible poids. La liste misé à jour est :

$(b((ef)c))$	6/10
(ad)	4/10

Étape 6

((b ((e f) c)) (a d))

L'arbre binaire construit est dans la figure 1.1.

Notre codage :

$'b'$	3/10	00
$'a'$	2/10	10
$'d'$	2/10	11
$'c'$	1/10	011
$'e'$	1/10	0100
$'f'$	1/10	0101

Exemple. *Considérons une source discrète sans mémoire qui produit des symboles avec la loi de probabilité* $\mathbf{p} = (\ p_0, p_1, \cdots, p_6\ ) = \{0.4, 0.2, 0.15, 0.1, 0.05, 0.05, 0.05\}$ *sur l'alphabet* $\{a, b, c, d, e, f, g\}$.

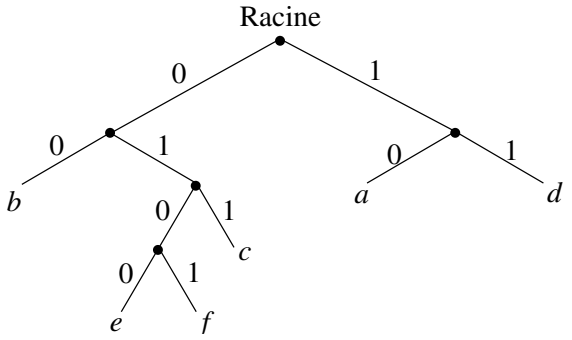

FIG. 1.1 – Arbre binaire

1. *Calculez $H(\mathbf{p})$.*
2. *Trouvez le code de Huffman associé*
3. *Calculez $\bar{l}$, la longueur moyenne des mots code, et comparer-la à $H(\mathbf{p})$.*

Le codage idéal donné par l'entropie sera $H(\mathbf{p}) = p_0 I_0 + p_1 I_1 + \ldots + p_6 I_6 = -p_0 \log_2 p_0 - p_1 \log_2 p_1 - \ldots - p_6 \log_2 p_6 = 2.38$. (Rappel : $\log_2 p = \frac{\ln p}{\ln 2}$)

Pour un codage sans statistiques supposant le cas équiprobable, chacune des sept lettres serait codée en $\log_2 N = log_2 7 = 2.8$ bits.

a	0
b	100
c	101
d	110
e	1110
f	11110
g	11111

On obtient comme longueur moyenne du codage Huffman $\bar{l} = 5(0.05 + 0.05) + 4(0.05) + 3(0.1 + 0.15 + 0.2) + 1(0.4) = 2.45$.

Exercice. *Considérons une source discrète sans mémoire sur l'alphabet a_0, a_1, a_2, a_3, a_4 qui produit selon la distribution de probabilité $p_0 = p_2 = 0.2$, $p_1 = 0.4$, $p_3 = p_4 = 0.1$.*

1. *Calculez l'entropie de la source $H(\mathbf{p})$.*
2. *Trouvez le code de Huffman associé*
3. *Trouvez le code de Fano associé*
4. *Calculez pour les deux codages les longueurs moyennes des mots code $\overline{l_H uffman}$, $\overline{l_S hano}$, et comparez-les à $H(\mathbf{p})$.*

1.2 Approximation de l'entropie par codage en blocs

Limitations du codage de Huffman

- Le codage de Huffman impose d'utiliser un nombre entier de bits pour un symbole source ⇒ codage Huffman sur des blocs de n symboles. On montre alors qu'on peut s'approcher de façon plus fine de l'entropie.
- Le codage de Huffman évalue les probabilités des symboles au debut, donc il n'est pas adapté dans le cas d'une source dont les propriétés statistiques évoluent ⇒ codage Huffman adaptatif

Modèle different : codage en blocs à n symboles

Soit une source sans mémoire, produisant N symboles $a_0, a_1, \ldots, a_{N-1}$ selon la distribution de probabilité $\mathbf{p} = (\ p_0, p_1, \ldots, p_{N-1}\ )$. Considérons les *mots de longueur* n $(n \geq 1)$ comme les unités de production de la source :

$\mathbf{x} = a_{j_1} a_{j_2} \ldots a_{j_n}$ sont les nouveaux symboles produits. Il y a N^n mots de longueur n sur un alphabet à N symboles.

