THEORIE DE L'INFORMATION

CORRIGES

TD 1 Corrigé. Modélisation mathématique d'une source d'information

Partie I. Entropie d'une source. Définitions et propriétés.

Exercice 1. Entropie d'une source.

On utilise la définition : $H(X) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$ en bits/symbole (n est la taille de l'alphabet). Ici n = 5.

1. $H(X) = -0.1 \cdot \log_2(0.1) - 0.2 \cdot \log_2(0.2) - 0.3 \cdot \log_2(0.3) - 0.15 \cdot \log_2(0.15) - 0.25 \cdot \log_2(0.25) \approx 2.23$ bits/symbole

2. $H(X) = -4 \cdot 0.05 \cdot \log_2(0.05) - 0.8 \cdot \log_2(0.8) \approx 1.12$ bits/symbole

3. Par la définition : $H(X) = -5 \cdot 0.2 \cdot \log_2(0.2) = -\log_2\left(\frac{1}{5}\right)$ bits/symbole , ou plus directement par la relation :

$H(X) = \log_2(n) = \log_2(5) \approx 2.32$ bits/symbole .

L'entropie représente l'incertitude sur l'issue de l'expérience; c'est l'information moyenne apportée par l'expérience.

On remarquera que l'entropie est la plus élevée dans le cas **3.** : c'est une propriété générale de l'entropie :

L'entropie atteint son maximum lorsque tous les symboles d'un alphabet donné de taille n *sont équiprobables* (l'incertitude sur la prédiction de détermination d'un symbole est maximale). *Elle vaut alors* $\log_2(n)$.

Exercice 2. Information moyenne.

1. L'alphabet ici ne contient que le symbole « pile ». On remarque que « *pile* » est un évènement certain (probabilité 1). Il n'y a aucune incertitude sur l'issue de l'expérience. On a donc $H(X) = 0$ (rappel : $\log_2(1) = 0$)

2. Soit $Y = \begin{cases} 1 & \text{si le chiffre sur le dé (1 à 6) est pair} \\ 0 & \text{si le chiffre sur le dé (1 à 6) est impair} \end{cases}$. L'alphabet est de taille 2.

$P(Y=1) = P(Y=0) = 0.5$.

D'après la propriété de l'entropie dans le cas d'une distribution uniforme, on a immédiatement :

$H(Y) = \log_2(2) = 1$ bit/symbole

3. Soit $Y \in \{\clubsuit, \blacklozenge, \heartsuit, \spadesuit\}$. L'alphabet est de taille 4. On calcule la distribution de probabilités associée :

Y	♠	♣	♥	♦
$P(Y)$	3/10	4/10	2/10	1/10

On en déduit l'entropie :

$$H(Y) = -\frac{3}{10} \cdot \log_2\left(\frac{3}{10}\right) - \frac{4}{10} \cdot \log_2\left(\frac{4}{10}\right) - \frac{2}{10} \cdot \log_2\left(\frac{2}{10}\right) - \frac{1}{10} \cdot \log_2\left(\frac{1}{10}\right) \approx 1.85 \text{ bits/symbole}$$

Exercice 3. Propriété de groupe.

1. $H(X) = -0.1 \cdot \log_2(0.1) - 0.2 \cdot \log_2(0.2) - 0.3 \cdot \log_2(0.3) - 0.15 \cdot \log_2(0.15) - 0.25 \cdot \log_2(0.25) \approx 2.23$ bits/symbole

2.

$p = P(A) = 0.1 + 0.3 + 0.25 = 0.65$

$q = P(B) = 1 - p = 0.2 + 0.15 = 0.35$

$H(p,q) = -p\log_2(p) - q\log_2(q) \approx 0.93$ bit/symbole

3. Les distributions de probabilité de A et B sont : **(on normalise pour que la somme des probabilités soit égale à 1)**

x	**1**	**3**	**5**
$P_A(x)$	0.1/0.65 = 10/65	0.3/0.65 = 30/65	0.25/0.65 = 25/65

x	**2**	**4**
$P_B(x)$	0.2/0.35 = 20/35	0.15/0.35 = 15/35

$H_A = -\frac{10}{65}\log_2\left(\frac{10}{65}\right) - \frac{30}{65}\log_2\left(\frac{30}{65}\right) - \frac{25}{65}\log_2\left(\frac{25}{65}\right) \approx 1.46$ bits/symbole $\qquad H_B = -\frac{20}{35}\log_2\left(\frac{20}{35}\right) - \frac{15}{35}\log_2\left(\frac{15}{35}\right) \approx 0.98$ bit/symbole

Vérification de la propriété de groupe : $H(X) = H(p,q) + pH_A + qH_B$:

$H(X) = H(p,q) + pH_A + qH_B = 0.93 + (0.65 \cdot 1.46) + (0.35 \cdot 0.98) = 0.93 + 0.95 + 0.34 \approx 2.23$ bits/symbole

Exercice 4. Vers le codage de Shannon.

1. $H(X) = -0.1 \cdot \log_2(0.1) - 0.2 \cdot \log_2(0.2) - 0.3 \cdot \log_2(0.3) - 0.15 \cdot \log_2(0.15) - 0.25 \cdot \log_2(0.25) \approx 2.23$ bits/symbole

2. *Propriété de groupe :* $H(X) = H(p,q) + pH_A + qH_B$ avec $q = 1 - p$

Interprétation : $H(X)$ est composée de :

1. l'incertitude sur le choix de l'une des 2 parties A et B : c'est $H(p,q)$

2. la moyenne des incertitudes associées à chacune des 2 parties séparément : c'est $pH_A + qH_B$

Ici :

$H(p,q)$ représente la quantité moyenne d'information apportée par l'observation d'une source binaire.

La source est la réponse à une question et p, la probabilité de la réponse « oui ».

$\rightarrow H(p,q) = -p\log_2(p) - q\log_2(q)$

3. D'après les propriétés de l'entropie : dans le cas d'un alphabet de taille n, l'entropie atteint son maximum (égal à $\log_2(n)$) uniquement dans le cas de distribution de probabilités uniforme. Dans le cas d'un canal binaire ($n = 2$) et une distribution de probabilités uniforme ($p = q = 0.5$), on a donc : $\max_{p \in [0,1]} H(p,q) = \log_2(2) = 1$ bit/symbole.

4. *Propriété de groupe :* La réponse à une question binaire partage l'ensemble des valeurs possibles de X en 2 parties disjointes : celle du « oui » et celle du « non ». On peut alors appliquer la propriété de groupe :

Propriété de groupe : $H(X) = H(p,q) + pH_A + qH_B = H(p,q) + pH_{oui} + qH_{non}$ $\qquad \begin{cases} H_A = H_{oui} \\ H_B = H_{non} \\ q = 1 - p \end{cases}$

Interprétation : $H(X)$ est composée de :

1. l'incertitude sur le choix de l'une des 2 parties A et B : c'est $H(p,q)$

2. la moyenne des incertitudes associées à chacune des 2 parties séparément : c'est $pH_A + qH_B$

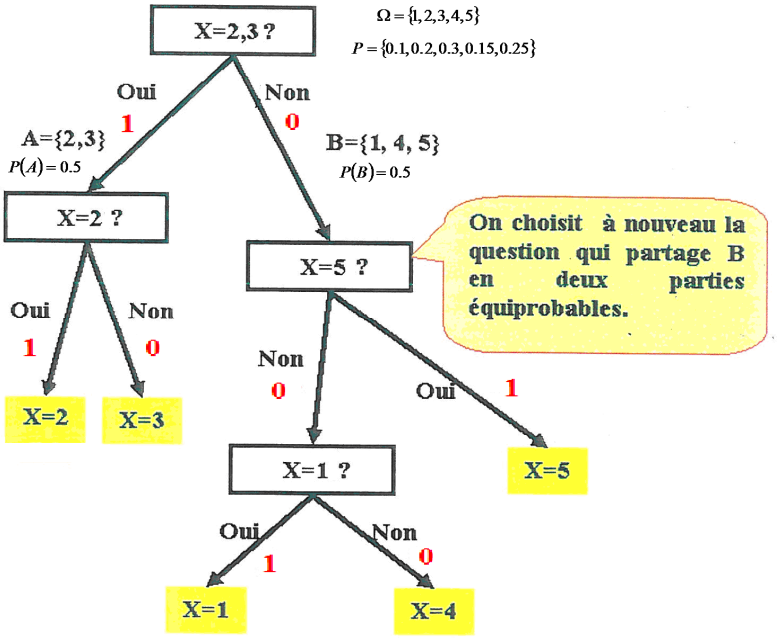
Si $H(X)$ est l'incertitude moyenne que l'on a sur le résultat du tirage de X en début de jeu, la quantité $H(p,q)$ représente l'information moyenne apportée par la réponse à la 1ère question, et la quantité $pH_{oui} + qH_{non}$ est l'incertitude moyenne restant sur la valeur de X une fois la réponse obtenue.

On a alors intérêt à choisir la *1ère question* de sorte qu'elle apporte le *maximum d'information*.

→La question à poser doit partager l'ensemble des valeurs de X en 2 *parties équiprobables* (à approcher autant que possible).

Ici, c'est possible en posant la question : $X \in A = \{2,3\}$? car $p(X \in A) = p(X=2) + p(X=3) = 0.2 + 0.3 = 0.5$

5.



6. A la lecture de l'arbre, on obtient la *table de code* :

X	**1**	**2**	**3**	**4**	**5**
m_i (mot-code)	001	11	10	000	01
l_i (longueur de mot-code)	3	2	2	3	2

Remarque : on verra au chapitre 3 (Codage de source) avec *l'inégalité de Kraft*, qu'un code *sans préfixe* existe car $\left.\sum_{i=1}^{n} d^{-l_i} = 2(2)^{-3} + 3(2)^{-2} = 1 \cdot \right| d=2;\ n=5$

L'inégalité de Kraft indique qu'un code *sans préfixe* de longueurs de mots-données $(l_1, l_2, \ldots, l_n)$ existe si et seulement si $\sum_{i=1}^{n} d^{-l_i} \le 1$ où d est la taille de l'alphabet du *canal* (d = 2 pour un canal *binaire* comme ici) et n, la taille de l'alphabet de la *source*.

