TD9 : Codes

1. Introduction aux codes correcteurs

Les codes correcteurs sont destinés à corriger les erreurs lors de la transmission de données sur des canaux non fiables. Ils sont utilisés non seulement pour les communications « classiques » (wifi, telecom, fibre optique…), mais aussi pour le stockage (CD, RAM…). Un premier exemple est le décodage à distance minimale. On définit la distance de Hamming entre deux mots x et y comme le nombre de bits différents entre x et y. Une suite de bits est décodée en choisissant le caractère dont le code est le plus proche pour la distance de Hamming. Dans la suite, on se donne $a = 0100$, $b = 0011$, $c = 1000$, $d = 1111$.

1.1. Décoder 0111 0110 0011. Que remarque-t-on ?

1.2. Quel est le rendement de ce code ? Quel argument peut-on présenter contre ce genre de codes ?

Une autre façon de détecter des erreurs est de rajouter des bits de contrôle. On présente ici un code de Hamming. On note $\oplus$ l'addition dans $\frac{\mathbb{Z}}{2\mathbb{Z}}$. Si le bloc à coder est $b_1b_2b_3b_4$, on définit $c_1 = b_1 \oplus b_2 \oplus b_3$, $c_2 = b_1 \oplus b_2 \oplus b_4$, $c_3 = b_1 \oplus b_3 \oplus b_4$ et on envoie $b_1b_2b_3b_4c_1c_2c_3$.

1.3. Coder 1101.

1.4. On suppose qu'il y a au plus une erreur (substitution) lors de la transmission. Décoder 1101101 et 0111110.

1.5. Avec la même hypothèse, montrer que l'on peut toujours recomposer le message d'entrée. Indication : soient x et y deux messages à coder, regarder la distance de Hamming minimale entre les codages de x et de y.

1.6. Expliquer l'appellation « code de Hamming $(7, 4)$ » et comparer avec le premier code (avantages, inconvénients, problèmes non résolus).

2. Code de Huffman adaptatif

On supposera connus ici le codage de Huffman et les arbres correspondants. La méthode usuelle de compression de Huffman présente deux inconvénients : il faut lire le texte entièrement avant de lancer la compression ; il faut aussi transmettre le code.
On souhaite utiliser une caractérisation des arbres de Huffman pour définir un algorithme adaptatif qui corrige ces défauts. Nous identifions un code préfixe d'une

source discrète X à un arbre binaire à $|X|$ feuilles étiquetées par les lettres de X. Le code est dit irréductible, si chaque nœud possède 0 ou 2 fils.

Définition : Soit un code préfixe d'une source discrète pour un texte t. Le poids d'une feuille est la probabilité de sa lettre ; le poids d'un nœud est la somme des poids de ses fils.
Définition : un ordre de Gallager $u_1, \dots u_{2k-1}$ sur les nœuds d'un code préfixe irréductible d'une source de cardinal k est un ordre total qui vérifie les conditions suivantes. (i) Les poids des u_i sont décroissants ; (ii) $\forall i < k, u_{2i}$ et u_{2i+1} sont frères.

2.1. Soit T un code préfixe d'une source X. Montrer que T est un arbre de Huffman de X si et seulement s'il existe un ordre de Gallager sur les sommets de T.

En utilisant cette propriété, on peut donc développer un nouvel algorithme (dû à Knuth). On dispose d'une table donnant le codage actuel d'un caractère (par exemple codage ASCII pour une lettre). L'arbre initial est constitué d'une seule feuille, celle de la lettre vide. À chaque fois qu'un caractère a est lu dans le texte source : si a n'a pas été lu précédemment, on imprime le codage de la lettre vide puis le codage initial de a, puis on remplace la feuille contenant la lettre vide par un nœud de poids 1 ayant pour fils la feuille contenant la lettre vide de poids 0 et la feuille a de poids 1 ; sinon :

1. On imprime le codage de a.
2. Si a est frère de ε, alors échanger a avec feuille de même poids d'indice le plus faible dans l'ordre de Gallager ;
3. On augmente le poids de la feuille de a.

Puis on met à jour l'arbre de la feuille vers la racine tant que la propriété de Gallager n'est pas respectée en mettant à jour les nœuds. Avant d'augmenter le poids d'un nœud, on l'échange avec le nœud de même poids et d'indice minimal pour l'ordre de Gallager. Évidemment, le code d'une lettre change en cours de transmission : quand la fréquence (relative) d'une lettre augmente, la longueur de son code diminue. Le destinataire du message compressé mime le codage de l'expéditeur : il maintient l'arbre de Huffman qu'il met à jour comme l'expéditeur, et il sait donc tout moment quel est le code d'une lettre.

2.2. Coder le mot abracra en version adaptative puis comparer avec le codage « classique ». Indication de codes initiaux : a 00000, b 00001, c 00010, r 10001.

2.3. Écrire un algorithme de décodage, puis décoder

$$0000010100010000011101101100101 01$$

Indication de codes initiaux : a 00000, d 00011, k 01010, r 10001, v 10101.