Étant donne que les symboles initiaux sont produits indépendamment, la distribution de probabilité des mots de longueur n est : $\mathbf{p}^{(n)} = (\prod p_{j_1} p_{j_2} \cdots p_{j_n})_{0 \leq j_1, j_2, \ldots j_n \leq N-1}$

Proposition. *Soit une source sans mémoire, produisant N symboles $a_0, a_1, \ldots, a_{N-1}$ selon la distribution de probabilité $\mathbf{p} = (\ p_0, p_1, \ldots, p_{N-1}\ )$. Soit C un code de Huffman associé* ***mots de longueur*** *n, et soit $\bar{l}_i$ la longueur moyenne des mots code par symbole initial. Alors :*

$$H(\mathbf{p}) \leq \bar{l}_i < H(\mathbf{p}) + \frac{1}{n} \tag{1.2.1}$$

Preuve. *Soit $\bar{l}_n = \sum p(\mathbf{x}) l(\mathbf{x})$ la longueur moyenne des mots du code C. Alors on a : $H\left(\mathbf{p}^{(n)}\right) \leq \bar{l}_n < H\left(\mathbf{p}^{(n)}\right) + 1$. Si $\bar{l}_n$ est le nombre de bits nécessaires pour coder des mots de longueur n, alors le nombre de bits nécessaires pour coder un symbole initial est $\bar{l}_i = \frac{\bar{l}_n}{n}$.*

Si nous avons aussi $H\left(\mathbf{p}^{(n)}\right) = n \cdot H(\mathbf{p})$ *la preuve est finie. Prouvons le !*

$$
\begin{aligned}
H\left(\mathbf{p}^{(n)}\right) = & \qquad (1.2.2)\\
&= -\sum_{j_1=1}^{N}\sum_{j_2=1}^{N}\cdots\sum_{j_n=1}^{N} p(a_{j_1}a_{j_2}\ldots a_{j_n})\log_2 p(a_{j_1}a_{j_2}\ldots a_{j_n})\\
&= -\sum_{j_1=1}^{N}\sum_{j_2=1}^{N}\cdots\sum_{j_n=1}^{N} p_{j_1}p_{j_2}\ldots p_{j_n}\log_2 p_{j_1}p_{j_2}\ldots p_{j_n}\\
&= -\sum_{j_1=1}^{N}\sum_{j_2=1}^{N}\cdots\sum_{j_n=1}^{N} p_{j_1}p_{j_2}\ldots p_{j_n}\sum_{k=1}^{N}\log_2 p_{j_k}\\
&= -\sum_{j_1=1}^{N} p_{j_1}\log_2 p_{j_1}\left\{\sum_{j_2=1}^{N}\sum_{j_3=1}^{N}\cdots\sum_{j_n=1}^{N} p_{j_2}p_{j_3}\ldots p_{j_n}\right\}\\
&- \sum_{j_2=1}^{N} p_{j_2}\log_2 p_{j_2}\left\{\sum_{j_1=1}^{N}\sum_{j_3=1}^{N}\cdots\sum_{j_n=1}^{N} p_{j_1}p_{j_3}\ldots p_{j_n}\right\}\\
&- \sum_{j_n=1}^{N} p_{j_n}\log_2 p_{j_n}\left\{\sum_{j_1=1}^{N}\sum_{j_2=1}^{N}\cdots\sum_{j_{n-1}=1}^{N} p_{j_1}p_{j_2}\ldots p_{j_{n-1}}\right\}
\end{aligned}
$$

