EPFL - Information, Calcul et Communication - Partie théorique

Série 11: Solutions

1 Un peu de magie noire

a) Calcul de l'entropie de la séquence (que l'on va appeler $\mathcal{X}$ comme dans le cours): le A a une probabilité d'apparition de $\frac{5}{12}$, le V et le D de $\frac{2}{12}$ et les 3 lettres restantes de $\frac{1}{12}$. Donc

$$\begin{aligned} H(\mathcal{X}) &= \frac{5}{12}\log_2\left(\frac{12}{5}\right) + 2\,\frac{2}{12}\log_2\left(\frac{12}{2}\right) + 3\,\frac{1}{12}\log_2(12) \\ &= \log_2(12) - \frac{5}{12}\log_2(5) - \frac{1}{3}\log_2(2) = \frac{5}{3} + \log_2(3) - \frac{5}{12}\log_2(5) \end{aligned}$$

ce qui donne numériquement: $H(\mathcal{X}) \simeq 2.28$.

b) Avec l'algorithme de Shannon-Fano, on peut trouver les dictionnaires suivants:

lettre	nb app.	1) nb Q	mot de code	2) nb Q	mot de code	3) nb Q	mot de code
A	5	2	11	2	11	1	1
V	2	2	10	2	10	3	011
D	2	3	011	2	01	3	010
K	1	3	010	3	001	3	001
E	1	3	001	4	0001	4	0001
R	1	3	000	4	0000	4	0000

Avec les dictionnaires 1) et 2), la séquence codée contient 29 bits; avec le dictionnaire 3), la séquence codée contient 28 bits.

c) Avec l'algorithme de Huffman, on trouve le code suivant:

lettre	nb apparitions	mot de code
A	5	1
V	2	011
D	2	010
K	1	001
E	1	0001
R	1	0000

et la séquence codée contient 28 bits. On peut aussi trouver des dictionnaires différents suivant l'arbre que l'on construit, mais la longueur de la séquence codée reste invariablement de 28 bits dans ce cas.

d) Avec l'algorithme de Shannon-Fano, la longueur moyenne du code $L(\mathcal{C}_{SF}) \simeq 2.42$ ou 2.33. Avec l'algorithme de Huffman, la longueur moyenne du code $L(\mathcal{C}_H) \simeq 2.33$, et l'entropie de la séquence vaut $H \simeq 2.28$. On vérifie donc bien les inégalités du cours:

$$H(\mathcal{X}) \leq L(\mathcal{C}_H) \leq L(\mathcal{C}_{SF}) \leq H(\mathcal{X}) + 1$$

2 De bien mauvais dicos

Explorons toutes les raisons pour lesquelles les minions ont failli à trouver de bons dictionnaires:

a) Le dictionnaire proposé ici n'utilise même pas un code binaire, mais un code "unaire". L'encodage du mot BANANA avec ce dictionnaire donne 1111111111, autrement dit quelque chose de tout à fait illisible! Il faudrait rajouter des caractères de séparation entre les mots de code si on voulait y comprendre quelque chose.

b) Ce dictionnaire est d'un côté celui qui utilise le moins de bits (7) pour represénter le mot BANANA, mais le code binaire ainsi généré n'est pas sans préfixe et est donc difficilement décodable: en particulier, quant on lit 11, on ne sait jamais si on doit interpréter ça comme B ou NN. De manière correspondante, la longueur moyenne du code binaire utilisé ici est $\frac{7}{6}$, qui est en dessous de la limite de Shannon (= l'entropie du mot $\simeq 1.46$; voir calcul ci-dessous).

c) Ici, le code est sans préfixe et donc décodable de manière unique, mais la longueur moyenne du code $L(\mathcal{C}) = \frac{10}{6} = \frac{5}{3}$ n'est pas optimale; le mot de code associé à la lettre B est en fait inutilement trop long.

d) Ce dictionnaire utilise un code sans préfixe (du fait que tous les mots de code ont la même longueur) et est donc décodable de manière unique, mais il n'atteint pas la performance optimale, car sa longueur moyenne $L(\mathcal{C}) = 2$ (forcément, puisque tous les mots de code sont de longueur 2), qui est plus éloignée de l'entropie du mot que nécessaire.

e) Le code utilisé ici est aussi sans préfixe, donc décodable de manière unique, mais le mot de code court (1) est attribué à la lettre la moins fréquente. La longueur moyenne du code résultante vaut $L(\mathcal{C}) = \frac{11}{6} \simeq 1.88$, qui est loin d'être optimale.

f) Ce code cumule deux défauts: sa longueur moyenne n'est pas optimale et il n'est pas sans préfixe (donc difficilement décodable).

