TD1

Exo 1

Remarques

Le codage 1 est un codage de longueur fixe (longueur 2).

Le codage 2 est un codage de longueur variable et 'préfix', c.à.d. le code d'une lettre n'est pas le préfixe du code d'une autre lettre. Et donc tout message peut être decoder efficacement et de manière non-ambiguë. Par exemple 1011001001110 est decodé comme 10 110 0 10 0 111 0 $\rightarrow a_2a_3a_1a_2a_1a_4a_1$.

Rappel du cours

Pour un alphabet source $A = \{a_1, a_2, \ldots, a_n\}$ et un codage où chaque letter de A et codée (représentée) par un mot binaire, *la longueur moyenne* est :

$$L = p(a_1) \cdot \ell(a_1) + p(a_2) \cdot \ell(a_2) + \cdots + p(a_n) \cdot \ell(a_n),$$

où

- $p(a_i)$ est la probabilité d'apparition de la letter a_i
- $\ell(a_i)$ est la longueur (nombre de bits) du mot binaire qui code la letter a_i.

L est donc la moyenne pondérée (avec les poids $p(a_i)$) des longueurs du codage de chaque lettre. Intuitivement, L est le nombre moyenne de symboles binaires necessaires pour coder une lettre de l'alphabet A.

Correction

La longueur moyenne L est

- si les quatre lettres sont équiprobables
  - pour le codage 1 : $L = \mathbf{2}$
  - pour le codage codage 2 : $L = \frac{1}{4} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 3 + \frac{1}{4} \cdot 3 = \mathbf{2,25}$
- si la distribution de probabilité est $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{8}$
  - pour le codage 1 : $L = \mathbf{2}$
  - pour le codage codage 2 : $L = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = \mathbf{1,75}$

Moralité Avec une distribution équiprobable on ne peut pas faire mieux qu'un codage de longueur fixe (codage 1 dans notre cas). Si les lettres ne sont pas équiprobables, on peut envisager un codage de longueur variable (le codage 2 dans notre cas) pour besser la longuer moyenne.

Exo 2

Remarques

Avant commencer l'exo 2 il et important de rappeler (les étudiants ne sont pas forcement très habitués ...) :

- la fonction $x \mapsto 2^x$ sur $\mathbb{N}$ et $\mathbb{R}$
- la fonction inverse, $x \mapsto \log_2 x$.

On commence par deux exemples.
• Si $n = 10$ (la source émet 10 lettres équiprobables), supposons que les letters sont $0, 1, 2, \ldots 9$ (les 10 chiffres sur notre téléphone), alors on peut voire que on a besoin de 4 bits (valeurs binaires) par lettre (le 1èr tableau ci-dessous).
• Si $n = 16$, supposons que les letters sont $0, 1, 2, \ldots 9, A, B, C, D, E, F$ (les 16 chiffres du système hexadécimal), alors 4 bits (4 valeurs binaires) par lettre sont suffisants (le 2ème tableau).

symbole	codage (binaire)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

symbole	codage (binaire)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Rappel du cours

Pour un alphabet source $A = \{a_1, a_2, \ldots, a_n\}$ *l'entropie* est :

$$\begin{aligned} H = & \quad p(a_1) \cdot \log_2 \tfrac{1}{p(a_1)} + p(a_2) \cdot \log_2 \tfrac{1}{p(a_2)} + \cdots + p(a_n) \cdot \log_2 \tfrac{1}{p(a_n)} \\ = & \quad -p(a_1) \cdot \log_2 p(a_1) - p(a_2) \cdot \log_2 p(a_2) - \cdots - p(a_n) \cdot \log_2 p(a_n) \end{aligned}$$

où $p(a_i)$ est la probabilité d'apparition de la letter a_i.

Il faut noter que : (1) l'entropie ressemble à la longueur moyenne, (2) l'entropie ne dépend pas du codage.

Correction

Question : De combien de valeurs binaires on a besoin pour coder chaque lettres?
Réponse : $\lceil \log_2 n \rceil$, qui est aussi la longeure moyenne de notre codage
($\lceil \cdot \rceil$ est la partie entière supérieure)

Question : Quelle est l'entropie de la source ?
Réponse : Comme la probabilité de chaque letter est $\frac{1}{n}$, l'entropie est $H = n \cdot \frac{1}{n} \cdot \log_2 \frac{1}{\frac{1}{n}} = \log_2 n$.

Question : Si $n = 2^r$?
Réponse : Dans ce cas $\lceil \log_2 n \rceil = \lceil \log_2 2^r \rceil = \log_2 2^r = r$, et donc la longueur moyenne est égale à l'entropie. C'est le cas pour $n = 16 = 2^4$ pour coder les chiffres du système hexadécimal.
Il faut remarquer que dans tous les cas la longeure moyenne $\geq$ à l'entropie.

Exo 3

Rappel du cours

La quantité d'information associée à un événement est

$$I = \log_2 \frac{1}{p} = -\log_2 p$$

où p est la probabilité que l'événement se produise. Plus la probabilité que l'événement se produise est faible, plus la quantité d'information associée est élevée.

