Combinatoire du codage convolutif

1 Eléments de base

Soit $\mathbb{F}_2$ l'alphabet binaire muni d'une structure d'anneau. Etant donnée une suite binaire finie m, on note m_i son $(i+1)^{\text{e}}$ terme et $|m|$ sa longueur. A toute suite binaire m, on peut associer un polynôme $m(x)$ d'une indéterminée sur l'anneau des polynômes $\mathbb{F}_2[x]$ par une bijection θ :

$$m_0, m_1, m_2, \ldots, m_k \quad \overset{\theta}{\mapsto} \quad m_0 + m_1.x + m_2.x^2 + \ldots + m_k.x^k$$

Exemple : A la suite $m = 1011$, on associe le polynôme $1 + x^2 + x^3$ et on notera $1011 \leftrightarrow 1 + x^2 + x^3$.

1.1 Opérations sur les suites

L'addition de deux suites binaires est la suite définie par l'addition terme à terme modulo deux.
On s'intéresse à l'opération de *convolution* entre deux suites binaires a et m notée $a \star m$:

$$\{a \star m\}_{i \geq 0} = \left\{ \sum_{j=0}^{i} a_j m_{i-j} \right\}_{i \geq 0}$$

Question 1 : Montrer que la suite $\theta^{-1}(1)$ est élément neutre du produit de convolution.
Corrigé : $(a \star 1)_i = \sum_{0 \leq j \leq |a|-1} a_j 1$ qui n'est définie que lorsque $i = j$, i.e. $(a \star 1)_i = a_i$. Comme on ne connait pas la commutativité de $\star$, il faudrait montrer de manière analogue que $1 \star a = a$

Question 2 : Montrer que l'opération de convolution correspond au produit des polynômes.
Corrigé : A la suite binaire a de longueur $k+1$, on associe le polynôme $a(x) = a_0 + a_1 x + \ldots + a_k x^k$ et à la suite binaire m de longueur $n+1$, le polynôme $m(x) = m_0 + m_1 x + \ldots m_n x^n$. Le polynôme $p(x) = a(x)m(x) = a_0 m_0 + \ldots + (\sum_{j=0}^{k} a_j m_{i-j}) x^i + \ldots + (\sum_{j=0}^{k} a_j m_{k+n-j}) x^{k+n}$ est de degré $k+n$ et correspond à la suite binaire (finie) $a \star m$. Réciproquement, à la suite $a \star m$, on associe le polynôme $p(x)$.

Question 3 : Montrer que l'opération de convolution est associative, commutative et qu'elle est distributive par rapport à l'addition.
Corrigé :
- $\left\{\sum_{j=0}^{i} a_j b_{i-j}\right\}_{i\geq 0} \star \{c\}_k = \{a\}_k \star \left\{\sum_{j=0}^{i} b_j c_{i-j}\right\}_{i\geq 0} =$
  $\{a_0b_0c_0, (a_1b_0 + a_0b_1)c_0 + a_0b_0c_1,$
  $(a_2b_0 + a_1b_1 + a_0b_2)c_0 + (a_1b_0 + a_0b_1)c_1 + a_0b_0c_1, \dots\} =$
  $\{a_0b_0c_0, a_1b_0c_0 + a_0(b_1c_0 + b_0c_1),$
  $a_2b_0c_0 + a_1(b_1c_0 + b_0c_1) + a_2(b_2c_0 + b_1c_1 + b_0c_2), \dots\}$
- $a \star b = b \star a = \{a_0b_0, a_1b_0 + a_0b_1, \dots\} = \{b_0a_0, b_1a_0 + b_0a_1, \dots\}$
- $a \star (b + c) = a \star b + a \star c = \{a_0(b_0 + c_0), a_1(b_0 + c_0) + a_0(b_1 + c_1), \dots\} =$
  $\{a_0b_0, a_1b_0 + a_0b_1, \dots\} + \{a_0c_0, a_1c_0 + a_0c_1, \dots\} =$
  $\{a_0(b_0 + c_0), a_1(b_0 + c_0) + a_0(b_1 + c_1), \dots\}$

2 Codage

On définit un *code convolutif* C au moyen de ses deux *polynômes générateurs* $p(x)$ et $q(x) \in \mathbb{F}_2[x]$ qui seront supposés premiers entre eux (autrement dit, il existe deux polynômes $u(x)$ et $v(x)$ tels que $p(x).u(x) + q(x).v(x) = 1$).