7. A la lecture de l'arbre : (la méthode utilisée ici est la même que celle du codage de Shannon, qui sera étudiée plus tard)

X	**1**	**2**	**3**	**4**	**5**
P_X (probabilité)	0.1	0.2	0.3	0.15	0.25
N (nombre de questions)	3	2	2	3	2

Nombre moyen de questions (moyenne de N) :

$$E[N] = (3 \cdot 0.1) + (2 \cdot 0.2) + (2 \cdot 0.3) + (3 \cdot 0.15) + (2 \cdot 0.25) = 2.25 \text{ questions.}$$

à comparer à l'entropie de X :

$$H(X) \approx 2.23 \text{ bits/symbde}.$$

On remarque que $E[N]$ est proche de $H(X)$ avec $E[N] > H(X)$ (sauf à perdre de l'information, on ne peut descendre en deça de $H(X)$ qui représente l'information moyenne intrinsèque).

TD 2 Corrigé. Cryptographie

Partie I. Entropie : un jeu d'espion !

Exercice 1. Rendons à César ce qui est à César : son chiffre !

1. L'ensemble des clairs possibles M, celui des chiffrés C ainsi que celui des clés K coïncident avec l'alphabet latin et on a :

$$\forall m \in \mathrm{M}, \quad P(M = m) = \frac{1}{26}$$

De plus, la fonction de chiffrement établit un lien entre le clair (m), le chiffré (c) et la clé (k) :

$$c = (m + k) \bmod 26$$

Alors, pour tout $c \in \mathrm{C}$ et pour tout $m \in \mathrm{M}$, on a : *Rappel :* $\begin{cases} P(A|B) = \dfrac{P(A \cap B)}{P(B)} \\ P(A \cap B) = P(A)P(B) \text{ si } AB \text{ indépendants} \end{cases}$

$$P[M = m | C = c] = \frac{P[M = m \text{ et } C = c]}{P[C = c]} = \frac{P[M = m \text{ et } K = c - m]}{P[C = c]}$$

et du fait de l'indépendance de m et k :

$$P[M = m | C = c] = \frac{P[M = m]P[K = c - m]}{P[C = c]} = \frac{P[M = m]P[K = k]}{P[C = c]}$$

Le choix du message et de la clé étant indépendants :

$$P(C = c) = P(K = k) = \frac{1}{26}$$

On a finalement :

$$P[M = m | C = c] = P[M = m]$$

2. Il est facile de constater qu'il n'existe aucun décalage de l'alphabet qui transforme le message $m = AB$ en le chiffré $c = DM$ (les lettres A et B sont consécutives dans l'alphabet, elles devraient l'être dans l'alphabet décalé).
Donc $P[M = m \text{ et } C = c] = 0$.

Nous avons alors $P[M = m | C = c] = 0 \quad (\neq P[M = m])$ car $P[M = m] = \frac{1}{26} \cdot \frac{1}{26} = \left(\frac{1}{26}\right)^2$.

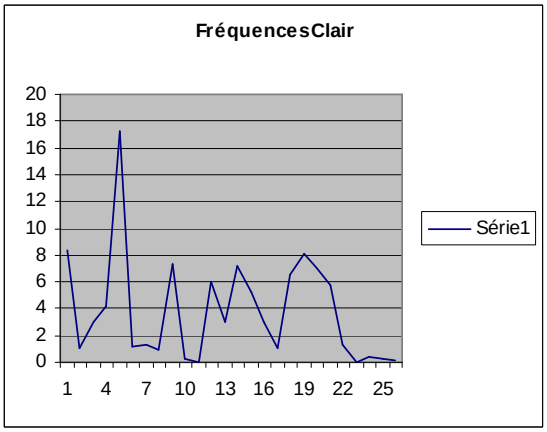
Le chiffrement de César avec $l > 1$ ne vérifie donc pas la condition de Shannon de système parfaitement sûr.

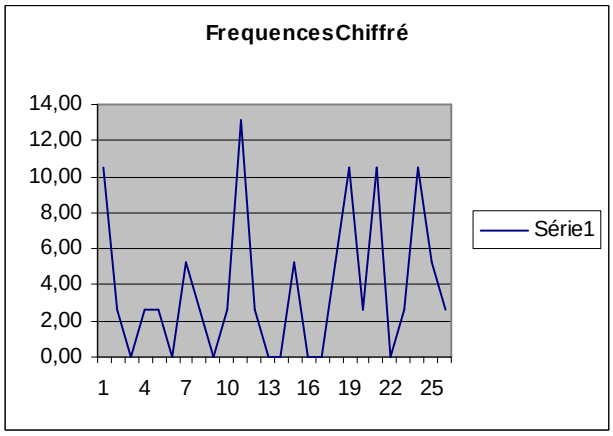
Exercice 2. Cryptanalyse du chiffre de César.

1. On peut trouver la clé par *force brute* (méthode exhaustive) (25 essais de décalage à faire). La méthode d'*analyse des fréquences* permet de trouver la clé (le décalage) car les lettres les plus fréquentes du message clair sont celles de la *langue française*.

Remarque : Avec la méthode d'analyse des fréquences, il faut que le texte soit suffisamment long (au moins la taille de l'alphabet latin, soit 26 lettres) sinon les fréquences ne sont pas significatives (théorème central limite des probabilités).

Clair : BELLE MARQUISE VOS YEUX ME FONT MOURIR D AMOUR
Chiffré : *HKRRK SGXWAOYK BUY EKAD SK LUTZ SUAXOX J GSUAX*





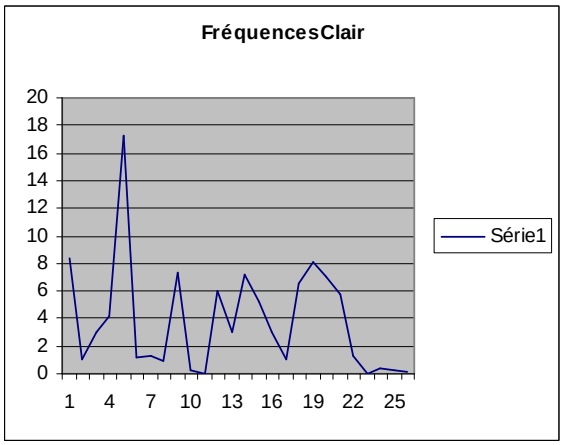
Décalage (clé) par analyse des fréquences : $k = 6$ = Pic Chiffré (11) - Pic Clair (5)
Décalage (clé) réel : $k = 6$

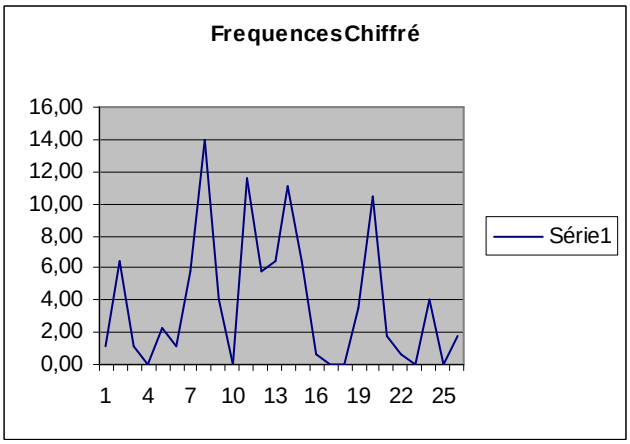
Lettre	*OccurChiffré*	*FréquencesChiffré*	*FréquencesClair*
A	4	10,53	8,4
B	1	2,63	1,06
C	0	0,00	3,03
D	1	2,63	4,18
E	1	2,63	17,26
F	0	0,00	1,12
G	2	5,26	1,27
H	1	2,63	0,92
I	0	0,00	7,34
J	1	2,63	0,31
K	5	13,16	0,05
L	1	2,63	6,01
M	0	0,00	2,96
N	0	0,00	7,13
O	2	5,26	5,26
P	0	0,00	3,01
Q	0	0,00	0,99
R	2	5,26	6,55
S	4	10,53	8,08
T	1	2,63	7,07
U	4	10,53	5,74
V	0	0,00	1,32
W	1	2,63	0,04
X	4	10,53	0,45
Y	2	5,26	0,3
Z	1	2,63	0,12
Som	38	100,00	99,97

2. On peut trouver la clé par *force brute* (méthode exhaustive) (25 essais de décalage à faire). La méthode d'*analyse des fréquences* est piégée ici car les lettres les plus fréquentes du message clair ne sont pas celles de la *langue française*.

Clair : BONJOUR NOUS VOUS PROPOSONS TROIS WAGONS POUR PARTIR AU ZANZIBAR VOTRE AVION VA PARTIR LA OU VOUS VOULEZ UN VRAI VAUTOUR VEUT TOUJOURS VOLER HAUT POUR VOIR SA PROIE UN ZOMBI MARCHAIT EN ZIGZAGANT SUR LA ROUTE

Chiffré : *UHGCHNK GHNL OHNL IKHIHLHGL MKHBL PTZHGL IHNK ITKMBK TN STGSBUTK OHMKX TOBHG OT ITKMBK ET HN OHNL OHNEXS NG OKTB OTNMHNK OXNM MHNCHNKL OHEXK ATNM IHNK OHBK LT IKHBX NG SHFUB FTKVATBM XG SBZSTZTGM LNK ET KHNMX*





Décalage (clé) par analyse des fréquences : $k = 3$ = Pic Chiffré (8) - Pic Clair (5)

Décalage (clé) réel : $k = 19$

Lettre	*OccurChiffré*	*FréquencesChiffré*	*FréquencesClair*
A	2	1,16	8,4
B	11	6,40	1,06
C	2	1,16	3,03
D	0	0,00	4,18
E	4	2,33	17,26
F	2	1,16	1,12
G	10	5,81	1,27
H	24	13,95	0,92
I	7	4,07	7,34
J	0	0,00	0,31
K	20	11,63	0,05
L	10	5,81	6,01
M	11	6,40	2,96
N	19	11,05	7,13
O	11	6,40	5,26
P	1	0,58	3,01
Q	0	0,00	0,99
R	0	0,00	6,55
S	6	3,49	8,08
T	18	10,47	7,07
U	3	1,74	5,74
V	1	0,58	1,32
W	0	0,00	0,04
X	7	4,07	0,45
Y	0	0,00	0,3
Z	3	1,74	0,12
Som	172	100,00	99,97

Partie II. Entropie et arbres de décision

Exercice 3. Le jeu de la fausse pièce.

1. $n = 8$ pièces.

(*a*) Il y a **2x8 = 16 réponses possibles.**

En effet, une réponse est un couple dont le 1er élément donne la fausse pièce et le 2nd le sens de la différence de poids. Si on suppose que les pièces sont numérotées et si on note – pour la pièce la moins lourde et + pour la pièce la plus lourde, les réponses possibles sont alors de la forme (i, -) ou (i, +) avec i = 1, 2, …, 8.

(*b*) **Le résultat de chaque pesée** peut prendre une des $3^1 = 3$ **valeurs possibles** G, E, D.

