Université de Caen Normandie, UFR des Sciences
Licence 3

Module " Algorithmique, structures informatiques et cryptologie"
TP - Codage de Hamming(7,4) ; durée 2h30 ;

1 Présentation du code de Hamming (7,4)

Nous avons vu en cours et en TD le code de Hamming(7,4) transformant des blocs de 4 bits en blocs de 7 bits. Le code de Hamming (7, 4) est un code binaire de cardinal $K = 2^4 = 16$, de longueur $n = 7$ et de distance minimale $d = 3$. Le taux de transmission de ce code est donc $4/7 \simeq 0.57$ et sa capacité de correction e est 1.

Encodage : Pour coder le vecteur binaire $\mathbf{m} = [m_1, m_2, m_3, m_4]$, on construit le vecteur binaire $\mathbf{x} = [x_1, \ldots, x_7]$ tel que $x_3 = m_1,\ x_5 = m_2,\ x_6 = m_3,\ x_7 = m_4$, et les bits x_1, x_2, x_4 sont calculés par les sommes de contrôle suivantes :

$$x_1 + x_3 + x_5 + x_7 = 0 \bmod 2, \quad x_2 + x_3 + x_6 + x_7 = 0 \bmod 2, \quad x_4 + x_5 + x_6 + x_7 = 0 \bmod 2.$$

Les vecteurs $\mathbf{x}$ envoyés ont donc un nombre pair de 1 dans les positions 1,3,5,7 ainsi que 2,3,6,7 et 4,5,6,7. L'ensemble des $2^4 = 16$ vecteurs x possibles calculés comme ci-dessus forment le code de Hamming (7,4) et sont appelés mots de code.

Exemple : Soit $\mathbf{m} = [1, 0, 0, 0]$ le message à coder. Alors $\mathbf{x} = [1, 1, 1, 0, 0, 0, 0]$ (vérifier les indices et les trois sommes de contrôle).

Décodage : Soit $\mathbf{y} = [y_1, \ldots, y_7]$ le mot binaire reçu après la transmission de $\mathbf{x}$. On suppose qu'il y a une erreur dans la composante d'indice i, avec $i = i_0 + 2i_1 + 4i_2$ (la décomposition de i en base 2). Si il y a un nombre impair de 1 dans les composantes d'indices 1, 3, 5, 7, alors l'erreur se situe sur un indice i tel que $i_0 = 1$. De même, si il y a un nombre impair de 1 dans les composantes d'indices 2, 3, 6, 7 (respectivement 4, 5, 6, 7), alors l'erreur se situe sur un indice i tel que $i_1 = 1$ (respectivement $i_2 = 1$). Ainsi pour décoder, on calcule le syndrome $\mathbf{x} = [s_0, s_1, s_2]$ de $\mathbf{y}$ défini par $s_0 = y_1 + y_3 + y_5 + y_7 \bmod 2$, $s_1 = y_2 + y_3 + y_6 + y_7 \bmod 2$ et $s_2 = y_4 + y_5 + y_6 + y_7 \bmod 2$. Si $\mathbf{s} = [0, 0, 0]$ alors il n'y a pas d'erreur et s'il y a une erreur, elle se situe à l'indice $i = s_0 + 2 \cdot s_1 + 4 \cdot s_2$.

Exemple : Supposons que l'on reçoit le mot $\mathbf{y} = [0, 1, 1, 0, 0, 0, 0]$. Son syndrome est $\mathbf{s} = [1, 0, 0]$, l'erreur est donc à la place $1 + 0 \cdot 2 + 0 \cdot 4 = 1$ et on corrige y_1. Le bloc envoyé était alors $\mathbf{x} = [1, 1, 1, 0, 0, 0, 0]$ et le message envoyé $\mathbf{m} = [1, 0, 0, 0]$.

Structures de données utilisées : Comme dans les exemples ci-dessus, tous les mots binaires sont représentés par des tableaux de 0 et de 1.

2 Travail à réaliser

Question 1 : Ecrire une fonction d'encodage, `encode(m)`, qui prend en entrée un mot binaire $\mathbf{m} = [m_1, \ldots, m_4]$ et retourne le mot de code de Hamming $\mathbf{x} = [x_1, \ldots, x_7]$ correspondant.