Remarques

Par la suite, on sera amené à calculer des $\log_2$, et sur la plupart des calculatrices (téléphones) on a que $\log_{10}$. Pour avoir, par exemple $\log_2 2,5$, on peut utiliser le site WolframAlpha. Ou la formule pour changer la base

$$\log_2 x = 3,32 \cdot \log_{10} x$$

Correction

Question : Calculer $I(b)$, la quantité d'information liée à l'apparition d'une boule blanche
Réponse : $I(b) = \log_2 \frac{5}{2} = 1,3219$

Question : Calculer $I(n)$, la quantité d'information liée à l'apparition d'une boule noire
Réponse : $I(n) = \log_2 \frac{5}{3} = 0,7369$

Question : Calculer H, l'entropie du système
Réponse :

$$H = p(b) \cdot I(b) + p(n) \cdot I(n),$$

où $p(b)$ est la probabilité de tirer une boule blanche et $p(n)$ est la probabilité de tirer une boule noire. On a donc

$$H = \frac{2}{5} \cdot \log_2 \frac{5}{2} + \frac{3}{5} \cdot \log_2 \frac{5}{3} = 0,9709$$

On remarque que l'entropie est la moyenne pondérée (avec les poids les probabilités p_i) des quantités d'information

Exo 4

• La longueur moyenne est :

$$\begin{aligned} L &= p(a) \cdot \ell(a) + p(b) \cdot \ell(b) + p(c) \cdot \ell(c) \\ &= \tfrac{1}{2} \cdot 1 + \tfrac{3}{16} \cdot 2 + \tfrac{5}{16} \cdot 2 \\ &= 1,5. \end{aligned}$$

• L'entropie est :

$$\begin{aligned} H &= p(a) \cdot \log_2 \tfrac{1}{p(a)} + p(b) \cdot \log_2 \tfrac{1}{p(b)} + p(c) \cdot \log_2 \tfrac{1}{p(c)} \\ &= \tfrac{1}{2} \cdot \log_2 2 + \tfrac{3}{16} \cdot \log_2 \tfrac{16}{3} + \tfrac{5}{16} \cdot \log_2 \tfrac{16}{5} \\ &= 1,47. \end{aligned}$$

Remarques

• $L \geq H$

Exo 5

Rappel du cours

L'efficacite d'un codage est

$$\eta = H/L,$$

où H est l'entropie et L est la longueur moyenne. Et $\eta \leq 1$

Correction

Question : Quelle est l'efficacite du premier codage ?
Réponse :

- l'entropie H du premier codage est

$$H = p(0)\cdot\log_2 \frac{1}{p(0)} + p(1)\cdot\log_2 \frac{1}{p(1)} + \cdots + p(9)\cdot\log_2 \frac{1}{p(9)} = 10\cdot\frac{1}{10}\cdot log_2 10 = log_2 10 = 3,32$$

  où $p(0) = \frac{1}{10}$ est la probabilité d'apparition de la lettre 0, ...

- la longueur moyenne du premier codage est $L = 4$

et donc l'efficacité est $\eta = H/L = \frac{3,32}{4} = 0,83$

Question : Quelle est l'efficacite du deuxième codage ?
Réponse : Pour le deuxième codage on a 100 'lettres' : $a_0 = 00$, $a_1 = 01$, ..., $a_{99} = 99$, chacune avec la probabilité $\frac{1}{100}$. Dans ce cas, après des calcules on a

- l'entropie H est $\log_2 100 = 6,64$
- La longueur moyenne du codage est $\lceil \log_2 100 \rceil = 7$

et on obtient l'efficacité $\eta = H/L = \frac{6,64}{7} = 0,95$

Moralité

En passant du premier codage au deuxième, on passe d'une efficacité de 83% à une de 95%.

La redondance est définie comme $1 - \eta = 1-$l'efficacité. En passant du premier codage au deuxième, on passe d'une redondance de 17% à une de 5%.

Exo 6

Je laisse les étudiants calculer l'entropie, c'est laborieux et sans un interêt pariculier. Le codage demandé est obtenu avec l'algorithme de Huffman, décrit ci-dessous.
Algorithme de Huffman :
Itérativement

- on fusionne les deux lettres avec les fréquences les plus faibles, disons x et y, dans une nouvelle lettre fictive xy, ayant comme fréquence la somme des fréquences de x et de y
- on trie le tableau par ordre décroissant de fréquences
- on met à jour l'arbre de Huffman qui au débur est vide

jusqu'à ce qu'il reste une seule letter dans le tableau.

Dans l'arbre (de Huffman) ainsi obtenu, on étiquette les branches à gauche par 0 et celles à droite par 1. Dans cet arbre, si on parcourt les branches entre la racine et une lettre, et on collecte les étiquettes des brances, on obtient le code de la lettre.

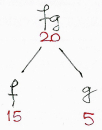
Correction

Le tableau d'origine est (les deux dernières lettres, en rouge, ont la fréquence la plus faible) et l'arbre de Huffman est vide.

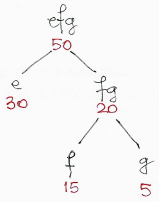
lettre	nb. occurences
a	70
b	60
c	45
d	35
e	30
f	15
g	5

Les itérations de l'algo. vont donner les transformations suivantes.

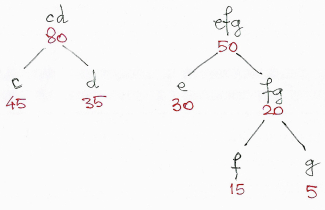
lettre	nb. occurences
a	70
b	60
c	45
d	35
e	30
fg	20