2.1 Codage d'un message

Au moyen des deux polynômes générateurs $p(x)$ et $q(x)$, on peut coder le *message* $m(x) \in \mathbb{F}_2[x]$ en $c(m(x)) \in \mathbb{F}_2[x] \times \mathbb{F}_2[x]$; $c(m(x))$ s'appelle le *message codé*. L'opération de codage est :

$$c(m(x)) = (p(x)m(x), q(x)m(x))$$

Exemple : Avec $p(x) = 1 + x^2$ et $q(x) = 1 + x + x^2$ comme polynômes générateurs, le message $m(x) = 1+x^2+x^3$ sera codé en $c(m(x)) = (1+x^3+x^4+x^5, 1+x+x^5)$. Ou, exprimé en terme de suites, le message $m = 1011$ sera codé en $c = (100111, 110001)$.

De cette manière, à chaque bit du message, on associera un couple de bits dans le message codé.

Dans la suite du sujet, on illustrera les différentes notions au moyen du code convolutif engendré par les deux polynômes générateurs précédents : $p(x) = 1 + x^2$ et $q(x) = 1 + x + x^2$.

Question 4 : Codez les messages :
- $m(x) = 1 + x^2$
- $m'(x) = 1 + x + x^2 + x^3 + \ldots + x^k = \sum_{i=0}^{k} x^i$

Corrigé :
- $((1 + x^2)(1 + x^2), (1 + x + x^2)(1 + x^2)) = (1 + x^4, 1 + x + x^3 + x^4)$
- $((1 + x^2)(\sum_{i\geq 0} x^i), (1 + x + x^2)(\sum_{i=0}^{k} x^i)) = (1 + x + x^{k+1} + x^{k+2}, 1 + (\sum_{i=2}^{k} x^i) + x^{k+2})$

2.2 Ensemble des messages codés

L'ensemble de tous les messages codés C est donné par :

$$C = \{c(m(x)) = (p(x)m(x), q(x)m(x)), m(x) \in \mathbb{F}_2[x]\}$$

Un élément de C sera appelé un *mot du code*.

Question 5 : Montrer que C est un code *linéaire*. En d'autres termes, que la somme (composante par composante) de deux mots distincts du code est un mot du code.
Corrigé : $c(m(x)) + c(m'(x)) = (p(x)m(x), q(x)m(x)) + (p(x)m'(x), q(x)m'(x)) = (p(x)(m(x)+m'(x), q(x)(m(x)+m'(x))$ est le mot du code correspondant au message $m(x)+m'(x)$.

Pour transmettre un mot du code $c(m(x)) \in \mathbb{F}_2[x] \times \mathbb{F}_2[x]$ au travers d'un canal (satellitaire, téléphonique), on convertit les polynômes obtenus en suites binaires. Au lieu d'émettre les deux suites binaires correspondant à $c(m(x))$, on transforme $c(m(x)) = (c^p(x) = \sum_{i=0}^{k} c_i^p x^i, c^q(x) = \sum_{i=0}^{k} c_i^q x^i)$ en une unique suite binaire c par *entrelacement* : on envoie successivement les premiers termes $c_0^p c_0^q$, puis les seconds termes $c_1^p c_1^q$, les troisièmes $c_2^p c_2^q$, etc. De cette manière, à chacun des bits de la suite binaire $m_0, m_1, m_2, \ldots$ correspondant au message m, on peut associer les deux bits correspondants de la suite binaire entrelacée du mot du code $c(m(x))$. Coder devient une opération sur les suites binaires dont nous allons découvrir quelques propriétés. Ainsi, à chaque bit m_i du message, on associera un couple de bits $c_{2i}c_{2i+1}$ dans le message codé.

