ASR1 – TD1 : Codage et calcul

{ Jeremie.Detrey, Patrick.Loiseau, Nicolas.Veyrat-Charvillon } @ens-lyon.fr
http://perso.ens-lyon.fr/jeremie.detrey/06_asr1/
25, 26 et 29 septembre 2006

1 Codage...

Étant donné un ensemble B de mots de n bits et un ensemble I d'informations différentes, on appelle *codage* de I sur n bits toute fonction de B dans I. Le codage est *total* si la fonction est surjective. Si la fonction n'est pas injective, le codage est dit *redondant*.

1. Quel est le nombre minimum de bits nécessaire pour un codage total de N informations ?

2 Les entiers

2.1 Comme au néolithique

1. Comment appelle-t-on un petit caillou en latin ?

Les bergers de l'antiquité, qui ne savaient ni compter ni poser une addition, avaient une méthode utilisant cette technologie (et un système de numération unaire) pour faire la somme des effectifs de deux troupeaux.

2. Retrouvez-la.

3. Comment pose-t-on une multiplication en cailloux ?

2.2 Comme maintenant

1. Rappelez les principes du codage des entiers positifs en numération simple de position en base β.

2. Faites quelques additions d'entiers positifs sur 8 bits pour vous faire la main.

3. Proposez une manière de coder les entiers relatifs.

4. Essayez de faire quelques additions. Par exemple $16 + 64$ et $42 + (-17)$. Quels sont les problèmes ?

5. Trouvez une autre manière de coder les entiers relatifs qui soit plus pratique.
Astuce : pensez au complément à la base.

6. Faites quelques additions et quelques soustractions d'entiers relatifs codés en complément à 2 sur 8 bits.

7. Posez quelques multiplications de nombres de 4 bits en binaire.

2.3 Comme chez les souris

Le *code de Gray* sur n bits permet de coder les nombres entre 0 et $2^n - 1$ de telle sorte que le codage de deux nombres consécutifs ne diffère que d'un seul bit. Par exemple un codage de Gray sur trois bits donne les codes : $000, 001, 011, 010, 110, 111, 101, 100$.

1. Un tel codage s'appelle aussi binaire réfléchi, pourquoi ?
2. À quoi sert-il ?
3. Écrivez une procédure récursive donnant un codage de Gray d'un nombre quelconque de bits.

2.4 Pour ceux qui sont en avance

1. Comment coder un entier naturel non borné ? Trouvez un codage de n sur $n + 1$ bits, et un autre sur $2\lfloor \log_2 n \rfloor + 3$ bits. Peut-on faire encore mieux ?
2. Généralisez le tout au cas où, au lieu de coder vers des bits, on code vers l'alphabet $\{0, 1, 2\}$.

3 Détection et correction d'erreurs

3.1 Généralités

Étant donnée une source émettant une information codée sur n bits reçue par un récepteur, on définit les termes suivants :

- un codage est *auto-vérificateur* s'il permet de déceler une erreur de transmission, et
- un codage est *auto-correcteur* s'il permet de reconstituer l'information.

1. Trouvez des exemples simples de codages auto-vérificateurs ou auto-correcteurs.

Le *contrôle de parité* utilise $n - 1$ bits pour l'information et un bit de contrôle dit bit de parité. Celui-ci est positionné de telle sorte que le nombre total de bits à 1 de la configuration soit pair (parité paire) ou impair (parité impaire). On retrouve cette méthode dans le protocole de communication du lien série (UART/RS-232) par exemple.

2. Quelles sont les possibilités et les limites d'un tel codage ?

3.2 Codes de Hamming

Le *code auto-correcteur de Hamming* permet de corriger un bit erroné dans une information de 4 bits (i_3, i_2, i_1, i_0). On code cette information au moyen de 3 bits supplémentaires p_2, p_1 et p_0 : le codage est $(i_3, i_2, i_1, p_2, i_0, p_1, p_0)$, où :

- p_0 est le bit de parité paire du sous-mot (i_3, i_1, i_0, p_0),
- p_1 est celui du sous-mot (i_3, i_2, i_0, p_1), et
- p_2 est celui de (i_3, i_2, i_1, p_2).

À la réception de $(i_3, i_2, i_1, p_2, i_0, p_1, p_0)$, on vérifie ces trois parités, et on définit t_k par : $t_k = 0$ si p_k est correct et $t_k = 1$ sinon.

Par magie, (t_2, t_1, t_0) est alors le rang dans $(i_3, i_2, i_1, p_2, i_0, p_1, p_0)$ du bit erroné, écrit en binaire : 000 si l'information est correcte, 001 si p_0 est erroné, 010 si p_1 est erroné, 011 si i_0 est erroné, 100 si p_2 est erroné, 101 si i_1 est erroné, 110 si i_2 est erroné ou 111 si i_3 est erroné.

On parle ici du code $(7, 4)$: 7 bits transmis pour 4 bits d'information effective.

1. Vérifiez le fonctionnement correct de l'algorithme de correction sur quelques exemples.
2. Justifiez la propriété auto-correctrice d'un tel codage.
3. Peut-on définir un codage auto-correcteur de 4 bits utilisant moins de 3 bits supplémentaires ?
4. Tentez de généraliser l'algorithme de Hamming à d'autres tailles de mots.

3.3 *Bonus* : Comme à la NASA

Un des codes auto-correcteurs les plus efficaces à l'heure actuelle (tellement efficace qu'il est utilisé pour causer avec les bébêtes de la NASA qui se balladent sur Mars) est le codage Reed-Solomon. Plus proche de vous, il est aussi utilisé dans tout ce qui est télécommunications (ADSL, Usenet, téléphonie mobile, etc...) et dans les CD.

On souhaite donc coder n blocs B_i de k bits chacun. La première étape est de s'assurer que chaque bloc est correct. Ceci est réalisé en rajoutant un CRC (*Cyclic Redundancy Check*, un code auto-vérificateur) à chaque bloc.

L'idée parachutée est de considérer le polynôme $P(X)$ de degré $n-1$ et dont les coefficients sont donnés par les B_i :

$$P(X) = \sum_{i=0}^{n-1} B_i X^i.$$

1. Dans le cas où certains blocs sont faux, quelle propriété des polynômes permet de corriger $P(X)$ et donc de retrouver les valeurs de ces blocs ?
2. Est-ce si facile que ça en a l'air ?