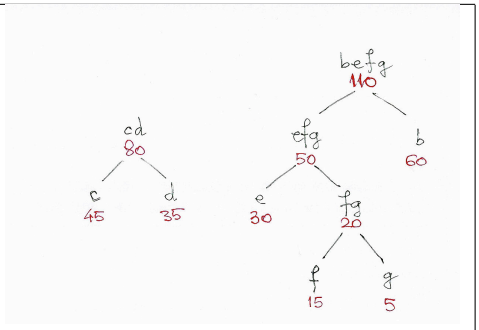


lettre	nb. occurences
a	70
b	60
efg	50
c	45
d	35

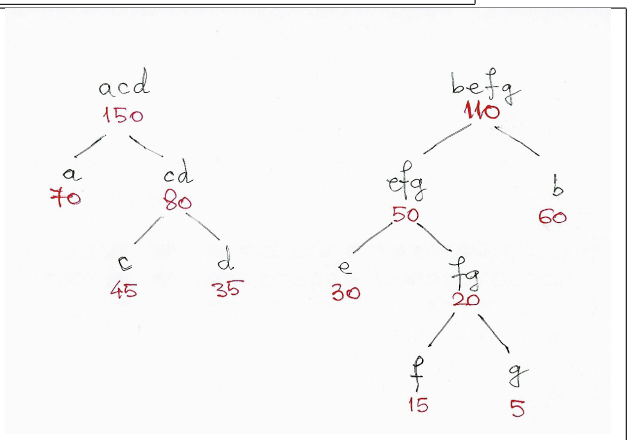


lettre	nb. occurences
cd	80
a	70
b	60
efg	50

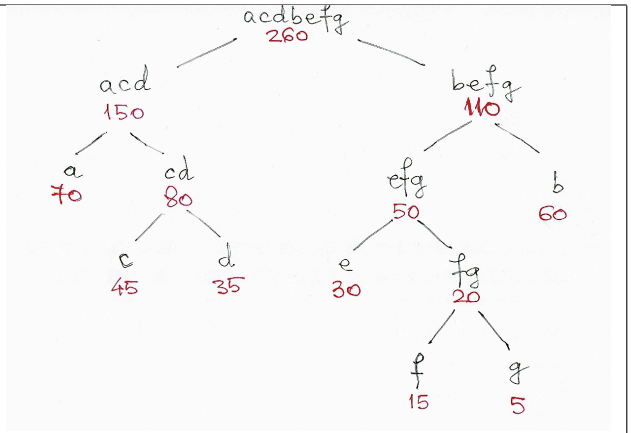




lettre	nb. occurences
befg	110
cd	80
a	70

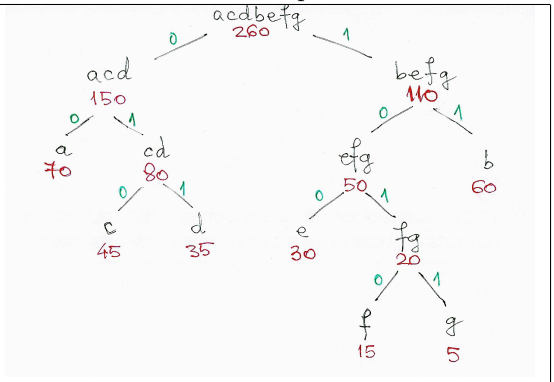


lettre	nb. occurences
acd	150
befg	110



lettre	nb. occurences
acdbefg	260

Finalement, dans l'arbre (de Huffman) ainsi obtenu, on étiquette les branches à gauche par 0 et celles à droite par 1. Si on parcourt les branches entre la racine et une lettre, et on collecte les étiquettes, on obtient le code de chaque lettre



lettre	nb. occurences	codage
a	70	00
b	60	11
c	45	010
d	35	011
e	30	100
f	15	1010
g	5	1011

Remarques

• L'arbre de Huffman (et donc le codage) obtenu n'est pas nécessairement unique
• le codage obtenu est un codage prefixe, de longueur variable
• dans tous les cas, le codage obtenu est optimal, c.à.d, pour une répartition des probabilités donnée des lettres, il minimise la longueur moyenne

TD 2-3 Compression de données

Exo 3 (Move-To-Front)

Remarques

MTF est un algorithme de pré-compression : il transforme un type de redondance (répétition des symboles par plages) par un autre type de redondance (apparition plus fréquente d'un symbole particulier, A ou 0). Ce dernier type de redondance permet la compresion avec le codage δ de Elias. Voir l'exo 4

Exo 5

La taille du tampon de recherche et du tampon de lecture est 3. Pour amorcer l'algorithme on ajoute en tête 3 espaces (_ _ _).
• Il faut appliquer l'algo LZ77 à la donnée

$$\underbrace{\_\_\_}_{t_1}\underbrace{aab}_{t_2}aacacb$$

où t_1 est le tampon de recherche et t_2 est le tampon de lecture.

		lexème	
$\underbrace{\_\_\_}\underbrace{\mathbf{a}ab}\ aacacb$	il n'y a pas de facteur en t_1 qui soit à la fois un prefixe de t_2	$(0,0,\mathbf{a})$	
$\_\underbrace{\_\_a}\underbrace{a\mathbf{b}a}\ acacb$	le plus long facteur en t_1 qui est à la fois un prefixe de t_2 est a. a sera representé comme $(1,1) = (decalage, longueur)$	$(1,1,\mathbf{b})$	
$\_\_\_\ \underbrace{aab}\ \underbrace{aa\mathbf{c}}\ acb$	le plus long facteur en t_1 qui est à la fois un prefixe de t_2 est aa. aa sera representé comme $(3,2) = (decalage, longueur)$	$(3,2,\mathbf{c})$	
$\_\_\_aab\ \underbrace{aac}\ \underbrace{ac\mathbf{b}}$	le plus long facteur en t_1 qui est à la fois un prefixe de t_2 est ac. ac sera representé comme $(2,2) = (decalage, longueur)$	$(2,2,\mathbf{b})$	

Finalement, la version compressée de $aabaacacb$ est
$(0,0,a)\ (1,1,b)\ (3,2,c)\ (2,2,b)$

Avec des textes plus longs, et des tampons plus grands, l'algo peut donner un bon taux de compression.

• Il faut appliquer l'algo LZ77 à la donnée $ababcacbacc$.