Les $n-1$ *sommes qui se trouvent entre les parentheses sont égales à* 1*. Donc,*

$$
\begin{aligned}
H(\mathbf{p}^{(n)}) = & \qquad (1.2.3)\\
&= -\sum_{j_1=1}^{N} p_{j_1}\log_2 p_{j_1} - \cdots - \sum_{j_n=1}^{N} p_{j_n}\log_2 p_{j_n}\\
&= n \cdot H(\mathbf{p})
\end{aligned}
$$

Exercice. *Considérons une source binaire qui produit un bit par unité de temps (par exemple, chaque µsec) selon la distribution de probabilité* $p_0 = \frac{3}{4}$*,* $p_1 = \frac{1}{4}$*. Supposons que notre flot binaire doit passer par un canal qui n'accepte que* 0.82 *bits par unité de temps.*
Construisez un adaptateur par codage de Huffman en blocs.

Théoriquement est-il possible de compresser à ce point le flot binaire ? Oui, parce que l'entropie de la source est : $H(\mathbf{p}) = \frac{3}{4}\log_2\frac{4}{3} + \frac{1}{4}\log_2 4 = 0.81$ Avant codage la longueur moyenne est $\bar{l} = 1$

Premier essai Codage avec 2-blocs – figure 1.2.

Nous avons :

$$p(00) = p(0)p(0) = \frac{3}{4}\frac{3}{4} = \frac{9}{16}$$

$$p(01) = p(10) = \frac{3}{16}, p(11) = \frac{1}{16}$$

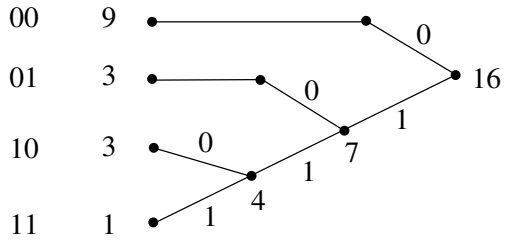

FIG. 1.2 – Codage avec 2-blocs

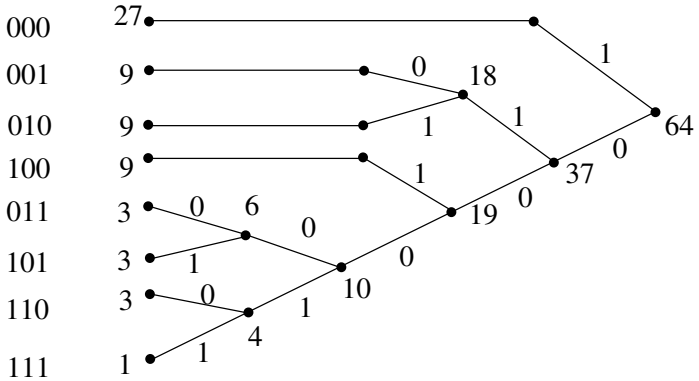

FIG. 1.3 – Codage avec 3-blocs

Le codage est

00	0
01	10
10	110
11	111

et

$$\bar{l}_2 = \frac{1}{16}(9 \cdot 1 + 3 \cdot 2 + 3 \cdot 3 + 1 \cdot 3) = \frac{27}{16}$$

$$\bar{l}_i = \frac{\bar{l}_2}{2} = \frac{27}{32} = 0.84$$

Donc le codage en blocs à deux symboles est un échec car $\bar{l}_i > 0.82$. ***Deuxième essai*** Codage avec 3-blocs – figure 1.3.

Nous avons :

$$p(000) = \frac{27}{64}, p(010) = p(100) = p(001) = \frac{9}{64}$$

$$p(111) = \frac{1}{64}, p(110) = p(101) = p(011) = \frac{3}{64}$$

Le codage est

000	1
001	010
010	011
100	001
011	00000
101	00001
110	00010
111	00011

Donc

$$\bar{l}_3 = \frac{1}{64}(1 \cdot 27 + 3 \cdot 3 \cdot 9 + 5 \cdot 3 \cdot 3 + 5 \cdot 1) = \frac{158}{64}$$

$$\bar{l}_i = \frac{\bar{l}_3}{3} = \frac{158}{192} = 0.823$$

Donc le codage en blocs à trois symboles est un échec car $\bar{l}_i > 0.82$.