Exemple : Le message $m(x) = 1+x^2+x^3$ est codé en $c(m(x)) = (1+x^3+x^4+x^5, 1+x+x^5)$. Ou, exprimé en terme de suites, le message $m = 1011$ sera codé en $c = (100111, 110001)$. Après entrelacement, on aura : 11 01 00 10 10 11

Question 6 : Quelle est la suite binaire entrelacée émise pour chacun des messages suivants :
- $m_1(x) = 1 + x^2 \leftrightarrow 101$
- $m_2(x) = 1 + x + x^2 + x^3 + \ldots + x^k = \sum_{i=0}^{k} x^i \leftrightarrow 11\ldots1$

Corrigé :
- 11 01 00 01 11
- 11 10 $(01)^{k-1}$ 10 11

Question 7 : Montrer comment calculer $c_{2i}c_{2i+1}$ en fonction de $m_i, m_{i-1}, \ldots, m_{i-k}$ pour k le plus grand degré des polynômes générateurs. En déduire que le codage peut se faire bit à bit avec une mémoire bornée.
Corrigé :

m_0	$c_0 = m_0 p_0$	$c_1 = m_0 q_0$
m_1	$c_2 = m_1 p_0 + m_0 p_1$	$c_3 = m_1 q_0 + m_0 q_1$
m_2	$c_4 = m_2 p_0 + m_1 p_1 + m_0 p_2$	$c_5 = m_2 q_0 + m_1 q_1 + m_0 q_2$
$\vdots$		
m_k	$c_{2k} = m_k p_0 + \ldots + m_0 p_k$	$c_{2k+1} = m_k q_0 + \ldots + m_0 q_k$
m_{k+1}	$c_{2k+2} = m_{k+1} p_0 + \ldots + m_1 p_k$	$c_{2k+3} = m_{k+1} q_0 + \ldots + m_1 q_k$
	car $p_j = 0$ pour $j \geq k+1$	car $q_j = 0$ pour $j \geq k+1$

Le codage peut donc se faire avec une mémoire d'au plus $k+1$ bits.

2.3 Arbre de codage

Comme à chaque bit de la suite binaire du message (l'entrée) on peut associer les deux bits correspondants de la suite binaire entrelacée du mot du code (la sortie), on peut construire un arbre binaire potentiellement infini appelé *arbre de codage* dont les arêtes sont étiquetées par les éléments de $(\mathbb{F}_2)^2$ de la sortie. Dans cet arbre, l'entrée permet de sélectionner une branche et la sortie est déterminée par les étiquettes des arêtes de la branche sélectionnée. Une entrée $m_i = 0$ prolonge le chemin sur l'arête de gauche et une entrée $m_i = 1$ sur celle de droite. Ainsi, pour l'entrée 1011, la branche choisie sera définie depuis la racine par l'arête de droite puis l'arête de gauche puis l'arête de droite puis l'arête de droite (cf. figure 1).

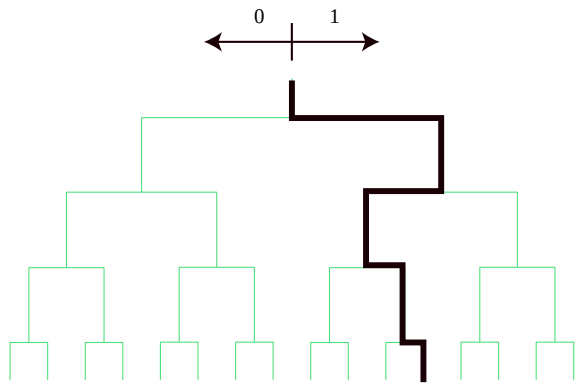


FIG. 1 – Parcours de l'arbre de codage pour l'entrée 1011

Question 8 : Construire l'arbre de codage du code engendré par $p(x) = 1 + x^2$ et $q(x) = 1 + x + x^2$ jusqu'à une profondeur de 4.
Corrigé : (cf figure 2.)

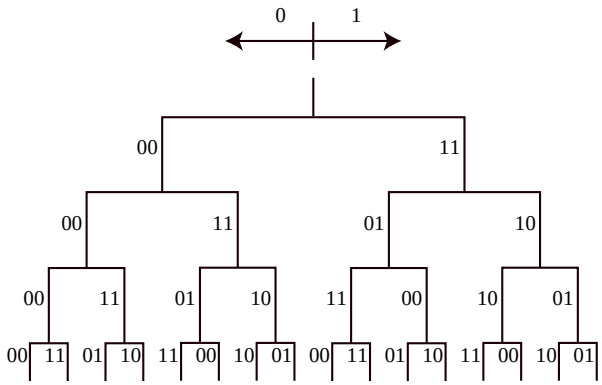


FIG. 2 – Corrigé de la question 8.