→ **Les résultats de 2 pesées** sont des **couples** prenant des valeurs dans $\{G, E, D\}$x$\{G, E, D\}$.
Il y a donc $3^2 = 9$ **résultats possibles.**

→ De façon analogue, **les résultats de 3 pesées** sont des **triplets** et il y a $3^3 = 27$ **valeurs possibles.**

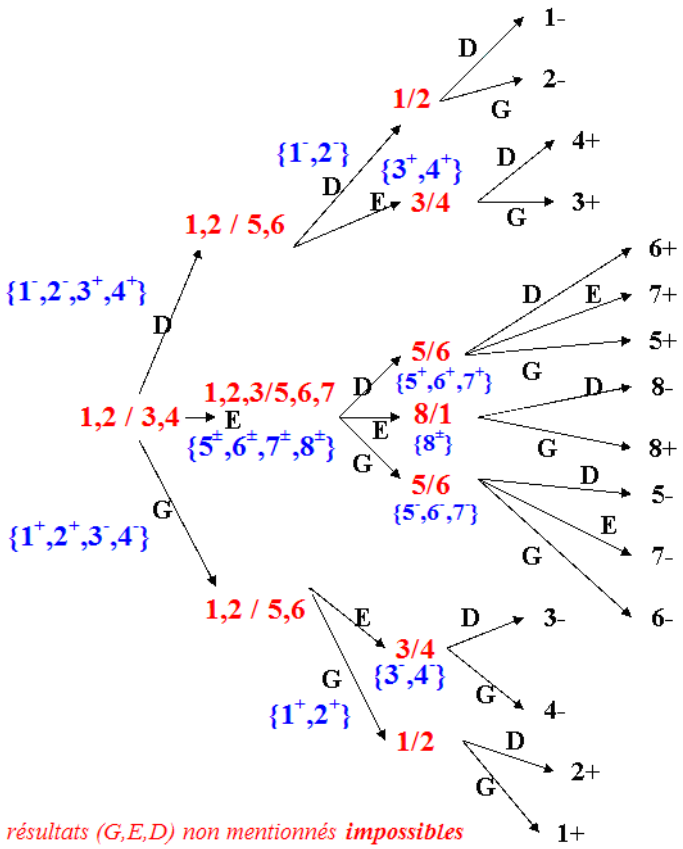
(*c*) **Non**, car le résultat de cette pesée est connu d'avance.

En effet, nous savons qu'il existe une seule pièce fausse. Elle sera obligatoirement dans l'un des 2 lots. La balance penchera vers la Gauche ou vers la Droite sans que cela nous apporte un maximum d'information.

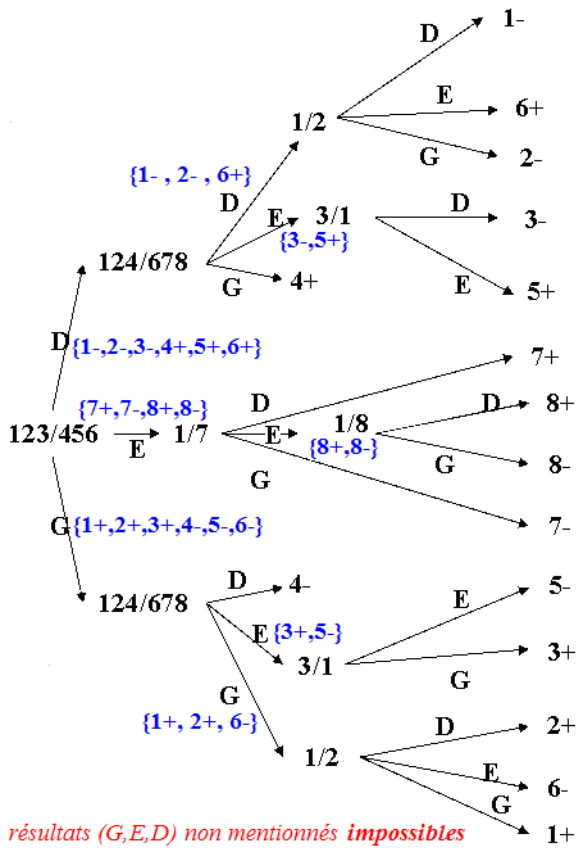
(*d*) Manipultation.

(*e*) **Ne pas effectuer de pesées dont on connaît le résultat (tout ou partie) et donc n'apportant aucune ou peu d'information** (entropie nulle ou faible). *Faire des pesées maximisant l'information apportée (donc l'entropie).*

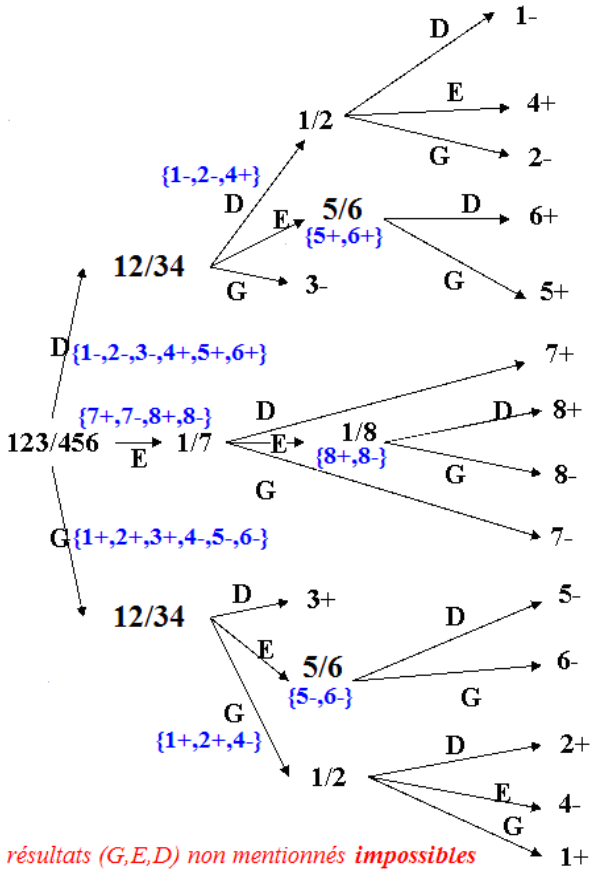
Voici une 1ère stratégie possible : n = 8 pièces, 3 pesées successives.



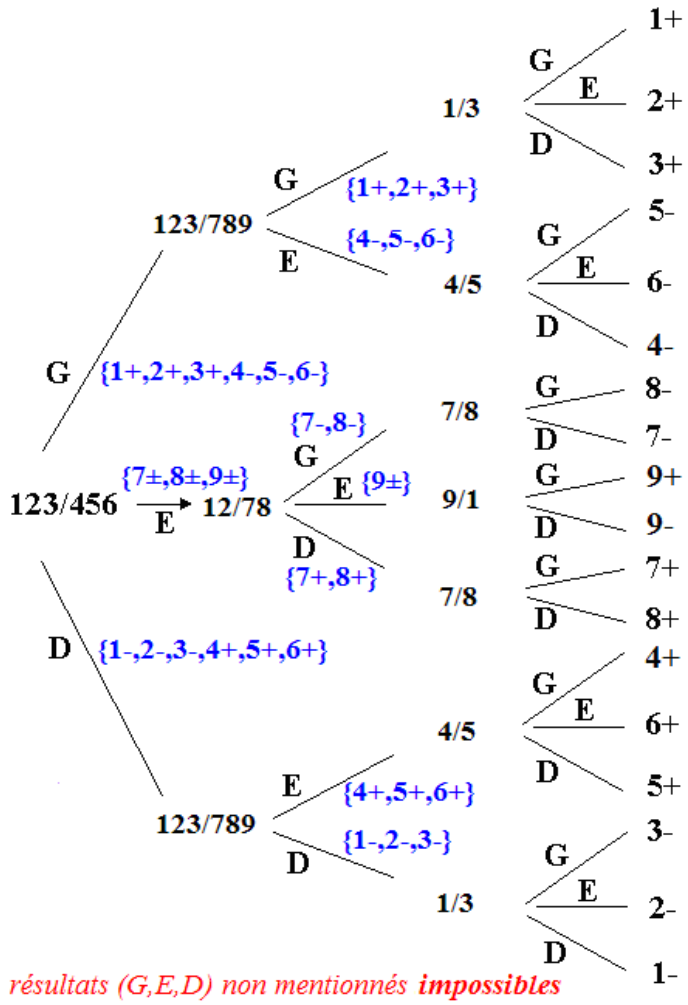
Voici une 2nde stratégie possible : n = 8 pièces, 3 pesées successives.



Enfin, voici une 3ème stratégie possible : n = 8 pièces, 3 pesées successives.



2. Voici une stratégie possible : n = 9 pièces, 3 pesées successives.



résultats (G,E,D) non mentionnés ***impossibles***

TD 3 Corrigé. Codage de source

Partie I. Codes sans préfixe.

Exercice 1. Décodage.

Procédure de décodage pas à pas : **1011 000 010 1010 100** donne la séquence ***dhief***

Exercice 2. Code sans préfixe.

Il suffit d'appliquer l'inégalité de Kraft, avec n = 10 (taille de l'alphabet de la *source*), d = 2 (*code binaire*) et $(l_1, l_2, \ldots, l_n)$ = {4, 4, 4, 3, 2, 4, 3, 4, 3, 4} :

Théorème 3.1. Inégalité de Kraft. Un **code instantané** (≡ **sans préfixe**) de longueurs de mots-données $(l_1, l_2, \ldots, l_n)$ existe si et seulement si

$$\sum_{i=1}^{n} d^{-l_i} \leq 1$$

où d est la taille de l'alphabet du *canal*, et n, la taille de l'alphabet de la *source* (d = 2 pour un canal *binaire*).

Ici : $\sum_{i=1}^{n} d^{-l_i} = \left(6 \cdot 2^{-4}\right) + \left(3 \cdot 2^{-3}\right) + \left(1 \cdot 2^{-2}\right) = \frac{6}{16} + \frac{3}{8} + \frac{1}{4} = \frac{6+6+4}{16} = 1$

→ l'inégalité de Kraft est vérifiée → un *code binaire sans préfixe* existe avec ces longueurs de mots.