$\underbrace{\_\_\_}\underbrace{aba}\ bcacbacc$	$(0,0,a)$
$\_\underbrace{\_\_a}\ \underbrace{bab}\ cacbacc$	$(0,0,b)$
$\_\ \underbrace{\_\_ab}\ \underbrace{abc}\ acbacc$	$(2,2,c)$
$\_\_\_ab\ \underbrace{abc}\ \underbrace{acb}\ acc$	$(3,1,c)$
$\_\_\_abab\ \underbrace{cac}\ \underbrace{bac}\ c$	$(0,0,b)$
$\_\_\_ababc\ \underbrace{acb}\ \underbrace{acc}$	$(3,2,c)$

Finalement, la version compressée de $ababcacbacc$ est
$(0,0,a)\ (0,0,b)\ (2,2,c)\ (3,1,c)\ (0,0,b)\ (3,2,c)$

TD4-5 Codes correcteurs d'erreurs

L'intérêt des exo

1. Un code simple, qui n'est pas linéaire

2. Deux codes linéaires simples et 'naturels'
3. Un code défini à partir de sa matrice génératrice. On retrouve ce code lors de l'exo 5 et 7
4. Un autre code défini à partir de sa matrice génératrice
5. Décodage (laborieux) avec le tableau standard
6. Familiarisation à la matrice de contrôle
7. Décodage par syndrome, plus efficace mais plus technique

8,9. Codes conçus pour accélérer le décodage.

Exo 1

C'est un correcteur d'erreurs ad-hoc, naïf et inefficace. C'est just pour sensibiliser aux notions utilisées. Dans le hypercube, les sommets marqués pas • sont des code valides (0100 pour a, 0011 pour b, 1000 pour c, et 1111 pour d), les autres résultent après une ou plusieurs erreurs. On aurait pu simplement avoir $a \mapsto 01$, $b \mapsto 00$, $c \mapsto 10$, et $d \mapsto 11$, mais dans ce cas il n'y a pas de redondance, donc pas de détection/correction d'erreurs.

Correction

Si on a reçu 0111, le plus proche voisin qui est un mot du code (=mot valid) sont en fait deux : 0011 et 1111, chacun à distance de Hamming =1. Donc, on detecte une erreur mais on ne peut pas la corriger.
Si on a reçu 0110, le plus proche voisin qui est un mot du code est unique : 0100. Donc, on detecte et on corrige une erreur.
Si on a reçu 0011, il est un mot du code, on conclut qu'il n'y a pas d'erreurs.

Pour la distance minimale, on calcule la distance de Hamming entre deux mots du code

	0100	0011	1000	1111
0100	-	3	2	3
0011		-	3	2
1000			-	3
1111				-

et la distance minimale est 2
Le taux d'information d'un code C, dans notre cas $C = \{0100, 0011, 1000, 1111\}$, est la rapport

$$R = \frac{\log_2 card(C)}{n}$$

où n est la longueur des mots du code. Donc $R = \frac{\log_2 4}{4} = 0,5$. Ce qui est assez intuitif, on envoie 0100 au lieu de 01 par exemple.

Soit un code avec la distance minimale d. Alors, le code

- détecte jusqu'à $d-1$ erreurs
- corrige jusqu'à $\lfloor \frac{d-1}{2} \rfloor$ erreurs.

Dans notre cas $d = 2$, le code détecte 1 erreurs, et corrige 0 (zéro) érreurs (dans des cas particuluers, le code peut corriger 1 erreur, c'est le cas de 0110).

Exo 2

L'énoncé necessite des précisions. L'intérêt de l'exo est de se familiariser avec les codes linéaires et d'illustre que des codes très simples sont en fait des codes linéaires.

Correction

(i)
Un exemple de code avec bit de parité est

$$\begin{array}{lcl} 000 & \mapsto & 0000 \\ 001 & \mapsto & 0011 \\ 010 & \mapsto & 0101 \\ 011 & \mapsto & 0110 \\ 100 & \mapsto & 1001 \\ 101 & \mapsto & 1010 \\ 110 & \mapsto & 1100 \\ 111 & \mapsto & 1111 \end{array}$$

et les mots du code sont $\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$

En multipliant un mot binaire de longueur 3 avec la matrice

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

on obtient son code. Toutes les operations sont effectuées modulo 2 ! Par exemple $101 \cdot G = 1010$. Il résulte que c'est un code linéaire (ce qui n'est pas le cas du code de l'exo. 1).
La dimension du code est 3 (=au nombre de lignes de la matrice génératrice= la longueur des mots binaires avant le codage).
La taille du code est 4 (=au nombre de colonnes de la matrice génératrice= la longueur des mots binaires après le codage).
Pour les codes linéaires, la distance minimale = le poids minimale. On calcule donc le poids (nombre de 1) du chaque mot du code, excepté 00...0.

mot du code	poids
0011	2
0101	2
0110	2
1001	2
1010	2
1100	2
1111	4

et donc le code à le poids minimale =2 =distance minimale.
Les paramètres du code sont :

$$\text{(taille, dimension, distance mini.)=(4,3,2).}$$

Pour les codes linéaire le taux d'information $=\frac{dimension}{taille}$, et dans notre cas $=\frac{3}{4}$.
Le code détecte $d-1=1$ erreurs (d=distance mini.=2), et corrige $\lfloor\frac{d-1}{2}\rfloor=0$ erreurs.

(ii)
Un exemple de code de deux répétitions d'une 'plage de trois' est

$000 \mapsto 000\,000$
$001 \mapsto 001\,001$
$010 \mapsto 010\,010$
$011 \mapsto 011\,011$
$100 \mapsto 100\,100$
$101 \mapsto 101\,101$
$110 \mapsto 110\,110$
$111 \mapsto 111\,111$

Sa matrice génératrice est

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

Le taux d'information est $0,5$. Le code détecte une erreur et corrige 0 (zéro) erreurs.