Question 2 : Écrire une fonction `canal(x,p)` modélisant un canal symétrique binaire. Cette fonction prend en entrée un mot binaire $\mathbf{x}$ (de longueur quelconque) et une probabilité $p \in]0, 1[$. Elle retourne un vecteur $\mathbf{y}$ dont chacune de ses composantes est égale à celle de $\mathbf{x}$ avec une probabilité $1 - p$. Autrement dit, une composante de $\mathbf{x}$ est modifiée avec probabilité p.

Question 3 : Écrire une fonction `syndrome(y)` qui prend en entrée un mot binaire $\mathbf{y} = [y_1, \ldots, y_7]$ et qui calcule son syndrome $\mathbf{s} = [s_0, s_1, s_2]$.

Question 4 : Écrire une fonction `correction(y,s)` qui prend en entrée un mot binaire $\mathbf{y}$ et son syndrome $\mathbf{s}$ et qui retourne le mot binaire corrigé. Vérifier la fonction avec les exemples.

Question 5 : Écrire une fonction `int_to_tab(n,k)` qui prend deux entiers en paramètres avec $0 \leq n < 2^k$ et qui retourne l'écriture binaire de n sur k bits (quitte à ajouter des 0 sur les bits de poids fort). Le résultat sera un tableau de 0 et de 1. Vous pourrez vous inspirer des commandes suivantes :

```
n=25; bin(n)[2:]
n=25; print( [int(c) for c in bin(n)[2:]])
```

Question 6 : A l'aide des commandes précédentes, écrire une fonction `dico_hamming()` qui ne prend pas de paramètre en entrée et qui retourne un dictionnaire dont les clés sont les 2^4 entiers de 0 à 15 et dont les valeurs correspondent aux codages de Hamming des mots binaires sur 4 bits des entiers de 0 à 15.

Question 7 : Écrire une fonction `dist_hamming(x,y)` qui prend en entrée deux vecteurs binaires $\mathbf{x}$ et $\mathbf{y}$ de même longueur et qui retourne la distance de Hamming $d_H(\mathbf{x}, \mathbf{y})$ entre x et y (nombre de composantes différentes entre x et y).

Question 8 : Écrire une fonction `dist_minimale(dico)` qui prend en entrée un dictionnaire `dico` dont les valeurs sont des tableaux binaires de même taille représentant les mots d'un code et qui calcule la distance minimale du code. Vérifier la distance minimale du code de Hamming (7,4) (vous devez trouver 3).

Question 9 : Le décodage est correct si le nombre d'erreurs est inférieur ou égal à la capacité de correction e du code. La probabilité qu'il y ait au plus e erreurs à travers le canal symétrique (de probabilité p) est $\sum_{i=0}^{e} C_n^i p^i (1-p)^{n-i}$, où n est la longueur du mot envoyé et le coefficient binomial $C_n^i = \frac{n!}{(n-i)!i!}$. Ecrire une fonction `calcule_proba(n,p,e)` qui calcule la probabilité que le décodage soit correct à partir de n, p et e. Si $p = 0.25$, la probabilité que le décodage soit correct est 0.445 pour le code de Hamming (7,4).

Question 10 : Écrire une fonction `calcule_proba_empirique_hamming(k,p)` dont l'objectif est de calculer empiriquement la probabilité qu'un décodage soit correct. La fonction génère k mots de code aléatoires, fait passer ces k mots de code dans la canal binaire symétrique de paramètre p et corrige les mots obtenus en sortie du canal. La fonction retourne la probabilité empirique de succès du décodage donnée par le nombre de corrections réussies divisé par le nombre total de tests (ici k). Faites le test avec $k = 10000$ et $p = 0.25$. Vous devez trouver un résultat proche de 0.445.