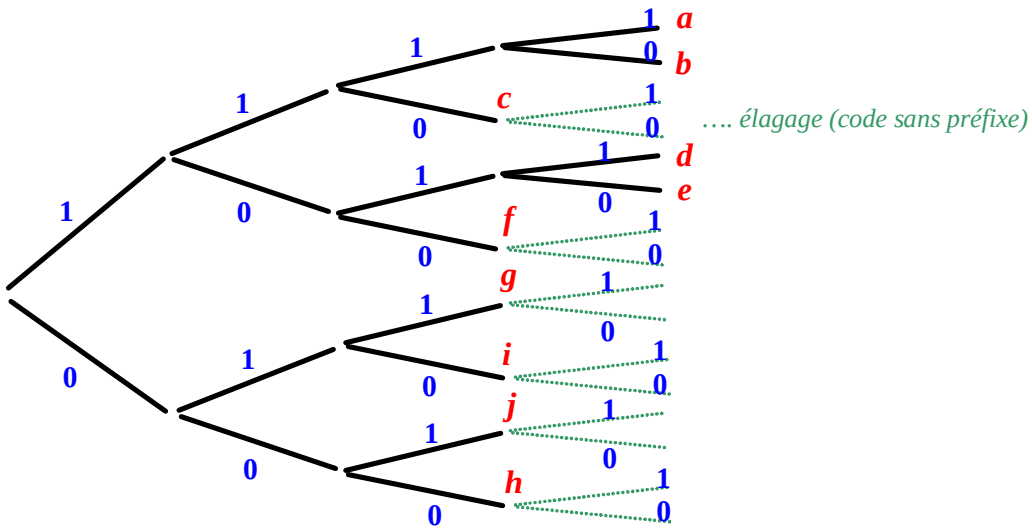
Partie II. Codes sans préfixe et arbres binaires.

Exercice 3. Code sans préfixe. Arbre binaire.

1. Arbre associé au code suivant :

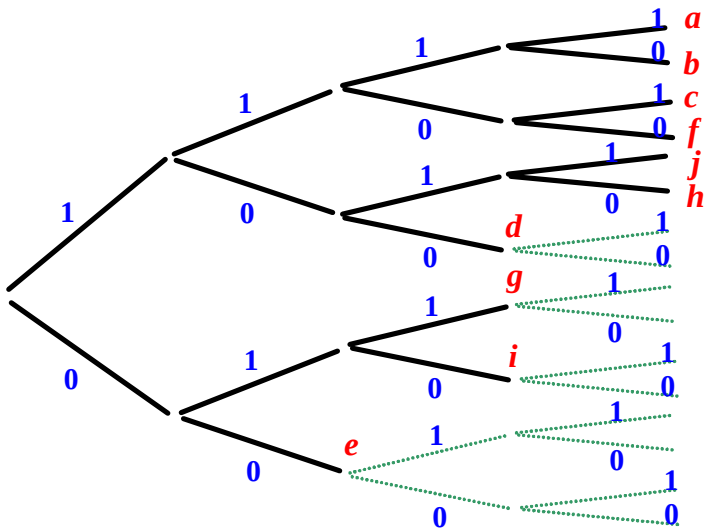
s_i	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
m_i	1111	1110	110	1011	1010	100	011	000	010	001

Il s'agit d'un *code sans préfixe*.



2. Un *code sans préfixe* de longueurs de mots suivantes :

s_i	a	b	c	d	e	f	g	h	i	j
l_i	4	4	4	3	2	4	3	4	3	4



…. élagage (code sans préfixe)

→ *Table de code :*

s_i	a	b	c	d	e	f	g	h	i	j
m_i	1111	1110	1101	100	00	1100	011	1010	010	1011
l_i	4	4	4	3	2	4	3	4	3	4

Partie III. Codage de Huffman

Exercice 4. Code sans préfixe optimal. Méthode de Huffman. Source d'alphabet Ω et de distribution de probabilité :

s_i	a	b	c	d	e	f	g	h	i	j
p_i	0.2	0.05	0.1	0.05	0.15	0.05	0.1	0.05	0.15	0.1

1. *Longueur moyenne minimale* pour un code *binaire* de cette source : on utilise le *théorème de la borne inférieure* :

Théorème 3.3. Borne inférieure. Soit une source S d'alphabet $\Omega_S = \{s_1, \cdots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \cdots, p_n\}$. Soit un canal d'alphabet binaire $\Omega_C = \{0,1\}$, donc de taille $d = 2$, sans bruit, stationnaire et sans mémoire. Soit un code déchiffrable $\{m_1, \cdots, m_n\}$ de longueurs de mots $\{l_1, \cdots, l_n\}$.
Alors la longueur moyenne de mots de code vérifie :

$$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) \geq \frac{H(S)}{\log_2(d)} \quad \Rightarrow \quad \left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) \geq H(S)$$

L'égalité n'est possible que si $\forall i = 1, \cdots, n \qquad p_i = d^{-l_i} = 2^{-l_i}$.

→ La *longueur moyenne minimale* des mots d'un code binaire de cette source est l'entropie :

$$\overline{L}_{\min} = \frac{H(S)}{\log_2(d)} = H(S) = \sum_{i=1}^{n=10} p_i \log_2(p_i) \text{ avec } H(S) = \sum_{i=1}^{n=10} p_i \log_2(p_i) = 3.14 \text{ bits/symbole} \rightarrow \overline{L}_{\min} = 3.14 \text{ bits/symbole}$$

2. Un *code absolument optimal* est tel que la *longueur moyenne des mots-codes est égale à la borne inférieure* $\frac{H(S)}{\log_2(d)}$. Le *théorème de la borne inférieure* indique que la borne inférieure est atteinte si les probabilités des symboles de la source sont des puissances de 2 : $\forall i = 1, \cdots, n \quad p_i = d^{-l_i} = 2^{-l_i}$. Ce n'est pas le cas ici.

→ Un *code absolument optimal* n'existe pas pour cette source, car les probabilités des symboles de la source ne sont pas toutes des puissances de 2.

3. Construction d'un *code sans préfixe optimal* selon la *méthode de Huffman* :
On trie les caractères de la source par *probabilité décroissante* pour que les symboles les plus probables (revenant donc souvent dans le message émis) soient codés avec un nombre de bit *minimal* (→ réduction de la quantité d'information, du débit, pour compression par exemple).

Parcours feuilles → racine :
Ici, on regroupe les 2 caractères les moins probables (***f*** et ***h***), on l'appelle ***fh***, on additionne les probabilités correspondantes et on recommence.

s_i	*a*	*e*	*i*	*c*	*g*	*j*	*b*	*d*	***f***	***h***
p_i	0.2	0.15	0.15	0.1	0.1	0.1	0.05	0.05	**0.05**	**0.05**

s_i	*a*	*e*	*i*	*c*	*g*	*j*	*fh*	***b***	***d***
p_i	0.2	0.15	0.15	0.1	0.1	0.1	0.1	**0.05**	**0.05**

s_i	*a*	*e*	*i*	*c*	*g*	*j*	***fh***	***bd***
p_i	0.2	0.15	0.15	0.1	0.1	0.1	**0.1**	**0.1**

s_i	*a*	*fhbd*	*e*	*i*	*c*	***g***	***j***
p_i	0.2	0.2	0.15	0.15	0.1	**0.1**	**0.1**

s_i	*a*	*fhbd*	*gj*	*e*	***i***	***c***
p_i	0.2	0.2	0.2	0.15	**0.15**	**0.1**

s_i	*ic*	*a*	*fhbd*	***gj***	***e***
p_i	0.25	0.2	0.2	**0.2**	**0.15**

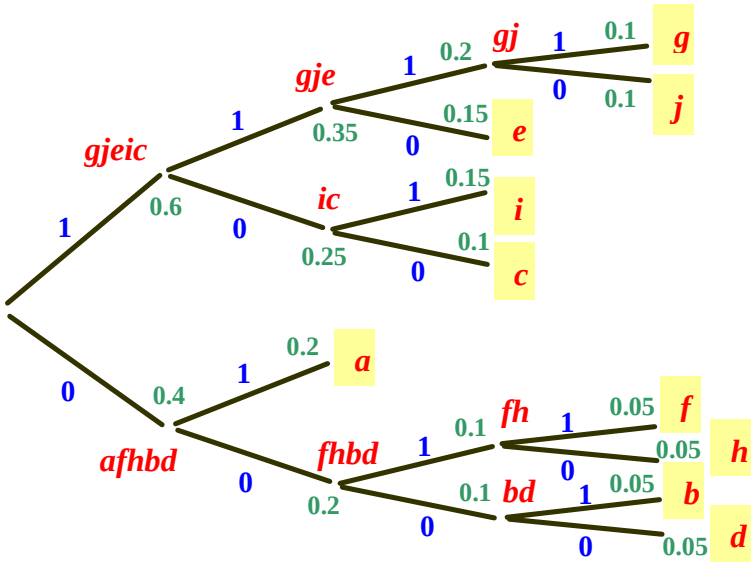
s_i	*gje*	*ic*	***a***	***fhbd***
p_i	0.35	0.25	**0.2**	**0.2**

s_i	*afhbd*	***gje***	***ic***
p_i	0.4	**0.35**	**0.25**

s_i	*gjeic*	*afhbd*
p_i	0.6	0.4

4. Représentation du *code* sous forme *d'arbre* :

Parcours racine → feuilles : *Convention arbitraire :* 1 ≡ symboles les plus probables, 0 ≡ symboles les moins probables



Un code optimal sans préfixe

→ *Table de code :*

s_i	a	b	c	d	e	f	g	h	i	j
p_i	0.2	0.05	0.1	0.05	0.15	0.05	0.1	0.05	0.15	0.1
m_i	**01**	**0001**	**100**	**0000**	**110**	**0011**	**1111**	**0010**	**101**	**1110**
l_i	2	4	3	4	3	4	4	4	3	4

Longueur moyenne du code :

$$\bar{L} = \sum_{i=1}^{n=10} p_i l_i = 2(0.2) + 4(0.05) + 3(0.1) + 4(0.05) + 3(0.15) + 4(0.05) + 4(0.1) + 4(0.05) + 3(0.15) + 4(0.1) = 3.2 \text{ bits/symbole}$$

$$\bar{L} > \left(H(S) = 3.14 \text{ bits/symbole}\right)$$

TD 4 Corrigé. Compression de données
Codage et compression d'images sans pertes

Exercice 1. Codages RLE (*Run Length Encoding*) et Huffman.

Méthode 1. Codage de Huffman.

1. Statistique du nombre de pixels (B, N, G) de l'image :

Ligne	B	N	G
1	8	1	0
2	6	3	0
3	4	0	5
4	2	1	6
5	0	3	6
6	2	1	6
7	4	0	5
8	6	3	0
9	8	1	0
Cumul	**40**	**13**	**28**

81

Chaque pixel étant codé sur 8 bits, l'image occupe une taille de : 9×9×8 = 648 bits = 81 Octets.