Exo 3

x (mot en clair)		$x\cdot G$ (mots du code)	poids
000	$\mapsto$	000 000	
001	$\mapsto$	001 110	3
010	$\mapsto$	010 101	3
011	$\mapsto$	011 011	4
100	$\mapsto$	100 011	3
101	$\mapsto$	101 101	4
110	$\mapsto$	110 110	4
111	$\mapsto$	111 000	3

7. Pour comparer ce code avec celui de l'exo 2 point (ii) on voit que les deux codes on un taux d'information de 0,5, mais le code de l'exo 3 a le
poids mini=3=distance mini, donc ce code

- detecte $d-1=2$ erreurs et
- corrige $\lfloor \frac{d-1}{2} \rfloor = 1$ erreur

Le code de l'exo 2 point (ii) détectait 1 erreur et corrigeait 0 (zéro) erreurs, donc ce dernier code est bien plus performant.
8. Soit C l'ensemble de mots valides d'un code linéaire de taille n qui corrige jusqu'à e erreurs. Alors la proposition de Hamming est :

$$card(C) \leq \frac{2^n}{\sum_{i=0}^{e} C_n^i}.$$

Elle dit aproximativement que, pour une taille donnée, plus on veut corriger d'erreurs (e est grand) moins il y aura de mots du code ($card(C)$ est petit). Autrement dit, si on veut augmenter le nombre d'erreurs corrigées, on est obligé de réduire le taux d'information. Dans notre cas

$$card(C) \leq \frac{2^n}{\sum_{i=0}^{e} C_n^i} \iff 8 \leq \frac{2^6}{\sum_{i=0}^{1} C_6^i} \iff 8 \leq \frac{64}{7}$$

Exo 4

En reformulant l'énoncé, le code est

$$a_1a_2a_3a_4 \mapsto a_1a_2a_3a_4\, c_1c_2c_3$$

avec :

$$\begin{aligned} c_1 &= a_1 + a_2 + a_3 \\ c_2 &= a_1 + a_2 + a_4 \\ c_3 &= a_2 + a_3 + a_4 \end{aligned}$$

l'addition est effectuée modulo 2.
Donc la matrice génératrice du code est :

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

000000	001110	010101	011011	100011	101101	110110	111000
100000	101110	110101	**111011**	000011	001101	010110	011000
010000	011110	000101	001011	**110011**	111101	100110	101000
001000	000110	011101	010011	101011	100101	111110	110000
000100	001010	010001	011111	100111	**101001**	110010	111100
000010	001100	010111	011001	100001	101111	110100	111010
000001	001111	010100	011010	100010	101100	110111	111001
010010	011100	000111	001001	110001	111111	100100	101010

Table 1: Tableau standard exo 5

Exo 5

Le code linéaire définie en exo. 3 a la taille =6.
Le tableau standard content tous les $2^{taille} = 2^6 = 64$ mots binaires de longueur 6. Ce tableau est construit ligne par ligne.
La première ligne est constituée de 8 mots du code valides.
La ligne i du tableau standard ($i > 1$) est obtenue de la manère suivante :

- le premier élément de la ligne i est un mot binaire de longueur 6, de poids minimale, qui n'existe pas dans les lignes précédentes. Dans notre cas pour la ligne 2 on a choisit 100000 mais on aurait pu choisir 010000, 001000 ...
- Un élément de de la ligne i (autre que le premier) est obtenu en faisant la somme bit-à-bit, modulo 2, du : premièr élément de sa collone et premièr élément de sa ligne. Par exemple **101001**=101101+000100

Pour la dernière ligne on choisit 010010, le choix n'est pas unique, on aurait pu choisir 001101 ou 100100.

Pour les codes linéaire le tableau ainsi obtenu contient une fois tous les mots binaires de longueur = taille.

Les classes latérales sont les lignes du tableau standard. Les chefs de classes sont les premiers mots des classes latérales, c.à.d., la première colonne du tableau.

Correction Si on a reçu un mot binaire x, il sera décodé comme le premier mot de sa colonne et avec comme vecteur d'erreur le premier mot de sa ligne.

- 111011 est décodé comme 011011 avec comme vecteur d'erreur 100000. C.à.d. si on a reçu 111011, alors 011011 a été envoyé et il y a eu une erreur en 1ère position
- 110011 est décodé comme 100011 avec comme vecteur d'erreur 010000.

Remarque

On peut aussi demander de décoder

- 110110. Ce mot est sur la première ligne du tableau standard, donc le vecteur d'erreur est 000000, c.à.d, il n'y a pas d'erreur

- 000111. Ce mot est sur la dernière ligne, donc il y a plus d'une erreur. Le code (voir l'exo 3) corrige une erreur et détecte 2. Donc pour 000111 on détécte 2 erreurs que on ne peut pas les corriger.

Exo 6

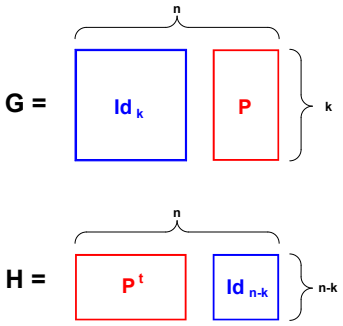
C'est un exo qui introduit la notion de matrice de contrôle, utilisée lors de l'exo 7.

> Si G est la matrice génératrice d'un code correcteur d'erreurs linéaire, et les premières colonnes de G forment la matrice identité (unité), alors on dit que G est sous forme normale.
> A partir d'une matrice génératrice sous forme normale, on peut obtenir une matrice H, appelé matrice de contrôle.

Voir la construction de la matrice de contrôle H à partir de la matrice génératrice G dans la figure suivante. Dans cette figure Id_k est la matrice identité de taille k, P^t est la transposée de P. (Dans mon cours, j'ai noté par ${}^tP$ la transposée de P)

La matrice de contrôle H est

- à son tour la matrice génératrice d'un code correcteur d'erreurs (le code dual)
- utilisée pour le décodage efficace (voir l'exo suivant)



Maintenant soit la matrice génératice de notre exo

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} Id_2 & P \end{pmatrix}$$

On peut remarquer en parenthese que G engendre le code avec bit de parité

$$\begin{array}{ccc} 00 & \mapsto & 000 \\ 01 & \mapsto & 011 \\ 10 & \mapsto & 101 \\ 11 & \mapsto & 110 \end{array} \tag{1}$$

La matrice de contrôle H est

$$H = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

(c'est un cas dégénéré) et H engendre le code

$$\begin{array}{ccc} 0 & \mapsto & 000 \\ 1 & \mapsto & 111 \end{array} \tag{2}$$

qui est le code de 'répétition de longueur 3'. Le code dual du code défini en (1) est le code défini en (2).