3 Trace d'exécution

```
Debut du programme principal
********* encode ***********
encode([0,0,0,0])= [0, 0, 0, 0, 0, 0, 0]
encode([0,0,0,1])= [1, 1, 0, 1, 0, 0, 1]
encode([0,0,1,0])= [0, 1, 0, 1, 0, 1, 0]
encode([0,1,0,0])= [1, 0, 0, 1, 1, 0, 0]
encode([1,0,0,0])= [1, 1, 1, 0, 0, 0, 0]
encode([1,1,0,0])= [0, 1, 1, 1, 1, 0, 0]
```

```
******* test du canal *************
2.498  est proche de  0.25

******** syndrome ************
y= [0, 1, 1, 1, 1, 0, 0]  syndrome(y)= [0, 0, 0]
y= [1, 1, 1, 1, 1, 0, 0]  syndrome(y)= [1, 0, 0]
y= [0, 0, 1, 1, 1, 0, 0]  syndrome(y)= [0, 1, 0]
y= [0, 1, 0, 1, 1, 0, 0]  syndrome(y)= [1, 1, 0]
y= [0, 1, 1, 0, 1, 0, 0]  syndrome(y)= [0, 0, 1]
y= [0, 1, 1, 1, 0, 0, 0]  syndrome(y)= [1, 0, 1]
y= [0, 1, 1, 1, 1, 1, 0]  syndrome(y)= [0, 1, 1]
y= [0, 1, 1, 1, 1, 0, 1]  syndrome(y)= [1, 1, 1]

********* correction ***********
y= [0, 1, 1, 1, 1, 0, 0]
syndrome(y)= [0, 0, 0]
x= [0, 1, 1, 1, 1, 0, 0]

y= [1, 1, 1, 1, 1, 0, 0]
syndrome(y)= [1, 0, 0]
x= [0, 1, 1, 1, 1, 0, 0]

y= [0, 0, 1, 1, 1, 0, 0]
syndrome(y)= [0, 1, 0]
x= [0, 1, 1, 1, 1, 0, 0]

y= [0, 1, 0, 1, 1, 0, 0]
syndrome(y)= [1, 1, 0]
x= [0, 1, 1, 1, 1, 0, 0]

y= [0, 1, 1, 0, 1, 0, 0]
syndrome(y)= [0, 0, 1]
x= [0, 1, 1, 1, 1, 0, 0]

y= [0, 1, 1, 1, 0, 0, 0]
syndrome(y)= [1, 0, 1]
x= [0, 1, 1, 1, 1, 0, 0]

y= [0, 1, 1, 1, 1, 1, 0]
syndrome(y)= [0, 1, 1]
x= [0, 1, 1, 1, 1, 0, 0]

y= [0, 1, 1, 1, 1, 0, 1]
syndrome(y)= [1, 1, 1]
x= [0, 1, 1, 1, 1, 0, 0]

********* int_to_tab ***********

int_to_tab( 0 ,4)= [0, 0, 0, 0]
```

```
int_to_tab( 1 ,4)= [0, 0, 0, 1]
int_to_tab( 2 ,4)= [0, 0, 1, 0]
int_to_tab( 3 ,4)= [0, 0, 1, 1]
int_to_tab( 4 ,4)= [0, 1, 0, 0]
int_to_tab( 5 ,4)= [0, 1, 0, 1]
int_to_tab( 6 ,4)= [0, 1, 1, 0]
int_to_tab( 7 ,4)= [0, 1, 1, 1]
int_to_tab( 8 ,4)= [1, 0, 0, 0]
int_to_tab( 9 ,4)= [1, 0, 0, 1]
int_to_tab( 10 ,4)= [1, 0, 1, 0]
int_to_tab( 11 ,4)= [1, 0, 1, 1]
int_to_tab( 12 ,4)= [1, 1, 0, 0]
int_to_tab( 13 ,4)= [1, 1, 0, 1]
int_to_tab( 14 ,4)= [1, 1, 1, 0]
int_to_tab( 15 ,4)= [1, 1, 1, 1]

******** dico_haming ************

dico_hamming()= {0: [0, 0, 0, 0, 0, 0, 0], 1: [1, 1, 0, 1, 0, 0, 1],
2: [0, 1, 0, 1, 0, 1, 0], 3: [1, 0, 0, 0, 0, 1, 1], 4: [1, 0, 0, 1, 1, 0, 0],
5: [0, 1, 0, 0, 1, 0, 1], 6: [1, 1, 0, 0, 1, 1, 0], 7: [0, 0, 0, 1, 1, 1, 1],
8: [1, 1, 1, 0, 0, 0, 0], 9: [0, 0, 1, 1, 0, 0, 1], 10: [1, 0, 1, 1, 0, 1, 0],
11: [0, 1, 1, 0, 0, 1, 1], 12: [0, 1, 1, 1, 1, 0, 0], 13: [1, 0, 1, 0, 1, 0, 1],
14: [0, 0, 1, 0, 1, 1, 0], 15: [1, 1, 1, 1, 1, 1, 1]}

*********dist_hamming ***********

x= [1, 1, 0, 1, 0, 0, 1]
y= [0, 1, 0, 1, 0, 1, 0]
dist_hamming(x,y)= 3

******** dist_minimale ************

dist_minimale(dico_hamming())= 3

********proba d'erreur théorique ************

proba erreur théorique= 0.4449462890625

*********proba erreur empirique ***********

proba erreur empirique= 0.44
```