→ *Table des fréquences des couleurs présentes dans l'image* :

Couleur	B	N	G
Nombre d'occurences	40	13	28
Fréquence p_i	**40/81**	**13/81**	**28/81**

2. *Entropie :* $$H = -\frac{40}{81}\log_2\left(\frac{40}{81}\right) - \frac{13}{81}\log_2\left(\frac{13}{81}\right) - \frac{28}{81}\log_2\left(\frac{28}{81}\right) \approx 1.46 \text{ bits/symbole}$$

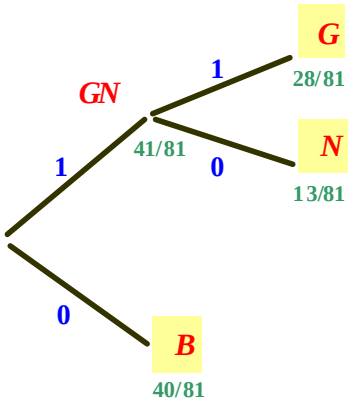
Efficacité du codage qui associe 8 bits à chaque pixel : L'entropie H (~ 1.46 bits/symbole) est inférieure au codage utilisé (8 bits/pixel) → Une compression sans pertes de l'image est possible.

3. *Arbre de Huffman* :

s_i	B	***G***	***N***
p_i	40/81	**28/81**	**13/81**

s_i	*GN*	B
p_i	41/81	40/81

Convention arbitraire : 1 ≡ symboles les plus probables, 0 ≡ symboles les moins probables



Un code optimal sans préfixe

Code associé :

s_i	*B*	*G*	*N*
p_i	40/81	28/81	13/81
m_i	**0**	**11**	**10**
l_i	1	2	2

4. *Longueur du code :*
40 pixels *B* → 40 x 1 bit = 40 bits
28 pixels *G* → 28 x 2 bits = 56 bits
13 pixels *N* → 13 x 2 bits = 26 bits
→ *Taille du code de Huffman* (en bits) *pour cette image :* 40 + 56 +26 = **122 bits**

Taux de compression : $t_1 = \frac{648-122}{648} \approx 81\%$

Rappel : $\begin{cases} \text{Taux de Compression } (\%) = \frac{NC - C}{NC} = 1 - \frac{C}{NC} \\ NC : \text{taille des données non compressées} \\ C : \text{taille des données compressées} \end{cases}$

Méthode 2. Codage RLE.

1. *Code RLE :*
(4, *B*) (1, *N*) (7, *B*) (3, *N*) (5, *B*) (5, *G*) (3, *B*) (3, *G*) (1, *N*) (3, *G*) (1, *B*) (3, *G*) (3, *N*) (3, *G*) (1, *B*) (3, *G*) (1, *N*) (3, *G*) (3, *B*) (5, *G*) (5, *B*) (3, *N*) (7, *B*) (1, *N*) (4, *B*)

2. *Longueur du code :* 25 couples x 16 bits = 400 bits → *Taille du code RLE* (en bits) *pour cette image :* **400 bits**

3. *Taux de compression* : $t_2 = \frac{648-400}{648} \approx 38\%$

Méthode 3. Codage RLE + Huffman.

1. *Table des fréquences des couples (nb, c) présents dans le code RLE de l'image* :

Couple	(4, *B*)	(1, *N*)	(7, *B*)	(3, *N*)	(5, *B*)	(5, *G*)	(3, *B*)	(3, *G*)	(1, *B*)
Nombre d'occurences	2	4	2	3	2	2	2	6	2
Fréquence p_i	**2/25**	**4/25**	**2/25**	**2/25**	**2/25**	**2/25**	**2/25**	**6/25**	**2/25**

2. *Entropie :* $H = -6 \cdot \frac{2}{25}\log_2\left(\frac{2}{25}\right) - \frac{3}{25}\log_2\left(\frac{3}{25}\right) - \frac{4}{25}\log_2\left(\frac{4}{25}\right) - \frac{6}{25}\log_2\left(\frac{6}{25}\right) \approx 3.03 \text{ bits/symbole}$

Efficacité du codage qui associe 16 bits à chaque couple : L'entropie H (~ 3.03 bits/symbole) est inférieure au codage utilisé (16 bits/couple) → Une compression sans pertes de l'image est possible.

3. *Arbre de Huffman* : **Pour alléger la notation, on associe une lettre à chaque couple**

s_i	(3, *G*) ≡ *a*	(1, *N*) ≡ *b*	(3, *N*) ≡ *c*	(4, *B*) ≡ *d*	(7, *B*) ≡ *e*	(5, *B*) ≡ *f*	(5, *G*) ≡ *g*	**(3, *B*) ≡ *h***	**(1, *B*) ≡ *i***
p_i	6/25	4/25	3/25	2/25	2/25	2/25	2/25	**2/25**	**2/25**

s_i	*a*	*b*	*Hi*	*c*	*d*	*e*	***f***	***g***
p_i	6/25	4/25	4/25	3/25	2/25	2/25	**2/25**	**2/25**

s_i	*a*	*b*	*Hi*	*fg*	*c*	***d***	***e***
p_i	6/25	4/25	4/25	4/25	3/25	**2/25**	**2/25**

s_i	*a*	*b*	*Hi*	*fg*	***de***	***c***
p_i	6/25	4/25	4/25	4/25	**4/25**	**3/25**

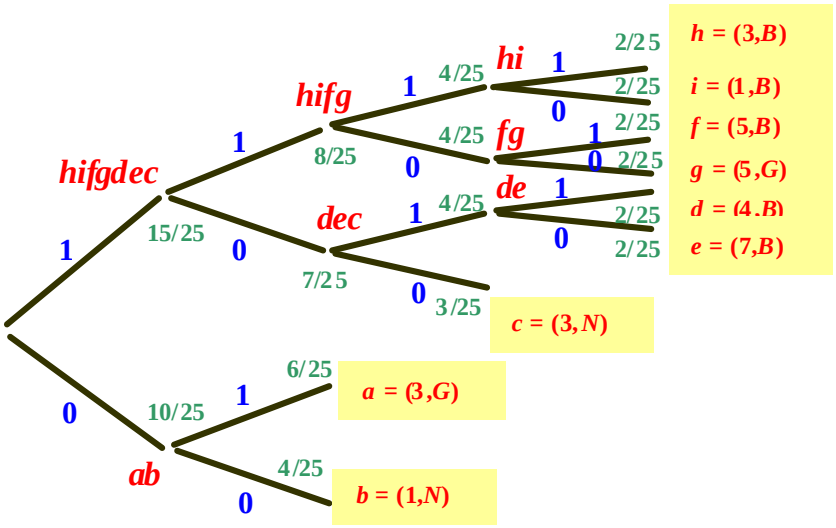
s_i	*dec*	*a*	*B*	***hi***	***fg***
p_i	7/25	6/25	4/25	**4/25**	**4/25**

s_i	*hifg*	*dec*	***A***	***b***
p_i	8/25	7/25	**6/25**	**4/25**

s_i	*ab*	***hifg***	***Dec***
p_i	10/25	**8/25**	**7/25**

s_i	*hifgdec*	*ab*
p_i	15/25	10/25

Convention arbitraire : 1 ≡ symboles les plus probables, 0 ≡ symboles les moins probables



Un code optimal sans préfixe

Code associé :

s_i	(3, *G*) ≡ *a*	(1, *N*) ≡ *b*	(3, *N*) ≡ *c*	(4, *B*) ≡ *d*	(7, *B*) ≡ *e*	(5, *B*) ≡ *f*	(5, *G*) ≡ *g*	**(3, *B*) ≡ *h***	**(1, *B*) ≡ *i***
p_i	6/25	4/25	3/25	2/25	2/25	2/25	2/25	**2/25**	**2/25**
m_i	**01**	**00**	**100**	**1011**	**1010**	**1101**	**1100**	**1111**	**1110**
l_i	2	2	3	4	4	4	4	4	4

4. *Longueur du code :* (6 couples x 2 bits) + (4 couples x 2 bits) + (3 couples x 3 bits) + 6 x (2 couples x 4 bits) = 77 bits
→ *Taille du code de Huffman* (en bits) *pour la séquence du code RLE de cette image* = **77 bits**

Taux de compression : $t_3 = \frac{648 - 77}{648} \approx 88.1\%$

5. La dernière méthode (*Codage RLE + Huffman*) présente le taux de compression le plus élevé.

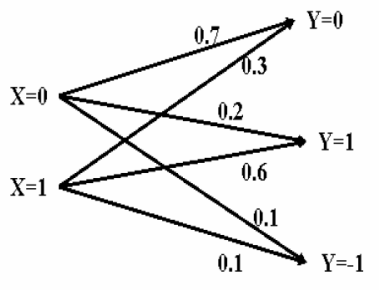
TD 5 Corrigé. Modélisation mathématique d'un canal

Partie I. Représentation mathématique d'un canal de transmission

Exercice 2. Un modèle probabiliste de transmission.

1. Les probabilités de réception données définissent la *distribution de probabilités conditionnelle* $P(Y|X)$.

2. Représentation sous forme de *graphe des probabilités de réception* :



3. Matrice de *distribution conditionnelle* $P(Y|X)$:

$P(Y|X) =$

$X \setminus Y$	0	1	-1
0	0.7	0.2	0.1
1	0.3	0.6	0.1

4. *Distributions* manquantes :

▪ **Distribution conjointe de** X **et** Y : $P(X,Y)$ On utilise la définition avec $n = 2$ et $m = 3$:

$$p(x_i, y_j) = p(y_j \mid x_i)p(x_i) \quad i = 1,\ldots,n \quad j = 1,\ldots,m$$

→

$P(X,Y) =$

$X \setminus Y$	0	1	-1
0	(0.7 x 0.5) **0.35**	(0.2 x 0.5) **0.1**	(0.1 x 0.5) **0.05**
1	(0.3 x 0.5) **0.15**	(0.6 x 0.5) **0.3**	(0.1 x 0.5) **0.05**

▪ **Distributions marginales** de X : $P_X = P(X)$, et de Y : $P_Y = P(Y)$ Définition avec $n = 2$ et $m = 3$:

$P_X = P(X)$: $p(x_i) = \sum_{j=1}^{m} p(x_i, y_j) \quad i = 1,\ldots,n$ $\qquad$ $P_Y = P(Y)$: $p(y_j) = \sum_{i=1}^{n} p(x_i, y_j) \quad j = 1,\ldots,m$

→ $P_X = P(X)$ (donnée) s'obtient comme **la somme des éléments de chaque ligne** de $P(X,Y)$: $\begin{cases} P_X(0) = 0.35 + 0.1 + 0.05 = 0.5 \\ P_X(1) = 0.15 + 0.3 + 0.05 = 0.5 \end{cases}$