Troisième essai Codage avec 4-blocs :

Nous avons :

$$p(0000) = \frac{81}{256}, p(1000) = p(0100) = p(0010) = p(0001) = \frac{27}{256}$$

$$p(1111) = \frac{1}{256}, p(1110) = p(1101) = p(1011) = p(0111) = \frac{3}{256}$$

$$p(1100) = p(1010) = p(1001) = p(0011) = p(0110) = p(0101) = \frac{9}{256}$$

voir figure 1.4.

Le codage est

0000	00
0001	100
0010	101
0100	110
1000	111
0011	01000
0101	01001
0110	01010
1001	01011
1010	01100
1100	01101
0111	011110
1011	011111
1101	011101
1110	0111000
1111	0111001

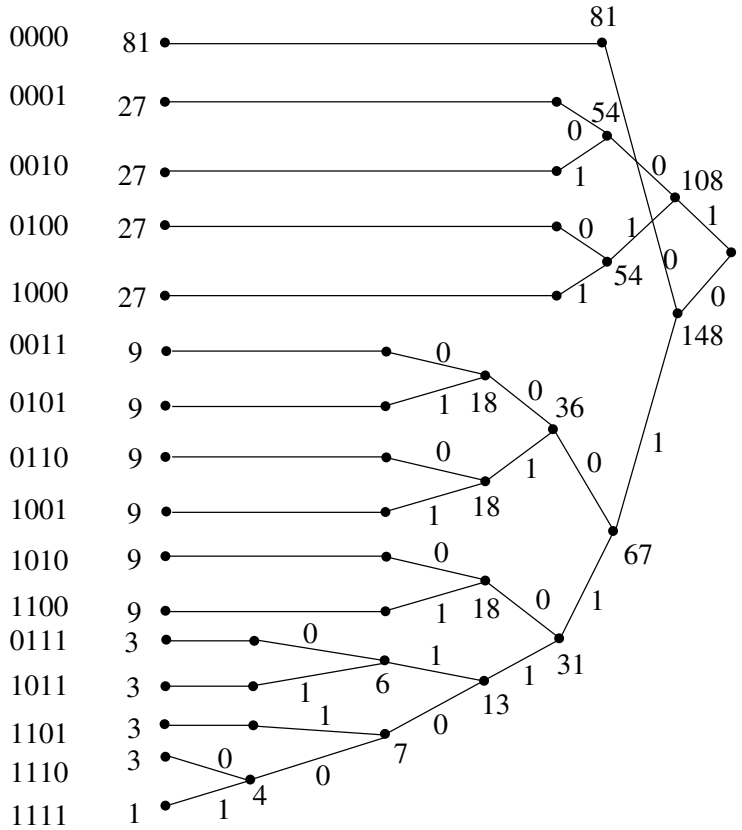

FIG. 1.4 – Codage avec 4-blocs

Donc

$$\bar{l}_4 = \frac{1}{256}(2 \cdot 81 + 3 \cdot 108 + 5 \cdot 54 + 6 \cdot 9 + 7 \cdot 4) = \frac{838}{256}$$

$$\bar{l}_i = \frac{\bar{l}_4}{4} = \frac{838}{1024} = 0.818$$

Donc le codage en blocs à quatre symboles est bon.

Exercice. *Considérons une source* $\{a_0, a_1, a_2\}$ *qui produit un bit par unité de temps (par exemple, chaque* μ*sec) selon la distribution de probabilité* $p_0 = 0.8$, $p_1 = 0.02$, $p_3 = 0.18$. *Supposons que notre flot de donées doit passer par un canal qui n'accepte que* 0.87 *bits par unité de temps.*

1. *Calculez l'entropie de la source* $H(\mathbf{p})$ *pour decider s'il est possible de compresser à ce point le flot*
2. *Construisez un adaptateur par codage de Huffman en blocs.*

Appendix Arbre d'un code. Optimalité du codage Huffman

Definition. *Un arbre fini ponderé avec une racine qui satisfait*

- *la racine a la probabilité* 1
- *la probabilité de chaque noeud (racine aussi) est la somme des probabilités de ses noeuds fils*

s'appelle un arbre avec des probabilités.

