Semaine 10 : Série d'exercices sur la compression de données

1 [N2] Algorithmes de compression

Supposons que nous ayons un texte avec les nombres d'apparitions suivants :

	A	B	C	D	E
nb. app.	39	17	16	15	13

1. Construire le code de Shannon-Fano correspondant.
2. Construire le code de Huffman correspondant.
3. Quelle est la longueur moyenne de chaque code ?
4. Comparer à la limite théorique donnée par le théorème de Shannon.

2 [N3] Un peu de magie noire

a) Calculez l'entropie de la séquence de lettres suivantes (sans l'espace) :

AVADA KEDAVRA

— en écrivant d'abord le résultat sous la forme $a + b\log_2(3) + c\log_2(5)$, où a, b, c sont des fractions de nombres entiers ;
— puis en calculant le résultat numérique ou en l'approximant avec $\log_2(3) \simeq 1.58$ et $\log_2(5) \simeq 2.32$.

b) Créez un dictionnaire pour cette même séquence de lettres à l'aide de l'algorithme de Shannon-Fano. Combien de bits utilisez-vous pour représenter la séquence ? Essayez différentes versions de l'algorithme et comparez.

c) Créez ensuite un dictionnaire à l'aide de l'agorithme de Huffman. Combien de bits utilisez-vous pour représenter la séquence ? A nouveau, essayez différentes versions de l'algorithme et comparez.

d) Comparez les résultats obtenus aux points a), b) et c). Ceci est-il cohérent avec ce que vous avez appris en cours ?

Si vous en voulez encore...

3 [N3] Encore plus de codes

a) En utilisant l'algorithme de Shannon-Fano, représentez la séquence suivante (sans tenir compte des espaces) par une séquence de bits :

INFORMATION CALCUL ET COMMUNICATION

b) Combien de bits par lettre en moyenne sont-ils nécessaires pour représenter cette séquence ?

c) Mêmes questions pour l'algorithme de Huffman.

d) Calculez l'entropie de la séquence. Vérifie-t-on les inégalités vues en cours (sur l'entropie et les codes) ?

Indication : Si le calcul de l'entropie vous fatigue, essayez `http://www.shannonentropy.netmark.pl/` (attention à supprimer les espaces !)

e) Si vous vous restreignez à utiliser un code qui ne tient pas compte des probabilités d'apparition et qui utilise exactement le même nombre de bits pour chaque lettre, de combien de bits aurez-vous besoin pour représenter la séquence ?

Voici une autre séquence de lettres (où on oublie à nouveau les espaces, les accents, les traits d'union, les virgules et les apostrophes !) :

DIDON DINA, DIT-ON, DU DOS D'UN DODU DINDON

f) A priori, pouvez-vous deviner laquelle des deux séquences ci-dessus a la plus faible entropie ?

g) Répondez à nouveau aux questions **a)** à **e)** pour cette deuxième séquence de lettres.

Cours ICC : liens théorie $\longleftrightarrow$ Programmation

L'exercice 7 de la série 9 du cours de Programmation I vous propose de programmer de réaliser des codes de Huffman de textes.

Retrouvez tous les exercices de programmation liés à la partie théorie du cours sous le lien « Exercices de C++ spécifiques à ICC (lien programmation - théorie) » en bas de la page Moodle du cours, dans la section « Ressources complémentaires / Références ».

Pour le fun, la culture générale... (à la maison)

4 Enquête policière et littérature « scientifique »

J'ai lu un jour, dans une revue pseudo-scientifique, le problème suivant :

> « Les mathématiciens sont vraiment des gens étranges ! » dit un commissaire à sa femme. « L'autre jour, nous avions toute cette rangée de verres d'une réception dont nous savions qu'un (et un seul) était empoisonné. Évidemment, notre laboratoire aurait pu tester les verres les uns après les autres, mais cela aurait coûté très cher ! Il nous fallait donc trouver une procédure déterminant le verre empoisonné en le moins de tests possibles, moyennant des mélanges de petits échantillons prélevés dans les verres. Vint alors ce mathématicien. J'ignore d'ailleurs d'où il venait. Il regarda les verres, qu'il était vraisemblablement en train de compter, puis me dit en souriant : « Mon cher commissaire, choisissez un verre au hasard et testez-le ». « Mais ce serait un gaspillage d'argent ! » dis-je, « pourquoi effectuer un test inutile ? ». « Non », me répondit-il, « cela fait partie de la meilleure procédure ! On peut tester en premier un seul verre, peut importe lequel. » »
> « Et combien y avait-il de verres ? » demanda sa femme.
> « Je ne me souviens pas exactement. Entre 100 et 200 je pense. » répondit le commissaire.
>
> Combien y avait-il de verres ?