Du fait que le codage peut se faire bit à bit avec une mémoire bornée (cf. question 7), il est possible d'engendrer de manière finie l'arbre de codage qui est potentiellement infini.

Autrement dit, dans l'arbre de codage, la connaissance de N niveaux permet de construire le niveau $N+1$.
Remarquons que le nombre de sous-arbres possibles de hauteur N est borné.

Question 9 : Donnez un majorant du nombre de sous-arbres possibles et proposez une représentation finie de l'arbre.
Corrigé : Pour la borne sur le nombre de sous-arbres possibles : on dispose de 4 étiquettes distinctes et l'étiquette du fils gauche doit être distincte de celle du fils droit, pour une profondeur de 1, on a 4.3=12 sous-arbres possibles. Lorsqu'on fait croître la profondeur à p, on obtient que le nombre de sous arbres est majoré par 12^{2^p-1}. Une représentation finie possible est la suivante : on construit un tableau indicé par le numéro des sous-arbres et on indique pour chacun quel est le numéro de son sous-arbre gauche et celui de son sous-arbre droit. Pour décrire l'arbre, il suffit alors de donner le numéro du sous-arbre gauche et du sous-arbre droit de la racine. Pratiquement, si on fait un parcours en largeur de notre arbre, on constate que le nombre de numéros peut être très inférieur à celui donné par la borne.

2.4 Diagramme de transition d'états

Du fait que l'arbre de codage admet une représentation finie, il est possible de construire un *diagramme de transition d'états* ou *codeur* qui va fonctionner à la manière d'une machine de Mealy et permettre le codage d'un message fourni en entrée (On rappelle qu'une machine de Mealy est un automate fini muni d'une fonction de sortie sur les transitions). Ce diagramme n'est autre qu'un graphe orienté dont les arcs sont étiquetés par un couple à valeurs dans $\mathbb{F}_2 \times (\mathbb{F}_2)^2$. La première composante de ce couple représente le symbole lu sur la suite fournie en entrée (le message) et la seconde composante le codage de ce symbole, c'est à dire la sortie. De cette manière, le codage d'un message m pourra être effectué «à la volée».
Chaque «état» (ou sommet du graphe) du codeur va mémoriser les k derniers bits lus sur l'entrée avec k le plus grand degré des polynômes générateurs. Puis, en fonction du nouveau bit lu sur l'entrée (de la gauche vers la droite), va
- changer d'état ;
- renvoyer la sortie de deux bits correspondants à la suite binaire entrelacée du mot du code pour le bit lu sur l'entrée.

Question 10 : Construire le diagramme de transition d'états pour le code engendré par $p(x) = 1 + x^2$ et $q(x) = 1 + x + x^2$. (On pourra utiliser le résultat de la question 8).
Corrigé : voir figure 3.

Question 11 : Montrer qu'en faisant abstraction des étiquettes (bit lu et bits de sortie), le diagramme de transition des états de tout code convolutif est un graphe de de Bruijn $\mathcal{B}(2,k)$, c'est à dire le graphe orienté dont les sommets sont les mots de longueur k sur l'alphabet $\{0,1\}$ et où les voisins du sommet étiqueté $s_0 s_1 \dots s_{k-1}$ sont les sommets $s_1 \dots s_{k-1}\sigma$ pour $\sigma \in \{0,1\}$.
Corrigé : Soit k le plus grand degré de $p(x)$ et $q(x)$. Chaque état du diagramme de transition d'états est une suite sur $(\mathbb{Z}_2)^k$. Il comporte bien 2^k sommets et on relie le sommet $s_0 s_1 \dots s_{k-1}$ à chacun des sommet $s_1 \dots s_{k-1}0$ et $s_1 \dots s_{k-1}1$. En faisant abstraction des étiquettes, le

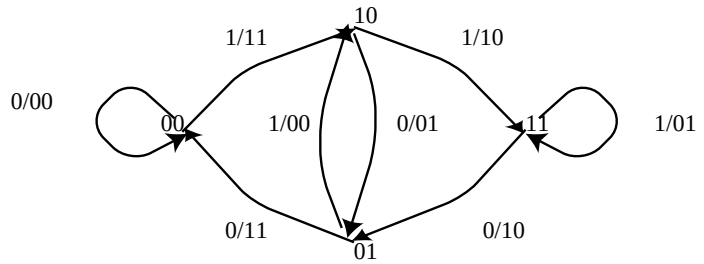


FIG. 3 – Diagramme de transition d'états

diagramme de transition d'états est bien isomorphe à $\mathcal{B}(2,k)$, à une renumérotation des états près.

