EISTI. **Année scolaire 2008-2009**

DÉPARTEMENT " INFORMATIQUE "

THÉORIE DE L'INFORMATION

Série d'exercices N^{o}4

PARTIE I. CODAGE DE HUFFMAN

Nous allons considérer une source d'alphabet

$$\Omega = \{a, b, c, d, e, f, g, h, i, j\}$$

et un canal d'alphabet binaire $\{0, 1\}$. Voici les définitions importantes.

Théorème de la borne inférieure de longueur de code

Théorème 0.1. *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 binaire $\Omega_C = \{0, 1\}$ 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 :

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

L'égalité n'est possible que si $\forall i = 1, \ldots, n, \;\; p_i = 2^{-l_i}$.

Code absolument optimal C'est un code dont la longueur moyenne de mots est égale à la borne inférieure, $H(S)$.

Code optimal. Un code est dit optimal dans une certaine classe de codes si sa longueur moyenne de mots est minimale dans cette classe. La classe de codes la plus importante est celle de codes sans préfixe.

Exercice 1. Soit une source d'alphabet Ω et de distribution de probabilité suivante

s_i	a	b	c	d	e	f	g	h	i	j
m_i	0.0625	0.125	0.0625	0.125	0.0625	0.125	0.0625	0.0625	0.25	0.0625

1. Calculer l'entropie de la source.
2. Montrer qu'il existe un code binaire **absolument optimal** pour cette source.
3. Quelles sont les longueurs des mots d'un tel code pour chaque caractère ?
4. Construire un code sans préfixe de longueurs correspondantes en utilisant un arbre binaire.

Exercice 2. Soit une source d'alphabet Ω et de distribution de probabilité suivante

s_i	a	b	c	d	e	f	g	h	i	j
m_i	0.2	0.05	0.1	0.05	0.15	0.05	0.1	0.05	0.15	0.1

1. Quelle est la longueur moyenne minimale pour un code binaire de cette source ?
2. Existe-t-il un code absolument optimal ?
3. Construire un code sans préfixe optimal selon la méthode de Huffman.
4. Représenter ce code sous forme d'arbre.

PARTIE II. PRÉPARATION ALGORITHMIQUE.

Cette partie a pour objectif de préparer la réalisation en java de l'algorithme de construction d'un code de Huffman.

Exercice 3. On reprend la même source que dans l'exercice 2. En appliquant la méthode de Huffman dans l'exercice précédant nous avons effectué deux parcours de l'arbre de code : d'abord des feuilles vers la racine et ensuite de la racine vers les feuilles.

Il est possible d'obtenir l'arbre de code directement lors du premier parcours. Pour cela nous allons rappeler la définition réccursive d'un arbre binaire : *un arbre binaire est soit vide, soit un triplet* (R, G, D) *où* R *est la racine et* G *est un arbre binaire, appelé sous-arbre gauche, et* D *est un arbre binaire, appelé sous-arbre droit.*

Nous allons en plus associer à la racine d'un arbre binaire un mot d'alphabet Ω et un poids (une probabilité). Il est alors possible d'effectuer l'algorithme de Huffman de façon équivalente en manipulant un ensemble d'arbres binaires au lieu d'un alphabet.

Pour initialiser l'algorithme, on associe à chaque symbole de l'alphabet Ω un arbre-feuille (composé d'une seule racine) de poids égal à la probabilité du symbole. Ensuite, à chaque "fusion" de symboles on crée un nouvel arbre de poids égal à la somme des poids de ses sous-arbres. La figure ci-dessous illustre ce principe.

Refaire l'algorithme de Huffman avec un ensemble d'arbres pour la source de l'exercice 2.

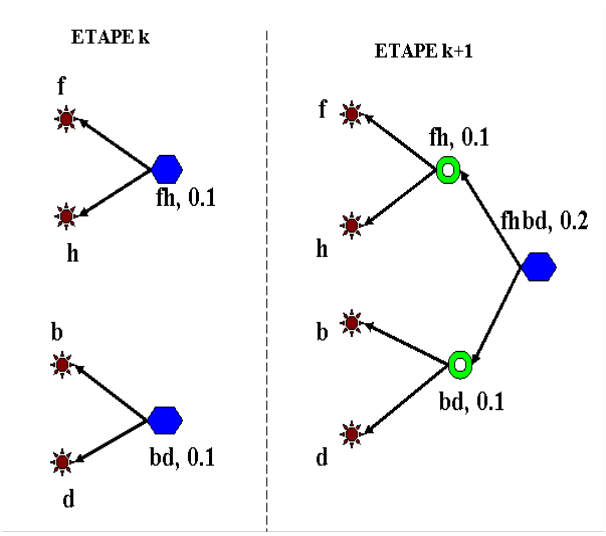

FIGURE 1 – Un exemple de fusion d'arbres