Exo 7

1. La matrice génératrice est (la même que dans l'exo 3 et 5)

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

et on a la matrice de contrôl

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

2. Les chefs de classes on déjà été calculés lors de l'exo 5 : sont les mots binaires de la première colonne du tableau standard :

$$0000000, 100000, 010000, 001000, 000100, 000010, 000001, 010010$$

(le choix de 010010 est un peu arbitraire, mais cela ne change pas le résultat).

> Pour un code linéaire de taille n, le syndrome d'un mot binaire x de longueur n, noté $S(x)$, est $x \cdot H^t$, où H^t est la transposée de la matrice de contrôl.

Dans notre cas

$$H^t = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

et on a

chef de classe x	$S(x) = x \cdot H^t$ =syndrome de x
000000	000
100000	011
010000	101
001000	110
000100	100
000010	010
000001	001
010010	111

Les syndromes des chefs de classes doivent être tous differents (c'est bien le cas)

3.

Soit f la fonction qui associe à un syndrome y le chef de classe x tel que le syndrome de x est y. Pour décoder x

- on calcule $S(x)$, le syndrome de x
- le vecteur d'erreur de x est $f(S(x))$

syndrome y	chef de classe $f(y)$
000	000000
011	100000
101	010000
110	001000
100	000100
010	000010
001	000001
111	010010

Comme pour l'exo 5 on va décoder 111011, 110011, 110110, 000111.

Décoder $x = 111011$.

- $S(x) = x \cdot H^t = 011$
- le vecteur d'erreur de x est $f(011) = 100000$.
  $x = \underline{1}11011$ est decodé comme $\underline{0}11011$

Décoder $x = 110011$.

- $S(x) = 101$
- le vecteur d'erreur est $f(101) = 010000$.
  $x = 1\underline{1}0011$ est decodé comme $1\underline{0}0011$

Décoder $x = 110110$.

- $S(x) = 000$
- le vecteur d'erreur est $f(000) = 000000$.
  Il n'y a pas d'erreurs.

Décoder $x = 000111$.

- $S(x) = 111$
- on obtient pour le vecteur d'erreur $f(111) = 010010$.
  On détécte 2 erreurs que on ne peut pas corriger de manière certaine

remarque Le décodage par syndrome est plus technique mais plus efficace que celui lors de l'exo 5, on n'a pas besoin du tableau standard, recherche dans le tableau standard ...

Exo 8

1. Le code détecteur et correcteur d'erreurs $Ham(r)$ est défini à partire de sa matrice de contrôle : les colonnes sont les vecteurs qui représentent les entiers de 1 à $2^r - 1$. Dans notre cas :

$$H = Ham(3) = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

2.

Les codes $Ham(r)$ ont une distance maxi égale à 3, donc ils détectent et corrigent une erreur. Pour les codes $Ham(r)$, le syndrome de y, lu comme un entier en base 2, donne la position de l'erreur. Cela accélère le décodage.

Le syndrome de $y = 1101011$ est

$$S(y) = y \cdot H^t = 110$$

Maitenant, si on lit 110 comme un entier en base 2, on a

$$S(y) = 6$$

ce qui veut dire que dans y il y a une erreur en 6ème position, donc y est corrigé comme

$$1101001$$

Remarque : les étudiants devraient savoir faire le point 2. de cet exercice. On le trouve comme exemple à la fin du cours sur les codes correcteurs d'erreurs

3. Pour les codes linéaires la relation entre la matrice génératrice G et la matrice de contrôle H est

$$G \cdot H^t = [0]$$

où $[0]$ dénote la matrice de taille convenable avec tous les éléments égaux à zéro (c'est la magie de l'algèbre linéaire).
Maintenant si on met

$$H = Ham(3) = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

sous forme normale, en permutant les colonnes de même couleur on obtient

$$H' = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

et si on regarde H' comme une matrice de contrôle, alors la matrice génératrice correspondante est

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Finalement, en faisant dans G' les même permutations des colonnes que dans H, on obtient la matrice génératrice qui correspond à la matrice de contrôle de $Ham(3)$

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

Il est facile à verifier que

$$G \cdot H^t = [0]$$

(parce que $G' \cdot (H')^t = [0]$)

Exo 9

Remarque : La matrice $\overline{H}$ est obtenue à partir de la matrice $Ham(3)$ en ajoutant une (dérnière) ligne et colonne.

TD6 Chiffrement à clé secrète

remarque C'est un TD qui ne me prend jamais plus d'une heure et demie. J'enchaine avec le TD suivant.

Exo 1

Selon Shannon, la *diffusion* est une propriété où la redondance dans les textes en clair est dissipée dans les textes chiffrés. En d'autres termes, un biais en entrée ne doit pas se retrouver en sortie. Des similarités entre les valeurs en entrée et en sortie pourraient être utiles pour le cryptanalyste.
Ce concept est lié à la notion plus moderne d'*effet avalanche*. Dans un chiffrement avec une bonne diffusion, le changement d'un seul valeur en entrée doit changer chaque valeur en sortie.
Les fonctions Γ, Δ et Λ ci-dessous illustrent la notion de *diffusion* ou *effet avalanche*. Elles ne sont pas des fonctions de chiffrement, car trop simples. Elles peuvent être utiliser comme un pré-traitement avant le chiffremet.