Coder revient à prendre en entrée le message bit après bit et à «retourner» la sortie étiquetée sur les arcs du graphe. Cependant, pour que la sortie corresponde à la définition donnée par les polynômes, il faut ajouter à la fin de l'entrée autant de zéros que le degré du plus grand des polynômes générateurs.

Question 12 : Au moyen de votre diagramme de transition des états précédent, coder le message $m_1(x) = 1 + x^2 \leftrightarrow 101$.
Corrigé : 11 01 00 01 11

3 Capacité de correction

Le but du codage convolutif (ainsi que celui des autres codes correcteurs d'erreurs) est de permettre la correction des messages codés lorsque ceux-ci traversent un canal (satellitaire, téléphonique) qui peut ajouter des erreurs. Le destinataire reçoit une suite binaire r qui est la somme de la suite binaire correspondant au mot du code émis c et de la suite binaire correspondant à l'erreur e :

$$r = c + e$$

Il doit alors *décoder* r pour tenter de retrouver c.
Pour ce faire le destinataire résout le problème d'optimisation suivant : étant donnée r, trouver la suite binaire $\hat{c}$ qui appartient au code telle que la distance entre r et $\hat{c}$ soit aussi petite que possible. Si cette distance est nulle, cela signifie que le message codé n'a pas été modifié. Ce procédé se nomme *décodage à distance minimum*.
La distance qui est utilisée est la *distance de Hamming* que l'on définit pour deux suites binaires x et y de longueur n comme le nombre de positions où x et y diffèrent terme à terme :

$$d_H(x,y) = \text{Card}\{i : x_i \neq y_i, 0 \leq i \leq (n-1)\}$$

Question 13 : Vérifier que la distance de Hamming vérifie les axiomes d'une distance :

$$d_H(x,y) \geq 0; \quad d_H(x,y) = 0 \Leftrightarrow x = y; \quad d_H(x,y) = d_H(y,x); \quad d_H(x,y) \leq d_H(x,z) + d_H(z,y)$$

Corrigé : Le seul point difficile est de vérifier l'inégalité triangulaire. Sans perte de généralité, on vérifie que $d_H(0,x) \leq d_H(0,y) + d_H(x,y)$ avec 0 la suite 0^n, $x = 1^e 0^{n-e}$ et $y = 0^a 1^b 0^c 1^d$ avec $a+b=e, c+d=n-e$. Alors $d_H(0,x) = e$, $d_H(0,y) = b+d$ et $d_H(x,y) = a+d$ On a donc $e = a+b \leq b+d+a+d$.

On définit également le *poids* d'une suite binaire x comme sa distance de Hamming à la suite binaire identiquement nulle.

$$\mathbf{w}(x) = d_H(0\ldots0, x)$$

Nous allons maintenant nous intéresser au poids maximum de la suite d'erreurs qu'un code convolutif peut corriger.

3.1 Distance libre du code

On considère tout d'abord la *distance libre* du code convolutif C, notée $d(C)$. Celle-ci, par définition, mesure la plus petite distance entre deux mots du code distincts.

$$d(C) = \min_{c,c' \in C, c \neq c'} \{d_H(c,c')\}$$

Question 14 : Montrer qu'une définition alternative de la distance libre est donnée par le poids :

$$d(C) = \min_{c \in C, c \neq 0} \{\mathbf{w}(c)\}$$

Corrigé : Par la linéarité du code, si $c, c' \in C$, $c + c' \in C$ et $d_H(c,c') = d_H(0\ldots0, c+c') = \mathbf{w}(c+c')$

La distance libre est le principal paramètre pour déterminer la capacité de correction du code.

