Rapport Codes Correcteurs d'Erreurs

LE Doan Phuoc – PHAM Tien Duy

STI 3A

I. Première tâche : l'encodage

1. Exercice 4

On a deux étapes pour ce l'algorithme :

- Première étape de l'algorithme :

- Placer les 1 sur la diagonale et les 0 en dessous :

On crée d'abord des matrices sur lesquels on pourra faire des opérations.
Car H = L| R on veut transformer R en l'identité ainsi effectue toutes les opérations qui vonts suivre sur R puis on reproduira toutes les opérations d'échange et d'additions faites précédemment, sur H Après tout ça on obtient H'=(M|ID).

- Il faut d'abord réussir placer les 1 sur la diagonale a partir de la colonne n – k pour cela on parcours les lignes de la première colonnes et on vérifie si le premier indice de la première ligne vaut 1 puis on met des 0 dans la colonne en dessous du 1 et ajoutant les lignes qui ont l'indice qui vaut 1 avec la première ligne qui contient le 1 de l'identité. Dans le cas ou le premier indice de la 1ere ligne ne vaut pas 1 il faut faire une recherche d'une ligne qui contient 1 dans la même colonne et l'échanger pour placer le 1 sur la diagonale.

- On passe ensuite à la 2eme colonne et on fait la meme chose placement du 1 sur la diagonale avec recherche de 1 dans les lignes et echange puis placement des 0 sous la diagonale en faisant la somme des indices qui valent 1 avec le 1 de la diagonale.

- On continue jusqu'à la dernière colonne puis on obtient une matrice avec des 1 sur la diagonale et des 0 sous la diagonale, il manque plus que les 0 au dessus de la diagonale pour obtenir la matrice identité.

- Pour faire cela on va parcourir les colonnes en partant de la fin car la dernière ligne ne contient que des 0 sauf le dernier indice. On vérifie si les éléments de la dernière colonne valent 1 si oui on fait la somme entre la dernière ligne et celle sur laquelle on trouve le 1 dans la dernière colonne. Et passe ensuite à l'avant dernière colonne et on fait la même chose on fait cela en parcourant toutes les colonnes en partant de la fin. En résumé, on part de la fin on vérifie s'il y a des 1 au dessus de la

diagonale si oui on ajoute cette ligne a celle qui contient le 1 sur la diagonale pour mettre la valeur à 0.

- On passe à la deuxième colonne si non on parcourt les lignes de la colonne a la recherche de 1 puis on fait un échange entre ces 2 lignes. Ensuite, on continue de la même manière en passant a la colonne suivante en vérifiant la 2eme ligne puis on fait pareil jusqu'à obtenir que des 1 sur la diagonale.

- Deuxième étape de l'algorithme :
- Mettre des 0 sous la diagonale de 1 :

Pour faire cela pour chaque élément d'une colonne on vérifie si les valeurs en dessous du 1 de la diagonale valent 1 Si oui on ajoute cette ligne avec celle qui contient le 1 de l'identité et ainsi de suite.

- A la fin de tout cela après avoir reproduit sur la matrice H les actions faites sur R on obtient : H' = M|ID.

```
Matrice de controle H :

[[0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1]
 [0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0]
 [1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0]
 [0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0]
 [1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0]]
```

```
Forme systématique de H :

[[1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]]
```

2. Exercice 6

- On a une matrice contenue dans le fichier matrix-15-20-3-4 qu'on appellera H et encode le mot suivant 10101. Pour encoder un mot en général il suffit de le multiplié par la matrice génératrice sous forme systématique.

```
Mot binaire u :

1 0 1 0 1
```

- Dans notre cas on a donc d'abord obtenu la forme systématique de H avec sysTransform puis on renvoi la matrice génératrice sous forme systématique avec genG de la forme H' = M|id.

```
Matrice génératrice G :

[[1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0]
 [0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0]
 [0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 1]
 [0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 1 1 0 1 0]
 [0 0 0 0 1 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0]]
```

- A la fin on multiplie donc u par H' et on obtient un encodage de u

  x = 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1

```
Encodage de u (x=u.G) :

[[1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1]]
```

- On a aussi le syndrome de x : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pour vérifier qu'il n'y a pas d'erreur dans le mot encodé.

```
Syndrome de x (s=H.x^t) :

[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

II. Deuxième tâche : le décodage

1. Exercice 7

- On a le graphe de Tanner

```
Graphe de Tanner :