→ $P_Y = P(Y)$ s'obtient comme **la somme des éléments de chaque colonne** de $P(X,Y)$: $\begin{cases} P_Y(0) = 0.35 + 0.15 = 0.5 \\ P_Y(1) = 0.1 + 0.3 = 0.4 \\ P_Y(-1) = 0.05 + 0.05 = 0.1 \end{cases}$

→

$P(X,Y) =$

$X \setminus Y$	0	1	-1	P_X
0	0.35	0.1	0.05	**0.5**
1	0.15	0.3	0.05	**0.5**
P_Y	**0.5**	**0.4**	**0.1**	

▪ **Distributions conditionnelles** $P(X|Y)$ et $P(Y|X)$ On utilise la définition avec $n=2$ et $m=3$:

$$P(X|Y): \quad p(x_i \mid y_j) = \frac{p(x_i, y_j)}{p(y_j)} \qquad i = 1,\ldots,n \qquad j = 1,\ldots,m$$

→

$P(X|Y) =$

$X \setminus Y$	0	1	-1
0	(0.35 / 0.5) **0.7**	(0.1 / 0.4) **0.25**	(0.05 / 0.1) **0.5**
1	(0.15 / 0.5) **0.3**	(0.3 / 0.4) **0.75**	(0.05 / 0.1) **0.5**

$$P(Y|X): \quad p(y_j \mid x_i) = \frac{p(x_i, y_j)}{p(x_i)} \qquad i = 1,\ldots,n \qquad j = 1,\ldots,m \quad \text{(donnée)}$$

→

$P(Y|X) =$

$X \setminus Y$	0	1	-1
0	(0.35 / 0.5) **0.7**	(0.1 / 0.5) **0.2**	(0.05 / 0.5) **0.1**
1	(0.15 / 0.5) **0.3**	(0.3 / 0.5) **0.6**	(0.05 / 0.5) **0.1**

5. *Entropies* associées au *canal* : $H(X)$, $H(Y)$, $H(X,Y)$, $H(X|Y)$ et $H(Y|X)$

▪ $H(X)$**. L'entropie de la source.** La source binaire X ayant une *distribution uniforme* :

$$H(X) = -\sum_{i=1}^{n=2} p(x_i)\log_2[p(x_i)] = \log_2(2) = 1 \text{ bit/symbole.}$$

▪ $H(Y)$**. L'entropie du récepteur.** On utilise la *distribution marginale* $P_Y = P(Y)$:

$$H(Y) = -\sum_{i=1}^{m=3} p(y_i)\log_2[p(y_i)] = -0.5\log_2(0.5) - 0.4\log_2(0.4) - 0.1\log_2(0.1) \approx 1.36 \text{ bits/symbole.}$$

▪ $H(X,Y)$**. L'entropie conjointe "source-récepteur".** On utilise la *distribution conjointe* $P(X,Y)$:

$$H(X,Y) = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2[p(x_i, y_j)] \approx 2.22 \text{ bits/symbole.}$$

▪ $H(X|Y)$**. L'entropie conditionnelle de la source, sachant le symbole reçu.**

On utilise la *distribution conjointe* $P(X,Y)$ et la *distribution conditionnelle* $P(X|Y)$:

$$H(X|Y) = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2\left[\frac{p(x_i, y_j)}{p(y_j)}\right] = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2[p(x_i|y_j)] \approx 0.86 \text{ bit/symbole.}$$

▪ $H(Y|X)$**. L'entropie conditionnelle du récepteur, sachant le symbole émis.**

On utilise la *distribution conjointe* $P(X,Y)$ et la *distribution conditionnelle* $P(Y|X)$:

$$H(Y|X) = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2\left[\frac{p(x_i, y_j)}{p(x_i)}\right] = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2[p(y_j|x_i)] \approx 1.22 \text{ bit/symbole.}$$

6. Déduction de *l'information mutuelle* $I(X;Y)$ du *canal :*

$$I(X;Y) = I(Y;X) = H(Y) - H(Y|X) = H(X) - H(X|Y) \approx 0.14 \text{ bit/symbole.}$$

Exercice 3. Capacité de canal.

1. *Canal symétrique* ? **Non**, ce canal ne correspond pas à la *définition d'un canal symétrique* :

Rappel : Définition d'un canal symétrique : « Un *canal symétrique* est caractérisé par une propriété particulière de sa matrice de transition $P(X|Y)$. *Toutes les lignes de cette matrice sont équivalentes entre elles par permutation et toutes les colonnes également*. Cela veut dire *que toutes les lignes sont obtenues par permutation d'une seule d'entre elles, et toutes les colonnes sont obtenues par permutation d'une seule d'entre elles*. »

Certes, ici les *lignes* de la *matrice de transition sont toutes équivalentes par permutation*, mais pas les *colonnes*.

2. ▪ **Distribution de probabilité conjointe de X et Y : $P(X,Y)$** On utilise la définition de la *distribution conditionnelle* $P(Y|X)$ impliquant la distribution *marginale* de X, P_X, avec $n=2$ et $m=3$:

$$P(Y|X):\ p(y_j \mid x_i)=\frac{p(x_i,y_j)}{p(x_i)} \quad i=1,\dots,n \quad j=1,\dots,m \qquad \rightarrow \qquad p(x_i,y_j)=p(y_j \mid x_i)p(x_i)$$

On en déduit :

▪ **Distribution marginale** de Y : $P_Y=P(Y)$ On utilise la définition avec $n=2$ et $m=3$:

$$P_Y=P(Y):\ p(y_j)=\sum_{i=1}^{n} p(x_i,y_j) \quad j=1,\dots,m$$ **Somme des éléments de chaque colonne** de $P(X,Y)$

→

$P(X,Y)=$

$X \setminus Y$	0	-1	1	P_X
0	**0.8*p***	**0.2*p***	**0**	***p***
1	**0**	**0.2(1-*p*)**	**0.8(1-*p*)**	**1-*p***
P_Y	**0.8*p***	**0.2**	**0.8(1-*p*)**	

On en déduit :

▪ **Distribution conditionnelle** $P(X|Y)$ On utilise la définition avec $n=2$ et $m=3$:

$$P(X|Y):\ p(x_i \mid y_j)=\frac{p(x_i,y_j)}{p(y_j)} \quad i=1,\dots,n \quad j=1,\dots,m$$

→

$P(X|Y)=$

$X \setminus Y$	0	-1	1
0	(0.8*p* / 0.8*p*) **1**	(0.2*p* / 0.2) ***p***	(0 / 0.8(1-*p*)) **0**
1	(0 / 0.8*p*) **0**	(0.2(1-*p*) / 0.2) **1-*p***	(0.8(1-*p*) / 0.8(1-*p*)) **1**

3. ▪ $H_X(p)$**. L'entropie de la source.**

$$H_X(p)=-\sum_{i=1}^{n=2} p(x_i)\log_2[p(x_i)]=-p\log_2(p)-(1-p)\log_2(1-p)$$

4. ▪ $H(X|Y)$. **L'entropie conditionnelle de la source, sachant le symbole reçu.**

On utilise la *distribution conjointe* $P(X,Y)$ et la *distribution conditionnelle* $P(X|Y)$:

$$H(X|Y) = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2\left[\frac{p(x_i, y_j)}{p(y_j)}\right] = -\sum_{i=1}^{n=2}\sum_{j=1}^{m=3} p(x_i, y_j)\log_2\left[p(x_i|y_j)\right]$$

$\rightarrow \quad H(X|Y) = -0.8p\log_2(1) - 0.2p\log_2(p) - 0.2(1-p)\log_2(1-p) - 0.8(1-p)\log_2(1)$

$\rightarrow \quad H(X|Y) = -0.2p\log_2(p) - 0.2(1-p)\log_2(1-p)$

$\rightarrow \quad H(X|Y) = 0.2H_X(p)$

5. $I(X;Y) = H_X(p) - H(X|Y) = H_X(p) - 0.2H_X(p) = 0.8H_X(p)$

6. $C = \max_{P_X} I(X;Y) \;\rightarrow\; C = \max_{p\in[0,1]} 0.8H_X(p)$

car la distribution d'une source *binaire* est complètement définie par la donnée de p.

Or, l'entropie d'une *source binaire,* $H_X(p)$**,** atteint son *maximum* ($\log_2(2)$) pour p = 0.5 (*équiprobabilité, distribution uniforme*) :

$\rightarrow C = \max_{p\in[0,1]} 0.8H_X(p) = 0.8H_X(p=0.5) = 0.8\log_2(2) = 0.8$ bit/symbole.

Partie II. Pour approfondir

Exercice 4. Nombre d'itérations d'un algorithme.

1. 4 cas possibles ($X_0 \in \{1,2,3,4\}$) :

▪ $X_0 = 1$: L'algorithme s'arrête tout de suite, sans entrer dans la boucle *Tant que*

$\rightarrow N = 0$ avec la probabilité 1.

▪ $X_0 = 2$: X_0 est pair $\rightarrow \forall A_0$, on a $X_1 = \frac{X_0}{2} = 1$ et $N = 1$ puis l'algorithme s'arrête

$\rightarrow N = 1$ avec la probabilité 1.

▪ $X_0 = 3$: L'exemple est traité à la figure **2.**

$$\rightarrow \begin{cases} N = 1 \text{ avec probabilité } \frac{1}{3} \\ N = 3 \text{ avec probabilité } \frac{2}{3} \end{cases}$$

▪ $X_0 = 4$: L'algorithme effectue 2 divisions par 2 indépendamment des valeurs aléatoires A_n

$\rightarrow N = 2$ avec la probabilité 1.