•
$\Gamma(363000) = 314444$
$\Delta(314444) = 363000$
remarque Γ est inversible, et son inverse est Δ, et réciproquement

$\Gamma(353000) = 303333$
remarque $3\underline{6}3000$ et $3\underline{5}3000$ different sur une seule position (la 2ème) mais leurs images par Γ different partout à partire de la 2ème position

•
$\Gamma(\Delta(163401)) = 163401$
remarque Γ et l'inverse de Δ (mais on l'avait déjà dit ...)
•
$\Gamma(\Lambda(250613) = 102403$, et
$\Gamma(\Lambda(251613) = 225736$
remarque 250613 et 251613 different d'une unité, en position 3, et les résultats different partout. Il est important de noter que Γ et Λ et donc $\Gamma \circ \Lambda$ sont inversibles

Exo 2

1. Les premières valeurs de k_n sont :

n	1	2	3	4	5	6	7	8	9	10
k_n	0	1	1	1	0	1	0	0	1	1

et la période est 7.

Moralité Il est difficile de générer des suites aléatoires avec des méthodes déterministes (algorithmes, programmes informatiques ...)

Exo 3

Soit la suite préudo-aléatoire obtenue dans l'exo. 2. pt 1 : $0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,\ldots$.
On va dégager 3 clés de longueur la moitié de la longueur du message m :

- $\underline{0\,1\,1}\,1\,0\,1\,0\,0\,1\,1\,\ldots \rightarrow k' = 0\,1\,1$
- $0\,1\,1\,1\,\underline{0\,1}\,0\,0\,1\,1\,\ldots \rightarrow k'' = 1\,0\,1$

- $0\,1\,1\,1\,0\,1\,\underline{0\,0\,1}\,1\,\ldots \rightarrow k''' = 0\,0\,1$

Chiffrement du message $m = 101\,011$. On note se message par C_0, on le coupe en deux, et on a

$$C_0 = (C_0^L, C_0^R) = 101\,011$$

On va appliqué la méthode de Feistel 3 fois (3 étapes) pour obternir C_1, C_2 et C_3, à partir de $m = C_0$. C_3 sera la version chiffrée de $m = C_0$
Avec la méthode de Feistel (c'est une version légèrement simplifiée) on a (avec l'addition bit-à-bit, modulo 2)

$$\begin{aligned} C_1 &= (C_0^R, f_1(C_0^R + k') + C_0^L) \\ &= (011, f_1(011 + 011) + 101) \\ &= (011, f_1(000) + 101) \\ &= (011, 011 + 101) \\ &= (011, 110) \\ &= (C_1^L, C_1^R) \end{aligned}$$

De manière similaire on calcule C_2.

$$\begin{aligned} C_2 &= (C_1^R, f_1(C_1^R + k'') + C_1^L) \\ &= (110, f_1(110 + 101) + 011) \\ &= (110, f_1(011) + 011) \\ &= (110, 000 + 011) \\ &= (110, 011) \\ &= (C_2^L, C_2^R) \end{aligned}$$

Et

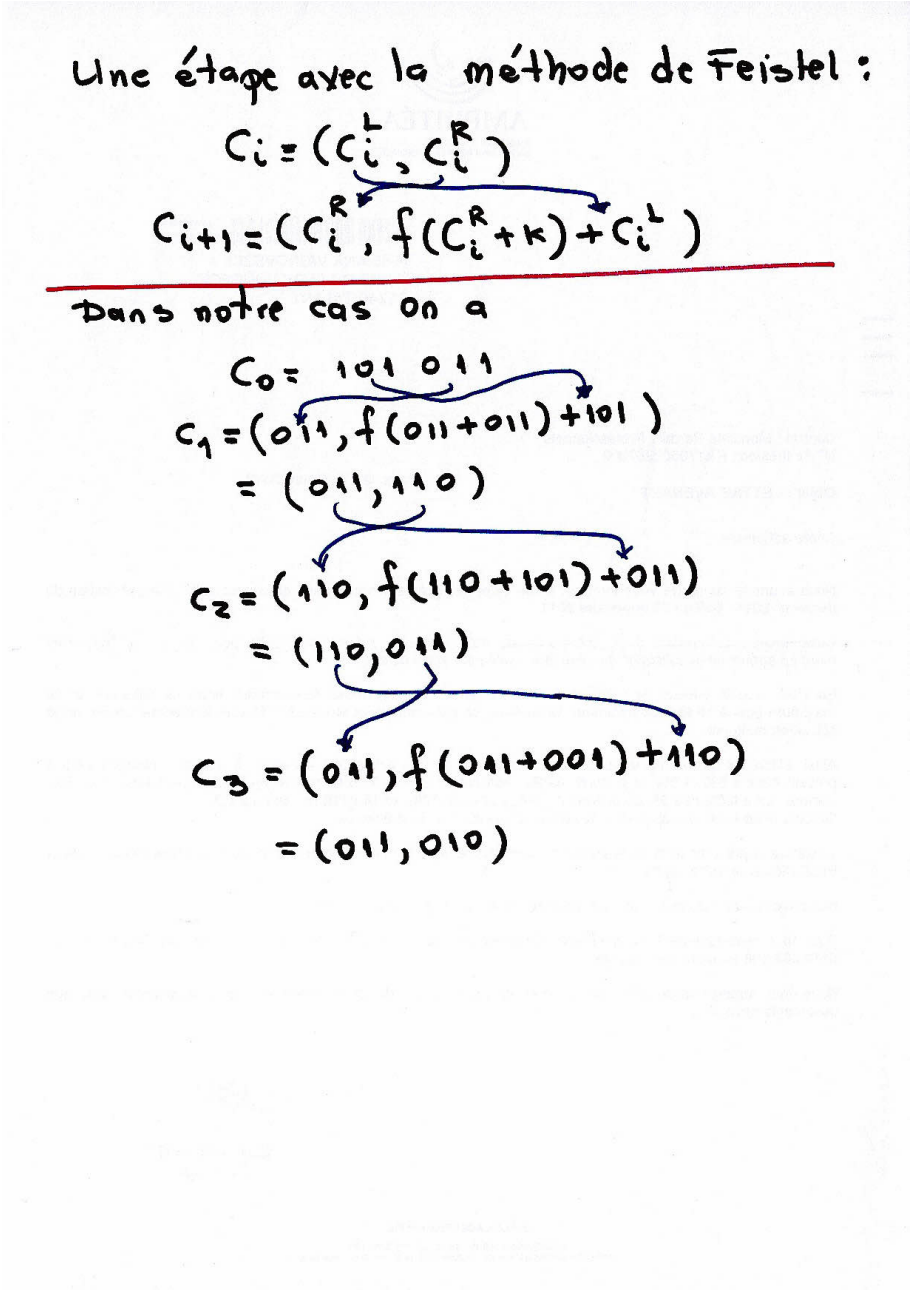
$$\begin{aligned} C_3 &= (C_2^R, f_1(C_2^R + k''') + C_2^L) \\ &= (011, f_1(011 + 001) + 110) \\ &= (011, f_1(010) + 110) \\ &= (011, 100 + 110) \\ &= (011, 010) \end{aligned}$$