0|  3  9 10            0|  3  9 16 19
0|  1  4 10            0|  1  2  6  9
0|  1  5 13            0|  6 11 12 17
0|  0 11 12            0|  0  4  7 19
0|  3  7 11            0|  1  7 13 15
0| 10 12 14            0|  2 15 16 17
0|  1  2  7            0|  7  8  9 11
0|  3  4  6            0|  4  6 15 18
0|  6 12 13            0| 12 14 18 19
0|  0  1  6            0|  0 13 14 16
0| 11 13 14            0|  0  1  5 17
0|  2  6 11            0|  3  4 10 11
0|  2  8 14            0|  3  5  8 14
0|  4  9 14            0|  2  8 10 18
0|  8  9 12            0|  5 10 12 13
0|  4  5  7
0|  0  5  9
0|  2  5 10
0|  7  8 13
0|  0  3  8
```

2. Exercice 9

- Au début, le mot encoder x = 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1, puis on crée les quatre vecteurs d'erreurs on récupère le vecteur d'erreur, le mot de code bruit, le syndrome et la correction.

```
------------------------------------
  Bruitage et correction du mot x :
------------------------------------
```

- On observe que dans le 1er vecteur il y a 2 erreurs, le 2ème vecteur il y a 2 erreurs, le 3ème a aussi 2 erreurs et pour finir le 4eme vecteur contient 3 erreurs on ne peut pas le corriger car le maximum que l'on peut corriger est deux erreurs.
- On obtient comme résultat :
+ 1er vecteur : la correction marche et on l'obtient cela vaut :
  1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1

```
Mot de code x :
[[1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1]]

Vecteur d'erreurs e1 :

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

Mot de code bruité y1=x+e1 :

[[1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1]]

Syndrome de y1 :

[[1 0 0 1 1 0 1 0 0 0 0 1 1 0 0]]

Correction x1 de y1 :

[[1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1]]

x1 = x : true
```

+ 2ème vecteur : la correction marche et le mot corrigé vaut :
  1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1

```
Vecteur d'erreurs e2 :

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

Mot de code bruité y2=x+e2 :

[[1 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1]]

Syndrome de y2 :

[[1 0 0 1 0 0 0 1 0 0 0 0 1 0 0]]

Correction x2 de y2 :

[[1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1]]

x2 = x : true
```

+ 3ème vecteur : la correction marche encore et le mot corrigé vaut :
  1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1

```
Vecteur d'erreurs e3 :

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

Mot de code bruité y3=x+e3 :

[[1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1]]

Syndrome de y3 :

[[0 1 0 0 0 0 0 0 0 1 1 0 0 0 1]]

Correction x3 de y3 :

[[1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1]]

x3 = x : true
```

+ 4ème vecteur : la correction échoue car on n'a ni correction et notre programme renvoi : -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

```
Vecteur d'erreurs e4 :

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

Mot de code bruité y4=x+e4 :

[[1 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1]]

Syndrome de y4 :

[[1 0 0 1 0 0 0 0 1 0 0 0 1 1 0]]

Correction x4 de y4 :

[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]

x4 = x : false
```

III. Troisième tâche : l'évaluation

1. Exercice 10

- Longueur des mots de source k = 2048
- Longueur des mots de code n = 6144
- Redondance r = n - k = 4096
- Rendement = k/n = 2048/6144 = 1/3.
- Calcul nombre d'erreurs moyenne que contiennent les mots reçus : on fait n*p= 6144*(1/50) = 122 erreurs.

2. Exercice 13

- Pour le décodage du mot y = x + e avec round = 200 itérations, on observe que le décodage a réussi.
- On a les résultats pour w = 124, w = 134, w = 144, w = 154 :

```
Pour w=124,
Nombre de mots correctement décodés : 10000
Nombre d'échecs : 0
Nombre d'erreurs : 0

0.0% de cas critiques dont :
    -0.0% d'echecs
    -0.0% d'erreurs
```

```
Pour w=134,
Nombre de mots correctement décodés : 9997
Nombre d'échecs : 3
Nombre d'erreurs : 0

0.030000000000001137% de cas critiques dont :
    -0.03% d'echecs
    -0.0% d'erreurs
```

```
Pour w=144,
Nombre de mots correctement décodés : 9896
Nombre d'échecs : 104
Nombre d'erreurs : 0

1.039999999999992% de cas critiques dont :
    -1.04% d'echecs
    -0.0% d'erreurs
```

```
Pour w=154,
Nombre de mots correctement décodés : 8563
Nombre d'échecs : 1437
Nombre d'erreurs : 0

14.370000000000005% de cas critiques dont :
    -14.37% d'echecs
    -0.0% d'erreurs
```