Voilà comment le problème était posé. À présent,

1. répondez au problème comme demandé par la revue ;
2. montrez que l'auteur de ce problème se trompe (et donc le mathématicien de l'histoire aussi) en :
   (a) déterminant la procédure optimale de test ;
   (b) calculant le nombre moyen de tests de cette procédure optimale ;
   (c) calculant le nombre moyen de tests d'après la procédure suggérée par le mathématicien de l'histoire.

Conclusion : méfiez-vous de vos lectures ! ;)

5 Codage par plages (run-length encoding)

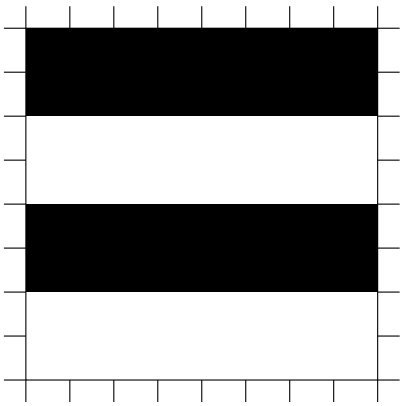

Image 1

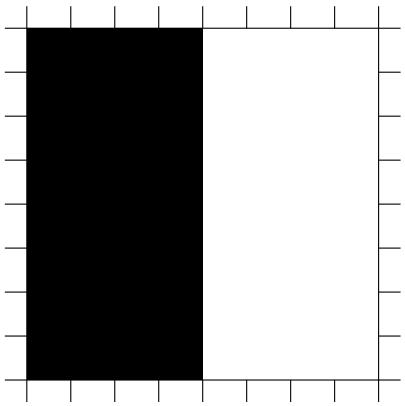

Image 2

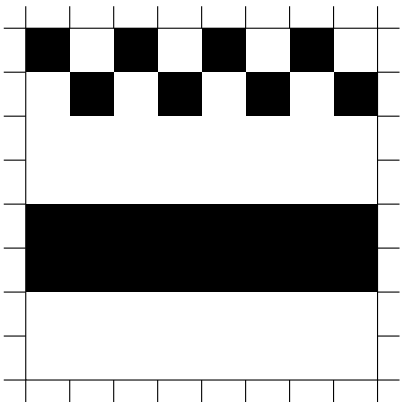

Image 3

Dans cet exercice, on considère une ancienne méthode de compression, utilisée principalement pour les images noir et blanc comme celles présentées ci-dessus. Il s'agit du *codage par plages* ou run-length encoding (RLE) en anglais. L'idée est la suivante : dans une image en noir et blanc, chaque pixel est représenté par un 1 (noir) ou un 0 (blanc). Pour comprimer une image avec $8 \times 8 = 64$ pixels, on transforme tout d'abord celle-ci un une séquence de 64 bits, en « lisant »l'image ligne par ligne. Ainsi l'image 1 ci-dessus est représentée de manière « brute »par la séquence de bits :

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Vu le grand nombre de 1 et de 0 consécutifs dans cette séquence, il semble qu'on peut économiser de l'espace mémoire en procédant comme suit : on divise la séquence en paquets de 4 bits de longueur ; dans chaque paquet, le premier bit symbolise la couleur (0 ou 1) de la suite de pixels et les 3 bits suivants indiquent en binaire le nombre de pixels consécutifs de cette couleur, moins 1. Exemples :

0010 signifie « 3 pixels consécutifs de couleur 0 »

1101 signifie « 6 pixels consécutifs de couleur 1 »

a) Quel est le codage RLE des trois images ci-dessus, et quelle est la longueur de ce code pour chaque image (c.-à-d. combien épargne-t-on de bits par rapport à la taille originale de l'image qui est de 64 bits) ?

b) Si vous avez résolu correctement la partie a), vous aurez constaté que le codage RLE de la seconde image est bien plus long que celui de la première image, alors que ces images sont très similaires par nature. Proposez une modification du format du codage qui considère qu'un nombre fixe de bits au début du codage sert à définir une convention utilisée pour le codage de toute l'image (en conservant le même type de paquet de 4 bits).

c) Si vous avez résolu correctement la partie a), vous aurez également constaté que le codage RLE de la troisième image pose problème. Cette fois-ci nous vous proposons de modifier la taille de paquet (sachant que cette taille reste utilisée pour toute l'image) et le rôle d'un ou de plusieurs bits de chaque paquet. Le but est d'introduire une plus grande flexibilité dans le codage pour pouvoir aussi bien coder des plages uniformes et des zones comme le début de l'image 3. Que proposez vous ?