$\rightarrow$ On obtient la *matrice de probabilités conditionnelles* $P(N|X)$:

$P(N|X) =$

$X \setminus N$	0	1	2	3
1	**1**	**0**	**0**	**0**
2	**0**	**1**	**0**	**0**
3	**0**	**1/3**	**0**	**2/3**
4	**0**	**0**	**1**	**0**

2. ▪ **Distribution marginale** de X : $P_X = P(X)$:

X *uniformément distribuée* $\rightarrow P_X(1) = P_X(2) = P_X(3) = P_X(4) = \frac{1}{4}$

▪ **Distribution conjointe de** X **et** N : $P(X,N)$ On utilise la définition avec $n=4$ et $m=4$:

$$p(x_i, N_j) = p(N_j \mid x_i)p(x_i) \quad i=1,\ldots,n \quad j=1,\ldots,m \qquad (p(x_i) = \frac{1}{4} \quad i=1,\ldots,n)$$

$\rightarrow$

$P(X,N) =$

$X \setminus N$	0	1	2	3
1	**1/4**	**0**	**0**	**0**
2	**0**	**1/4**	**0**	**0**
3	**0**	**1/12**	**0**	**1/6**
4	**0**	**0**	**1/4**	**0**

3. ▪ **Distribution marginale** de N : $P_N = P(N)$ On utilise la définition avec $n=4$ et $m=4$:

$$P_N = P(N) : \; p(N_j) = \sum_{i=1}^{n} p(x_i, N_j) \quad j=1,\ldots,m$$ **Somme des éléments de chaque colonne** de $P(X,N)$

$\rightarrow$

$P(X,N) =$

$X \setminus N$	0	1	2	3	P_X
1	1/4	0	0	0	**1/4**
2	0	1/4	0	0	**1/4**
3	0	1/12	0	1/6	**1/4**
4	0	0	1/4	0	**1/4**
P_N	**1/4**	**1/3**	**1/4**	**1/6**	

4. A partir de la *distribution marginale* de N : $P_N = P(N)$, on détermine $E[N]$:

N	0	1	2	3
P_N	**1/4**	**1/3**	**1/4**	**1/6**

$\rightarrow E[N] = 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{3} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{6} = \frac{4}{3}$ itérations.

———————

TD 6 Corrigé. Codage de canal.
Codes linéaires correcteurs d'erreurs

Exercice 1. Code à répétition.

I. Expérience.

Manipulation (facultative)

1. Fichier *image.sci* : *tauxErr* = 0.1 : le *décodage* n'est pas parfait : des *erreurs* sont produites au décodage.
2. *tauxErr* = 0.05 : moins d'*erreurs* issues du décodage; *tauxErr* = 0.3 : davantage d'*erreurs* issues du décodage.

Interpétation

3. Lien entre la *matrice de cumul d'erreurs* et la possibilité de *correction des erreurs* ?

- Les mots-sources sont : 0 et 1
- Les mots du code sont : $\omega_1 = 000$ et $\omega_2 = 111$

On décode « 0 » par vote majoritaire si après transmission il y a 2 ou 3 « 0 » dans le mot reçu. Idem pour « 1 ».

Supposons que l'on envoie le mot-code « 000 » correspondant donc au mot source « 0 » à émettre. On a 4 cas possibles d'erreurs de transmission :

- **0 erreur :** on reçoit le mot $m = 000$: on décode « 0 » par vote majoritaire (OK).
- **1 erreur :** on reçoit le mot $\begin{cases} m = 100 \\ \text{ou } m = 010 \\ \text{ou } m = 001 \end{cases}$: on décode « 0 » par vote majoritaire (OK)
  → **1 erreur est corrigée.**
- **2 erreurs :** on reçoit le mot $\begin{cases} m = 101 \\ \text{ou } m = 110 \\ \text{ou } m = 011 \end{cases}$: on décode « 1 » par vote majoritaire (au lieu de « 0 » !)
  → **2 erreurs ne peuvent pas être corrigées.**
- **3 erreurs :** on reçoit le mot $m = 111$: on décode « 1 » par vote majoritaire (au lieu de « 0 » !)
  → **3 erreurs ne peuvent pas être corrigées.**
  ici on ne se rend même pas compte qu'il y a eu des erreurs ! (*unanimité* du vote majoritaire)

Conclusion : le décodage est correct seulement s'il y a 0 ou 1 erreur sur les 3 transmissions.

II. Analyse.

1. *Distance minimale* du code : comme il n'y a que 2 mots-codes : $\begin{cases} \omega_1 = 000 \\ \omega_2 = 111 \end{cases}$ la *distance minimale du code* est :

$$d = d_h(000,111) = 3$$

Capacité de détection : on utilise la définition : $e_d = d - 1 = 2 \;\rightarrow\; e_d = 2$:
au maximum, il est **toujours possible** de *détecter* 2 *erreurs*.

Capacité de correction : on utilise la définition : $e_c = E\left[\frac{d-1}{2}\right] \;\rightarrow\; e_c = 1$:
au maximum, il est **toujours possible** de *corriger* 1 *erreur*.

On confirme l'observation (*interprétation de l'expérience*) où on a vu qu'1 erreur au maximum pouvait être corrigée.

2. Code *linéaire* ? **Oui**. On applique la définition :

Définition 6.15. Code correcteur linéaire (*k*, *n*). Un code linéaire (*k*, *n*) est un code de classe (*k*, *n*) tel que l'application $\phi : G_k \to G_n$ qui le génère est une application linéaire, c'est-à-dire, tel qu'il existe une matrice $G \in \mathrm{M}^{n,k}(0,1)$ telle que :

$$\phi(m) = Gm \qquad \forall m = \begin{pmatrix} m_1 \\ \vdots \\ m_p \end{pmatrix}$$

On appelle G la **matrice génératrice** du code. Elle est de dimension $n \times k$.

Ici, on a :

k = 1 colonne : (*dimension* du code) taille des mots-sources ($m_1 = 0,\ m_2 = 1$) (avant codage)

n = 3 lignes : (*longueur* du code) taille des mots-codes ($\omega_1 = [000]^{\mathbf{T}}$, $\omega_2 = [111]^{\mathbf{T}}$) (après codage)

$G = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = [1 \quad 1 \quad 1]^{\mathbf{T}}$: *matrice génératrice* (3x1).

On a bien : $\begin{cases} \omega_1 = [000]^{\mathbf{T}} = \phi(m_1) = Gm_1 = [1 \quad 1 \quad 1]^{\mathbf{T}} \cdot 0 \\ \omega_2 = [111]^{\mathbf{T}} = \phi(m_2) = Gm_2 = [1 \quad 1 \quad 1]^{\mathbf{T}} \cdot 1 \end{cases}$

Exercice 2. Code linéaire.

1. G, **matrice génératrice** du code, est de dimension $n \times k$ = 8x4.

k = 4 colonnes : (*dimension* du code) taille des *mots-sources* (ou blocs d'origine) ($m_i \quad i = 1 \cdots l$) (avant codage)

n = 8 lignes : (*longueur* du code) taille des *mots-codes* ($\omega_i \quad i = 1 \cdots p$) (après codage)

2. G sous forme *systématique* : $\tilde{G} = \begin{pmatrix} I_k \\ \hline G' \end{pmatrix}$ avec : $\begin{cases} I_k : \text{matrice identité } (k \times k) \\ G' : \text{matrice } ((n-k) \times k) \end{cases}$

→ L'obtention de $\tilde{G}$ implique un échelonnage de G suivant ses **colonnes**. Notons les c_1, c_2, c_3, c_4.

$$G = \begin{array}{c} \begin{matrix} c_1 & c_2 & c_3 & c_4 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Il suffit de réaliser les combinaisons linéaires $\begin{cases} c_1 \leftarrow c_1 \\ c_2 \leftarrow c_2 \\ c_3 \leftarrow c_3 \\ c_4 \leftarrow c_1 + c_2 + c_3 + c_4 \end{cases}$ $\rightarrow \tilde{G} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

$$\tilde{G} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \rightarrow \tilde{G} = \begin{pmatrix} I_k \\ \hline G' \end{pmatrix} \quad \text{avec} \quad k = 4 \quad \text{et} \quad G' = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

3. Ensemble de *mots de code* (*image* C du code) ?

L'image du code donne les *mots-codes* ω_i en balayant tous les *mots-sources* m_i avec la relation : $\omega_i = \phi(m_i) = Gm_i$.

On peut obtenir également *la même image du code* sous **forme systématique** : $\omega_i = \phi(m_i) = \tilde{G}m_i$.

Cette dernière relation présente l'avantage d'engendrer les *mots-codes* ω_i avec les *mots-sources* m_i pour *préfixes* :

$$\omega_i^{\mathbf{T}} = (Gm_i)^{\mathbf{T}} = (\tilde{G}m_i)^{\mathbf{T}} = \left[\begin{pmatrix} I_k \\ \hline G' \end{pmatrix} m_i\right]^{\mathbf{T}} = \left(\frac{m_i}{G'm_i}\right)^{\mathbf{T}} = \left[m_i^{\mathbf{T}} \middle| (G'm_i)^{\mathbf{T}}\right]$$

G **et** $\tilde{G}$ **produisent la même image, mais les mots-codes générés ne correspondent pas aux mêmes mots-sources.**

Avec le code construit sous **forme systématique**, on a donc : $\omega_i = \phi(m_i) = \tilde{G}m_i$, on a : $\omega_i^{\mathbf{T}} = \left[m_i^{\mathbf{T}} \middle| (G'm_i)^{\mathbf{T}}\right]$:

- la partie *préfixe* d'un *mot-code* ω_i est le *mot-source* m_i
- la partie *suffixe* d'un *mot-code* ω_i est $G'm_i$:

Ici $G' = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$. En notant $\begin{pmatrix} m_{i_1} \\ m_{i_2} \\ m_{i_3} \\ m_{i_4} \end{pmatrix}$ les 4 bits d'un *mot-source* m_i, le *suffixe* d'un *mot-code* ω_i est donc : $G'm_i = \begin{pmatrix} m_{i_1} + m_{i_3} + m_{i_4} \\ m_{i_2} + m_{i_3} + m_{i_4} \\ m_{i_1} + m_{i_2} + m_{i_4} \\ m_{i_1} + m_{i_2} + m_{i_3} \end{pmatrix}$

Rappel : les sommes se font *modulo* 2.

Table de code ***systématique*** (*image* C du *code* ***systématique***) :

#	*mot source* $m_i^{\mathbf{T}}$	*mot-code* $\omega_i^{\mathbf{T}} = (Gm_i)^{\mathbf{T}} = (\tilde{G}m_i)^{\mathbf{T}} = \left[\begin{pmatrix} I_k \\ \hline G' \end{pmatrix} m_i\right]^{\mathbf{T}} = \left(\frac{m_i}{G'm_i}\right)^{\mathbf{T}} = \left[m_i^{\mathbf{T}} \middle| (G'm_i)^{\mathbf{T}}\right]$	*distance, poids* $d_h(\omega_i, \mathbf{0}) = w(\omega_i)$ (= *nombre de bits* 1 *de* ω_i)
1	0000	0000 0000	0
2	1000	1000 1011	4
3	0100	0100 0111	4
4	0010	0010 1101	4
5	0001	0001 1110	4
6	1100	1100 1100	4
7	1010	1010 0110	4
8	1001	1001 0101	4
9	0101	0101 1001	4
10	0011	0011 0011	4
11	0110	0110 1010	4
12	1110	1110 0001	4
13	1101	1101 0010	4
14	1011	1011 1000	4
15	0111	0111 0100	4
16	1111	1111 1111	8

$\rightarrow d = 4$

4. *Distance minimale* du code :

La *distance minimale d'un code linéaire* est égale au *plus petit poids non nul des mots de code* $\rightarrow\ d = 4$.