On notera que un arbre avec des probabilités n'est pas nécessairement D-aire (i.e. chaque noeud a au plus D fils).

Rappel(TD2) : Un codage binaire peut être représenté de manière graphique par un arbre binaire dans lequel un déplacement à gauche (droite) correspond à un "0" (respectivement à un "1"). Les mots code associes aux symboles sont les chemins de la racine vers les feuilles. Dans cette correspondance *un code préfixe est un code dont les symboles codés sont des feuilles.* .

L'**arbre d'un code binaire préfixe** peut être muni naturellement de **une structure de probabilités :** en associant aux feuilles qui correspondent aux mots code, les probabilités des symboles à coder. Par convention, aux feuilles qui ne correspondent pas à un mot code on associe une probabilité zero. Aux noeuds non-terminaux on associe la probabilité cumulé de leurs fils. La racine aura la probabilité 1.

Lemme 1. *Soit une source discrete et sans mémoire produisant N symboles d'un alphabet selon la distribution de probabilité $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$. Dans l'arbre binaire d'un code préfixe nous changeons un noeud intermédiaire en feuille en combinant toutes feuilles descendant de lui dans un mot composé d'un alphabet réduit. Si l'arbre original était optimal pour l'alphabet original, l'arbre réduit est optimal pour l'alphabet réduit.*

Preuve. *dans le cas contraire on trouve un codage avec une longueur moyenne plus courte pour l'alphabet réduit. Nous développons le sous arbre du mot composé et nous obtenons un codage plus court pour l'alphabet initial.*

Lemme 2. *Soit une source discrete et sans mémoire produisant N symboles d'un alphabet selon la distribution de probabilité $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$. L'arbre binaire d'un code préfixe optimal code utilise toutes les feuilles de l'arbre.*

Preuve. *: Supposons qu'il existe dans l'arbre du code au moins une feuille non-utilisé, f_i.*

Parce que le code est optimal ces feuilles doivent être au niveau de la profondeur de l'arbre. Dans le cas contraire il existe une branche de l'arbre qui est strictement plus profonde $l > l_{f_i}$. Parmi les feuilles de cette branche il y a au moins un mot code, soit $f = c_k$(l'arbre est associe a un code préfixe).

En associant au symbol a_k la feuille (et son chemin comme mot code) f_i à la place de la feuille f nous obtenons un code préfixe qui a une longueur moyenne strictement plus petite : $p_k \cdot l > p_k \cdot l_{f_i}$. Ce qui contredit notre hypothèse.

Donc, f_i se trouve a une profondeur maximale dans l'arbre. Parce que cet arbre correspond à un code, dans cette branche il existe un mot code. Donc, il existe au moins une valeur c_j , qui diffère de cette feuille seulement par son dernier chiffre binaire (digit). Si nous effaçons ce chiffre binaire nous retrouvons l'écriture binaire d'un noeud non-terminal. Le code étant préfixe ne contenait pas ce mot code. Le nouveau code préfixe a donc une longueur moyenne plus petite. Donc le premier code n'était pas optimal.

Proposition. *Il y a un code binaire préfixe optimal pour lequel les mots code associés au deux symboles de poids le plus faible, disons a_k et a_r sont des feuilles qui different seulement par le dernier chiffre binaire (digit). Si nous changeons le noeud père de a_k et a_r en feuille correspondant à un mot composé d'un alphabet réduit, l'arbre réduit est optimal pour l'alphabet réduit. Les longueurs moyennes optimales des codages sont liées par :*

$$\bar{l}_{\min} \;=\; \overline{L}_{\min} \;+\; p_k \;+\; p_r. \tag{1.2.4}$$

Preuve. *Soit C un codage optimal. Soit c_j un des plus longues mots code dans C. Donc $l_j \geq l_k$. Parce que toutes les feuilles sont bien utilisés, c_j a un pere commun avec un autre mot code c_i. Donc* $\begin{matrix} 0c_i \\ 1c_j \end{matrix}$ *ou* $\begin{matrix} 1c_i \\ 0c_j \end{matrix}$