Finalement, un bon dictionnaire est le suivant (qu'on obtient indifféremment par l'algorithme de Shannon-Fano ou de Huffman):

A	N	B
0	10	11

Il utilise 9 bits en tout pour la représentation du mot BANANA et sa longueur moyenne vaut donc $L(\mathcal{C}) = \frac{9}{6} = \frac{3}{2}$, qui est très proche de l'entropie du mot: $H(\mathcal{X}) = \frac{1}{2}\log_2(2) + \frac{1}{3}\log_2(3) + \frac{1}{6}\log_2(6) = \frac{2}{3} + \frac{1}{2}\log_2(3) \simeq 1.46$.

3 A la recherche d'un trésor

a) On doit chaque fois se mettre au point sur la grille qui est au milieu des possibilités qui nous restent, de sorte à diviser cet ensemble de possibilités en quatre parties égales; vu qu'il y a 64 possibilités au départ, le nombre de questions à poser est $\log_4(64) = 3$.

b) (5,5), (7,7), (6,8) (et la suite des réponses correspondantes de l'oracle est NE, NO, SE).

c) Deux possibilités:

- En vous basant sur la suite des questions ci-dessus, vous encodez chaque réponse obtenue avec 2 bits, p.ex.: NE=00, NO=01, SE=10, SO=11 (on peut aussi dire, N=0, S=1, O=0, E=1 d'ailleurs, mais se rappeler alors qu'on indique d'abord la direction nord-sud avant la direction est-ouest). Dans le cas présent, la séquence est donc 000110.

- Sans rapport avec le jeu des questions ci-dessus, on peut aussi simplement encoder la position de la case avec 6 bits également: 3 bits pour la position horizontale (de 1 à 8: plus précisément, on va encoder le nombre moins 1, comme dans l'exercice sur le codage par plages: ainsi, 000 encode 1, 001 encode 2, jusqu'à 111 qui encode 8), et 3 bits pour la position verticale. Avec cette représentation, la position du trésor ci-dessus est (6,7), et donc l'encodage est 101110 (on utilise la convention que l'abscisse vient avant l'ordonnée: pas besoin donc d'utiliser un bit pour ça: ça fait partie de la manière dont on définit notre dictionnaire).

4 Pour le plaisir: codage par plages (run-length encoding)*

a) Vu que chaque ligne de la première image est unicolore, son code RLE est donné par

1111|1111|0111|0111|1111|1111|0111|0111 (longueur totale de 32 bits)

Sur chaque ligne de la deuxième image, on voit le même motif de 4 pixels noirs (encodés par 1011) suivis de 4 pixels blancs (encodés par 0011), donc le code RLE de l'image est

1011|0011|1011|0011|... (longueur totale de 64 bits)

Sur la troisième image, les deux premières lignes sont des séries alternées de noir et blanc, donc avec le codage RLE, chacun des pixels est représenté par 4 bits! (0000 si le pixel est blanc, ou 1000 si le pixel est noir). Donc rien que l'encodage des deux premières lignes requiert $16 \times 4 = 64$ bits[1]! Les six prochaines lignes ne requièrent chacune que 4 bits, comme sur la première image. Au total, on a donc besoin de $64 + 24 = 88$ bits; clairement pas une compression!

b) Dans cet exercice, on a choisi d'encoder l'image ligne par ligne, mais rien ne nous empêche de faire la même chose colonne par colonne. Pour préciser notre choix dans le code RLE, il suffit d'ajouter un bit au début: 0 pour "ligne par ligne" et 1 pour "colonne par colonne". On peut aini représenter la deuxième image avec 33 bits au total:

1|1111|1111|1111|1111|0111|0111|0111|0111

c) On voit bien que le codage RLE des deux premières lignes est loin d'être optimal. Une façon de remédier à ça est de changer la structure des paquets, en ajoutant un bit au début de chaque paquet (dont la longueur passe donc à 5 bits): si ce bit vaut 0, alors les quatre prochains bits sont à interpréter au sens du code RLE ci-dessus; par contre, si le premier bit du paquet vaut 1, alors les quatre prochains bits sont simplement les couleurs des 4 pixels de l'image. Ainsi, la troisième image sera encodée par la séquence de bits suivante:

11010|11010|10101|10101|00111|00111|01111|01111|00111|00111

pour une longueur totale de 50 bits (51 si on ajoute le premier bit 0 de la partie b) pour dire qu'un procède ligne par ligne).

[1] On néglige ici les deux pixels blancs qui se suivent de la fin de la première ligne au début de la seconde.