Paris Université Diderot
L3 informatique
Sécurité 2017-2018

Corrigé exercice code

Exercice 1.— On considère l'ensemble suivant: S=$\{010101, 110011, 001111\}$

1. Quel est le code engendré par S?

   $\mathcal{C}$=$\{000000, 010101, 110011, 001111, 100110, 011010, 111100, 101001\}$

2. Donner une matrice génératrice sous forme standard de ce code (ou d'un code équivalent).

   La matrice génératrice G sous forme standard est

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

3. Quel est le codage de 101 ? 101 001

4. Décrire l'algorithme de codage. Quelle est sa complexité? Pour une entrée de taille n, l'algorithme lit un mot de 3 lettres abc de l'entrée, fait le codage $a(100110) + b(010101) + c(001111)$ et le met sur la sortie. L'algorithme est donc linéaire.

5. Quelle est la distance de Hamming de ce code? La distance de Hamming d'un code est le minimum des distances entre 2 mots du codes. Dans le cas d'un code linéaire c'est aussi le minimum des poids des mots du code 3. Tous les mots du code de l'exercice ont un poids supérieur ou égale à 1 donc la distance de Hamming est 3

6. Combien peut-on corriger d'erreurs? Un code de distance d détecte $d-1$ et corrige $\lfloor (d-1)/2 \rfloor$, donc ici 1

7. Donner une matrice de contrôle.

   Le code est un sous espace vectoriel de dimension 3 dans un espace vectoriel de dimension 6.

   Si G est sous la forme

$$G = \begin{bmatrix} 1 & 0 & 0 & a & b & c \\ 0 & 1 & 0 & x & y & z \\ 0 & 0 & 1 & u & v & w \end{bmatrix}$$

   i.e $G = Id.X$. Tout mot du code est le produit d'un mot de 3 lettres par G (i.e tous les mots du code sont obtenus par $(abc).G = a(100110) + b(010101) + c(001111)$

avec $a, b, c \in \{0, 1\}$)

L'espace vectoriel orthogonale est engendré par

$$H = \begin{bmatrix} a & x & u & 1 & 0 & 0 \\ b & y & v & 0 & 1 & 0 \\ c & z & w & 0 & 0 & 1 \end{bmatrix}$$

i.e $H = -X^t.Id = X^t.Id$

et H^t vaut

$$H^t = \begin{bmatrix} a & b & c \\ x & y & z \\ u & v & w \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} X \\ Id \end{bmatrix}$$

Ainsi le produit d'un élément de H (par exemple $byv010$) par un élément de G (par exemple $001uvw$) donne le $v + v = 0$. Ce qui fait que $G.H^t = (0, 0, 0)$

Si il y a une erreur e sur un mot de code α i.e on le mot $\beta = \alpha + e$. Comme le mot α est un mot du code il s'écrit comme $m.G$ où m est un mot de 3 lettres donc $\beta = mG$ On a $\beta H^t = mGH^t + eH^t = eH^t$. Si on a calculer avant tous les eH^t (les syndromes) on peur savoir ou était l'erreur. Dans notre cas P, on veut corriger 1 erreur, on calcule donc eH^t pour e=100000, e=010000, e=001000 etc....

pour e=100000, eH^t= abc la premiere ligne de H^t, etc.. Donc la $i$ème ligne de H^t correspond à une erreur sur le $i$ème bit.

8. Calculer les syndromes pour une erreur.

   pour 100 000 donne 110
   pour 010 000 donne 101
   pour 001 000 donne 111
   pour 000 100 donne 100
   pour 000 010 donne 100
   pour 000 001 donne 001

9. On suppose qu'il y a au plus une erreur dans le mot codé, à quel mot correspond le mot codé 101001? on calcule 101001.H^t= 000 donc a un mot du code. Le mot initial est en tête. C'est le code de 101

10. On suppose qu'il y a au plus une erreur dans le mot codé, à quel mot correspond le mot codé 101000? on calcule 101000.H^t= 001 donc ce n'est pas un mot du code. Par regarde les syndromes, l'erreur est sur le dernier bi. On corrige c'est donc 101001, c'est le code de 101.

    Si on avait eu le mot (en supposant toujours au plus une erreur) 110 100. on a 110100.H^t = 111 ce n'est pas un mot de code, l'erreur est sur le 3eme bit. Donc le mot du code est 111 100 et c'est le code de 111

11. Décrire l'algorithme de décodage. Quelle est sa complexité? L'entrée est de taille n. On décompose l'entrée en mot u de 6 bits on calcul $u.H^t$ soit on trouve 000 et le mot codé est les 3 premieres lettre de u que l'on met sur la sortie. Soit on trouve une valeur différente. On cherche dans H^t a quelle ligne i cela corresponds. On modifie le iême bit, on met les 3 premiers bits sur la sortie. L'algorithme de décodage est linéaire.