Si $j \neq k$ nous échangeons les feuilles pour les mots code c_j et c_i. Cette manœuvre n'augmente pas la longueur moyenne du codage C. Parce que $p_j \geq p_k$ $l_j \geq l_k$ alors

$(p_j - p_k)(l_j - l_k) \geq 0$. Donc $p_j l_j + p_k l_k \;\geq\; p_j l_k + p_k l_j$. Donc le nouveau code préfixe est optimal aussi. Si $i \neq r$ nous échangeons les feuilles pour les mots code c_r et c_i et le nouveau code préfixe est optimal aussi. Mais dans ce code, les mots code associés aux deux symboles de poids le plus faible diffèrent seulement par le dernier chiffre binaire (digit). En utilisant le Lemme 1 nous avons que l'arbre réduit est optimal pour l'alphabet réduit avec un symbole agrégé $a_k + a_r$ à la place de a_k et a_r.

Soit les longueurs des mots code (L pour le codage réduit C' , l pour C) :

$$l_j = \begin{cases} L_{(k,r)} + 1 & si \quad j = k,r \\ L_j & sinon \end{cases}$$

On obtient donc pour les longueurs moyennes : $\bar{l} \;=\; \sum_{j \neq k,r} \; p_j l_j \;+\; p_k l_k \;+\; p_r l_r \;=\; \sum_{j \neq k,r} \; p_j L_j \;+\; p_{(k,r)} L_{(k,r)} \;+\; p_k \;+\; p_r \;=\; \overline{L} \;+\; p_k \;+\; p_r$.

Mais les deux codages sont optimales donc $\bar{l}_{\min} \;=\; \overline{L}_{\min} \;+\; p_k \;+\; p_r$.

On notera que dans un code binaire préfixe optimal si $p_j \;>\; p_k$ alors $l_j \;\leq\; l_k$.

Corollaire. *Considérons une source* $\mathbf{S}$ *produisant* N *symboles selon la distribution de probabilité* $\mathbf{p} = (\ p_0, p_1, \ldots, p_{N-1}\ )$. *Remplaçons les deux symboles* a_k *et* a_r *de probabilités minimales par un seul symbole* $a_{(k,r)}$ *de probabilité* $p_{(k,r)} = p_k + p_r$, *et soit* $\mathbf{S}$*' la source à* $N-1$ *états ainsi obtenue.*
Soit C' *un code binaire préfixe optimal pour* $\mathbf{S}$*', et soit* $\mathbf{x}$ *le mot code pour* $a_{(k,r)}$. *Soit* C *le code binaire préfixe pour* $\mathbf{S}$ *qui s'en déduit :*

$$\begin{array}{lcl} a_k & \longmapsto & \mathbf{x}0 \\ a_r & \longmapsto & \mathbf{x}1 \end{array} .$$

Alors, C *est optimal pour* $\mathbf{S}$.

Preuve. *Si* C' *est un code binaire préfixe optimal pour* $\mathbb{S}$*' sa longueur est minimale* $\overline{L}' = \overline{L}_{\min}$. *Si on remplace* $a_{(k,r)}$ *avec les deux symboles* a_k, a_r, $\overline{l}' = \overline{L}' + p_k + p_r = \overline{L}_{\min} + p_k + p_r$ *qui est égal à* $\overline{l}_{\min}$. *Donc* C *est optimal.*

L'algorithme de Huffman produit des codes binaires préfixes optimaux. En suivant la proposition nous avons un algorithme pour construire l'arbre binaire associe à un code binaire préfixe optimal.

Remarque. *Un algorithme similaire peut-être construit dans le cas* D*-aire.*