Capacité de détection : $e_d = d - 1 \ \rightarrow \ e_d = 3$: *au maximum*, il est **toujours possible** de *détecter* 3 *erreurs*.

Capacité de correction : $e_c = E\left[\dfrac{d-1}{2}\right] \ \rightarrow \ e_c = 1$: *au maximum*, il est **toujours possible** de *corriger* 1 *erreur*.

5. *Matrice de contrôle* du code : d'après la définition de la matrice de contrôle H : $H = \left(G' \mid I_{n-k}\right)$

ici $n \text{ x } k \ = \ 8 \text{ x } 4 \ \rightarrow \ H = \left(G' \mid I_4\right)$ avec $G' = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

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

6. On reçoit les mots suivants. *Sans utiliser la table de mots de code*, comment savoir s'ils sont des *mots-codes* ?

- $m_1 = (0001\,0101)^{\mathbf{T}}$
- $m_2 = (0100\,0111)^{\mathbf{T}}$
- $m_3 = (00111010)^{\mathbf{T}}$

Pour savoir si un mot reçu m est un *mot de code*, on utilise la propriété de la *matrice de contrôle* : on calcule le *syndrome* : $S = Hm$ et on a : $m \in C$ (C : *image* de code) $\Leftrightarrow \ S = Hm = \mathbf{0}$:

$$\begin{matrix} & H & & m_1\ m_2\ m_3 & S_1\ S_2\ S_3 \end{matrix}$$

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

calcul simultané des 3 syndromes

$\rightarrow$

- $S_1 = Hm_1 \neq \mathbf{0} \ \Rightarrow \ m_1 \notin C$
- $S_2 = Hm_2 = \mathbf{0} \ \Rightarrow \ m_2 \in C$ (une vérication consiste en l'examen de la *table de code* (*image* C du code) (ligne 3)
- $S_3 = Hm_3 \neq \mathbf{0} \ \Rightarrow \ m_3 \notin C$

7. En supposant qu'il y a eu *au plus* 1 *erreur de transmission par mot*, est-il possible de *retrouver* les mots de code transmis ?

La *capacité de correction* étant $e_c = 1 \rightarrow$ **oui** : *au maximum*, il est **toujours possible** de **corriger 1 erreur** par mot.

A titre d'exemple, considérons les 3 cas :

- $m_2 = (0100\ 0111)^{\mathbf{T}}$
- $m_1 = (0001\ 0101)^{\mathbf{T}}$
- $m_3 = (00111010)^{\mathbf{T}}$

- $m_2 = (0100\ 0111)^{\mathbf{T}}$

Comme m_2 est un *mot-code* ($S_2 = Hm_2 = \mathbf{0} \Rightarrow m_2 \in C$), il sera décodé directement et par son *préfixe*: $(0100)^{\mathbf{T}}$ (*table C* ligne 3).

En outre, comme il est supposé qu'il y a *au plus* **1 erreur**, alors m_2 est **sans erreur**, car la *distance minimale* du code est **4**; cela veut dire qu'il faut *au minimum* **4 erreurs** pour transformer un *mot du code* en un autre *mot du code*.

- $m_1 = (0001\ 0101)^{\mathbf{T}}$

Comme m_1 n'est pas un *mot-code* ($S_1 = Hm_1 \neq \mathbf{0} = (1011)^{\mathbf{T}} \Rightarrow m_1 \notin C$), on sait qu'il y a **erreur** de transmission.

Décodage & Correction : méthode du syndrome

Soit e_1 le vecteur *erreur* défini par $m_1 = \omega_1 + e_1$ avec $\omega_1 \in C$ (ω_1 issu de la correction de m_1 et tel que $H\omega_1 = 0$).
$\Rightarrow S_1 = Hm_1 = H\omega_1 + He_1 = He_1$: *l'erreur* e_1 a le même *syndrome* que m_1 !

En outre, comme il est supposé qu'il y a *au plus* **1 erreur**, le vecteur e_1 est de la forme : $e_1 = (0\cdots010\cdots0)^{\mathbf{T}}$, où le bit 1 se trouve à la i-ème position sur 8 bits.

$\rightarrow He_1 = c_i$ où c_i représente la i-ème colonne de la matrice H .

$\rightarrow$ s'il y a 1 seule erreur, au i-ème bit, on doit avoir : $c_i = He_1 = Hm_1 = S_1$

$\rightarrow$ 1 des colonnes de la matrice H doit être égale au syndrome $S_1 == (1011)^{\mathbf{T}}$.

On constate en effet que **la colonne c_1 de H est égale au syndrome S_1**: $c_1 = S_1$

$\rightarrow$ le vecteur *erreur* est $e_1 = (10\cdots0)^{\mathbf{T}}$ et le *mot-code* est :

$$\omega_1 = m_1 + e_1 = (0001\ 0101)^{\mathbf{T}} + (1000\ 0000)^{\mathbf{T}} = (1001\ 0101)^{\mathbf{T}}$$

$\rightarrow$ on décode ω_1 (comme *préfixe* de ω_1 ou par la *table de code* C, ligne 8) et on trouve le *mot-source* $(1001)^{\mathbf{T}}$.

Autre méthode de *Décodage & Correction : méthode de décodage au sens de minimum de distance de Hamming*

Le mot reçu m_1 est corrigé en un mot-code ω_i de sorte que la distance de Hamming $d_H(m_1, \omega_i)$ soit minimale :

Table de code ***systématique*** (*image* C du *code* ***systématique***) :

#	*mot source* $m_i^{\mathbf{T}}$	*mot-code* $\omega_i^{\mathbf{T}}$	*distance de Hamming* $d_H(m_1, \omega_i)$ avec $m_1 = (0001\ 0101)^{\mathbf{T}}$
1	0000	0000 0000	3
2	1000	1000 1011	5
3	0100	0100 0111	3
4	0010	0010 1101	3
5	0001	0001 1110	3
6	1100	1100 1100	5
7	1010	1010 0110	5
8	**1001**	**1001 0101**	**1**
9	0101	0101 1001	3
10	0011	0011 0011	3
11	0110	0110 1010	7
12	1110	1110 0001	5
13	1101	1101 0010	5
14	1011	1011 1000	5
15	0111	0111 0100	3
16	1111	1111 1111	5

Le mot-code #8 est à la *distance minimale* (1) de $m_1 \rightarrow m_1$ corrigé = $\omega_1 = (1001\ 0101)^{\mathbf{T}}$ $\rightarrow$ *mot-source* (*décodage*) : $(1001)^{\mathbf{T}}$.

▪ $m_3 = (00111010)^{\mathbf{T}}$

Comme m_3 n'est pas un *mot-code* ($S_3 = Hm_3 \neq \mathbf{0} = (1001)^{\mathbf{T}} \Rightarrow m_3 \notin C$), on sait qu'il y a **erreur** de transmission.

Décodage & Correction : méthode du syndrome

Soit e_3 le vecteur *erreur* défini par $m_3 = \omega_3 + e_3$ avec $\omega_3 \in C$ (ω_3 issu de la correction de m_3 et tel que $H\omega_3 = 0$).

$\Rightarrow S_3 = Hm_3 = H\omega_3 + He_3 = He_3$: *l'erreur* e_3 a le même *syndrome* que m_3 !

En outre, comme il est supposé qu'il y a *au plus* **1 erreur**, le vecteur e_3 est de la forme : $e_3 = (0\cdots010\cdots0)^{\mathbf{T}}$, où le bit 1 se trouve à la *i*-ème position sur 8 bits.

→ $He_3 = c_i$ où c_i représente la *i*-ème colonne de la matrice H .

→ s'il y a 1 seule erreur, au *i*-ème bit, on doit avoir : $c_i = He_3 = Hm_3 = S_3$

→ 1 des colonnes de la matrice H doit être égale au syndrome $S_3 = (1001)^{\mathbf{T}}$.

On constate qu'**aucune colonne de H n'est égale au syndrome S_3** → il y a donc **2 erreurs ou plus** !

La *capacité de correction* étant $e_c = 1$ → **au maximum**, il est *toujours possible* de **corriger 1 erreur** par mot.

→ **la correction à coup sûr de plus d'1 erreur n'est pas possible**.

Pour néanmoins essayer de trouver les erreurs, on ne peut que tester des *hypothèses de* 2, 3, 4 … *erreurs*.

• Par exemple, à partir de *l'hypothèse de 2 erreurs*, il doit donc exister $i \neq j$ tel que $c_i + c_j = He_3 = Hm_3 = S_3$

→ plusieurs vecteurs *erreurs* e_3 candidats (e_3 **non unique**) → après correction, plusieurs *mots-codes* $\omega_3 \in C$ candidats :

- ♦ $c_5 + c_8 = (1000)^{\mathbf{T}} + (0001)^{\mathbf{T}} = (1001)^{\mathbf{T}} \rightarrow e_3 = (0000\,1001)^{\mathbf{T}} \rightarrow \omega_3 = m_3 + e_3 = (00111010)^{\mathbf{T}} + (0000\,1001)^{\mathbf{T}} = (0011\,0011)^{\mathbf{T}}$ (*C* ligne 10)
- ♦ $c_1 + c_7 = (1011)^{\mathbf{T}} + (0010)^{\mathbf{T}} = (1001)^{\mathbf{T}} \rightarrow e_3 = (1000\,0010)^{\mathbf{T}} \rightarrow \omega_3 = m_3 + e_3 = (00111010)^{\mathbf{T}} + (1000\,0010)^{\mathbf{T}} = (10111000)^{\mathbf{T}}$ (*C* ligne 14)
- ♦ $c_3 + c_6 = (1101)^{\mathbf{T}} + (0100)^{\mathbf{T}} = (1001)^{\mathbf{T}} \rightarrow e_3 = (0010\,0100)^{\mathbf{T}} \rightarrow \omega_3 = m_3 + e_3 = (00111010)^{\mathbf{T}} + (0010\,0100)^{\mathbf{T}} = (00011110)^{\mathbf{T}}$ (*C* ligne 5)

Rappel : La *capacité de correction* étant $e_c = 1 \rightarrow$ **au maximum**, il est *toujours possible* de **corriger 1 erreur** par mot.

→ **la correction à coup sûr de plus d'1 erreur n'est pas possible → on peut ne pas pouvoir corriger**.

_____