Donc m chiffré avec la méthode de Feistel à trois etapes est 011010.

On peut refaire avec le deuxième message $m = 101001$ et constater que le résultat et très different, malgré le fait que les deux massages 101011 et 101001 diffèrent sur une seule position.

La méthode de Feistel est utilisée dans des algorithmes de chiffrement à clé secrète par bloc comme DES (Data Encryption Standard)

Le diagramme pour $m = 01011$ est sur la page suivante :



TD 7-9 Chiffrement à clé publique

Remarque J'utilise sans distinction la notation $\mathbb{F}_p$ et $\mathbb{Z}/p\mathbb{Z}$

Exo 1

Un algorithme naïf calcule g^x avec

$$g^x = \underbrace{g \cdot g \cdot g \cdot \dots \cdot g}_{x}$$

ce qui demande $x - 1$ multiplications.

Il existe des algorithmes bien plus rapides, voir `http://v.vincent.u-bourgogne.fr/OENS/THEO-INFO/exp_rapide.txt`.
Ces algorithmes demandent $\lceil \log_2 x \rceil$ iterations avec une ou deux multiplications par itération. $\lceil \log_2 x \rceil$ est la longueur de l'écriture de x en binaire. En conlusion, les algorithmes rapides demandent au plus $2 \cdot \lceil \log_2 x \rceil$ multiplications.
Exemple

- $g^{25} = \underbrace{g \cdot g \cdot g \cdot \dots \cdot g}_{25}$, avec 24 multiplications
- $g^{25} = g \cdot g^8 \cdot g^{16} = g \cdot (((g \cdot g^2)^2)^2)^2$ avec 6 multiplications

Exemple Si x est un entier de 100 chiffres en base 10 ($x \sim 10^{100}$), alors

- l'algorithme naïf demande $(x - 1) \sim 10^{100}$ multiplications et
- l'algorithme rapide demande au plus $2 \cdot \lceil \log_2 10^{100} \rceil = 664 \leq 10^3$ multiplications

Avec (par exemple) 1000 multiplications par seconde

- l'algorithme naïf demande 10^{89} années ! (l'âge estimé de l'univers observable est de $1,38 \cdot 10^{10}$ d'années), et
- l'algorithme rapide demande moin de 1 seconde

Moralité On peut calculer efficacement (dans un temps raisonnable) la fonction

$$x \mapsto g^x,$$

mais on ne sait pas calculer efficacement son inverse

$$y \mapsto \log_g y$$

Même remarque si les calcules sont effectués mod n
C'est précisément la difficulté de calculer $x \mapsto \log_g x$ (en $\mathbb{N}$ ou $\mathbb{F}_p^*$) qui rends le protocole de Diffie-Hellman sûr (dans l'état actuel de nos connaissances)

Exo 11

remarque Cet exo est censé de souligner que le protocole de Shamir et les codes détecteurs d'erreurs sont basés sur la redondance.

Correction

Soit

$$m = m_1 m_2 m_3 m_4 m_5 m_6 \cdots$$

un message, et sans perte de généralité supposons que m est écrit avec des symboles en $\mathbb{Z}/5\mathbb{Z}$ (c.à.d., $m_1, m_2, m_3, m_4, m_5, m_6, \cdots \in \mathbb{Z}/5\mathbb{Z}$).
Maintenant, on découpe le message, et on ajoute de la redondance :

$$m' = m_1 m_2 m_3 f(0)\ m_4 m_5 m_6 g(0)\ \cdots$$

où

- f est l'unique polynôme : $\mathbb{Z}/5\mathbb{Z} \to \mathbb{Z}/5\mathbb{Z}$ avec $f(1) = m_1$, $f(2) = m_2$, $f(3) = m_3$
- g est l'unique polynôme : $\mathbb{Z}/5\mathbb{Z} \to \mathbb{Z}/5\mathbb{Z}$ avec $g(1) = m_4$, $g(2) = m_5$, $g(3) = m_6$
- $\cdots$

Si une erreur c'est produite en $m_1 m_2 m_3 f(0)$, elle peut être détectée. En effet, supposons par exemple que on a reçu

$$m_1 m_2' m_3 f(0)$$

au lieu de

$$m_1 m_2 m_3 f(0)$$

Dans ce cas, si f' est le polynôme d'interpolation déterminé par $m_1 m_2' m_3$, alors $f'(0) \neq f(0)$, et l'erreur est détectée
Pareil pour $m_4 m_5 m_6 g(0)$, $\cdots$