Projet n°3 (codage arithmétique)

Le codage de Huffman, bien qu'optimal, présente l'inconvénient de perdre jusqu'à un bit d'information par symbole codé. Même si l'on peut théoriquement remédier à ce défaut en considérant des sources étendues (c'est-à-dire des blocs de plusieurs symboles), en pratique cette solution n'est guère utilisable car le calcul des fréquences empiriques pour les mots de grande taille devient rapidement infaisable. Beaucoup d'algorithmes de compression actuels utilisent un codage beaucoup plus efficace, appelé codage arithmétique. Le but de ce projet est de mettre en œuvre cette technique de codage.

1. Soit $\mathcal{X} = \{1, 2, \dots n\}$ l'alphabet de la source à coder, muni de la distribution $p(i) = P(X = i)$. On note
$$F(i) = \sum_{j=1}^{i-1} p(j).$$
L'idée du codage arithmétique est de coder un message $x_1 x_2 \dots x_N$ par un nombre réel de l'intervalle $[0, 1[$ dont le développement binaire est fini. Plus précisément, on pose
$$a_0 = 0,\ \delta_0 = 1,$$
puis récursivement
$$a_k = a_{k-1} + \delta_{k-1} \cdot F(x_k),\ \delta_k = \delta_{k-1} \cdot p(x_k), \quad k = 1, 2, \dots N.$$
On code alors le message $x_1 x_2 \dots x_N$ par le nombre binaire le plus court qui appartient à l'intervalle $[a_N, a_N + \delta_N[$, Ce nombre binaire est noté $0, b_1 b_2 b_3 \dots b_L$. L est donc le nombre de bits nécessaires pour coder le message $x_1 x_2 \dots x_N$. Le nombre N est aussi joint au code. Montrer que ce code est régulier sur $\mathcal{X}^*$ (on pourra remarquer que les intervalles $[a_k, a_k + \delta_k[$ sont emboités).

2. On définit la distribution empirique du message $m = x_1 x_2 \dots x_N$ par
$$f(i) = \frac{1}{N} \#\{k \in \{1, \dots N\},\ x_k = i\},$$
et son entropie empirique par
$$H(m) = \sum_{i=1}^{n} -f(i) \log_2 f(i).$$

Montrer que $L \leqslant \lceil 1 - \log_2 \delta_N \rceil$. En déduire que si l'on code m à l'aide de sa distribution empirique (i.e. en prenant $p(i) = f(i)$ pour tout i), alors

$$L < NH(m) + 2.$$

3. En pratique, on doit se limiter à des nombres réels dont la représentation binaire est finie (et même pas trop longue). On suppose donc que les $p(i)$ (et donc les $F(i)$) sont des multiples de 2^{-p}, où p est un entier fixé à l'avance. D'autre part, on simplifie l'écriture des δ_k pour ne conserver que des réels à au plus r bits significatifs, i.e.

$$\delta_k \simeq \tilde{\delta}_k \in S_r = \{2^{-l}k,\ 0 \leqslant k \leqslant 2^r,\ l \geqslant r\}.$$

On note $\lfloor \delta \rfloor_r$ le plus grand réel de S_r inférieur ou égal à δ. En pratique on veillera à ce que l'entier 2^{r+p} soit représentable exactement dans le langage de programmation utilisé. L'algorithme effectif est alors

> $a \leftarrow 0$ et $\tilde{\delta} \leftarrow 1$
> pour $k = 1, 2, \ldots N$
> $a \leftarrow a + \tilde{\delta} \cdot F(x_k)$
> $\tilde{\delta} \leftarrow \lfloor \tilde{\delta} \cdot p(x_k) \rfloor_{2p}$
> fin pour
> trouver le premier entier q pour lequel les q-ièmes bits de a et $a + \tilde{\delta}$ diffèrent
> retourner les q premiers bits de $a + \tilde{\delta}$.

Comme $0 < \tilde{\delta}_k < \delta_k$, on conserve la propriété de code régulier. Implémenter cet algorithme pour $\mathcal{X} = \{0, 1, \ldots 255\}$, p et r quelconques. On pourra représenter a comme un tableau de chiffres binaires, δ par deux entiers k et l tels que $\delta = k \cdot 2^{-l}$ (avec $l \leqslant r$ et $0 \leqslant k < 2^r$), et utiliser la fonction bin_add donnée en complément sur le site web du cours (ou s'inspirer de cette fonction si Scilab n'est pas choisi comme langage de programmation).

4. Tester cet algorithme avec $p = 12$ (ou plus, ou moins, selon la rapidité de votre programme) sur les 2^p premiers caractères du fichier goriot.txt (avec un tel choix, les fréquences empiriques seront naturellement multiples de 2^{-p}). Comparer, en fonction de r, la longueur du code obtenu à l'entropie empirique du fichier. Commenter la différence.

5. Pour décoder le code (N, c) (où $c = 0, b_1 b_2 b_3 \ldots b_L$), on peut procéder comme suit:

> $a_0 \leftarrow 0$ et $\tilde{\delta}_0 \leftarrow 1$
> pour $k = 1, 2, \ldots N$
> trouver l'unique $i \in \{1, 2, \ldots, n\}$ tel que
> $a_{k-1} + \tilde{\delta}_{k-1} \cdot F(i) \leqslant c < a_{k-1} + \tilde{\delta}_{k-1} \cdot F(i+1)$.
> $x_k \leftarrow i$
> $a_k \leftarrow a_{k-1} + \tilde{\delta}_{k-1} \cdot F(i)$
> $\tilde{\delta}_k \leftarrow \lfloor \tilde{\delta}_{k-1} \cdot p(i) \rfloor_{2p}$
> fin pour

Implémenter cet algorithme, et le tester (en combinant codage et décodage) sur un texte de faible taille, par exemple les 2^8 premiers caractères du fichier goriot.txt. On pourra utiliser la fonction bin_leq donnée en complément sur le site web du cours.

6. Quel est l'inconvénient de la méthode de décodage décrite précédemment ? Proposer une manière d'accélérer le décodage en éliminant les bits de c qui deviennent inutiles au fur et à mesure du décodage. Implémenter ce décodage accéléré et le valider sur un fichier de grand taille.