Question 15 : Si on note $\mathcal{E}_t$ l'ensemble de toutes les suites d'erreur de poids inférieur ou égal à t, montrer qu'un code convolutif C peut corriger tous les éléments de $\mathcal{E}_t$ si et seulement si $d(C) > 2t$.

Corrigé : Supposons $d(C) > 2t$. On suppose avoir reçu la suite binaire r alors que le mot du code c a été émis. On décode r en $\hat{c} \in C$ le plus proche mot du code de r pour la distance de Hamming. Si $r = c + e$ pour $e \in \mathcal{E}_t$, alors $d_H(r,c) \leq t$. Le mot du code $\hat{c}$ satisfait aussi $d_H(r,\hat{c}) \leq t$. D'où $d_H(c,\hat{c}) \leq d_H(r,c) + d_H(\hat{r},c) \leq 2t < d(C)$, et $c = \hat{c}$.

Réciproquement, supposons que $d(C) \leq 2t$. Soient c et c' deux éléments de C tels que $d_H(c,c') = d(C)$. On définit r comme suit :

$$r_i = \begin{cases} c_i = c'_i & \text{pour tous les } i \text{ tels que } c_i = c'_i \\ c'_i & \text{sur les } t \text{ 1}^{\text{res}} \text{ positions pour lesquelles } c_i \neq c'_i \text{ si } t < d(C) \\ & \text{toutes les positions pour lesquelles } c_i \neq c'_i \text{ sinon} \\ c_i & \text{sur les } d(C) - t \text{ positions restantes} \end{cases}$$

On a $d_H(c,r) = t$ et $d_H(c',r) = d(C) - t$ si $t \leq d(C)$ ou $d_H(c,r) = d(C)$ et $d_H(c',r) = 0$ sinon. On a deux erreurs e et e' de $\mathcal{E}_t$ qui satisfont $r = c + e = c' + e'$ mais on ne sait pas comment décoder r ; une contradiction.

3.2 Calcul de la distance libre

Question 16 : Donner un algorithme qui calcule la distance libre d'un code convolutif à partir du diagramme de transition des états.
Indication : On pourra adapter un algorithme de graphes pour le calcul des distances minimales entre toute paire de sommets.
Corrigé : On utilise l'algorithme de Warshall. La matrice d'incidence d du graphe est initialisée avec le poids des étiquettes de la fonction de sortie, sauf pour la boucle sur $0\ldots0$ quand une transition existe et ∞ pour les autres. La fonction de mise à jour devient simplement $d(i,j) = \min\{d(i,j), d(i,k) + d(k,j)\}$.

Question 17 : Calculer la distance libre du code de la question 10 ; en déduire le poids maximum des erreurs.
Corrigé : Une application rapide de l'algorithme donne à l'avant-dernière itération la valeur de 5. Ce code peut donc corriger des erreurs de poids au plus 2.

3.3 Calcul de la taille de fenêtre de décodage

Il reste maintenant à déterminer le nombre de bits de la suite qu'il va falloir conserver en mémoire afin de mener à bien le décodage à distance minimum. En d'autres termes, quelle va être la taille ℓ (en bits) de la fenêtre qui va contenir une sous-suite extraite de la suite correspondant au message reçu r et qui peut contenir une erreur ?
Pour assurer le décodage, il suffit de considérer sur le diagramme de transition des états la longueur des cycles ayant pour origine le sommet $0\ldots0$ autres que le cycle de longueur 1 (qui correspond à la boucle sur l'état $0\ldots0$). On pourra décoder convenablement dès lors que le poids des étiquettes de la fonction de sortie de l'un des cycles sera strictement supérieur à $2t$ (t correspond au poids de l'erreur). On prend la longueur ρ du plus court cycle (en terme de nombre d'arcs) qui vérifie cette propriété et la taille $\ell = 2\rho$ puisque la fonction de sortie renvoie deux bits par arc.

Question 18 : Modifier l'algorithme de calcul de la distance libre de la question 16 pour qu'il fournisse également ρ et calculer la valeur de ℓ pour le code défini à la question 10.
Corrigé : Il suffit de prendre la valeur de l'indice de boucle k dès que $d(0,0) > 2t$. Appliqué au cas particulier, la première affectation de $d(0,0)$ est convenable, aussi $\rho = 2$ et $\ell = 2\rho = 4$.

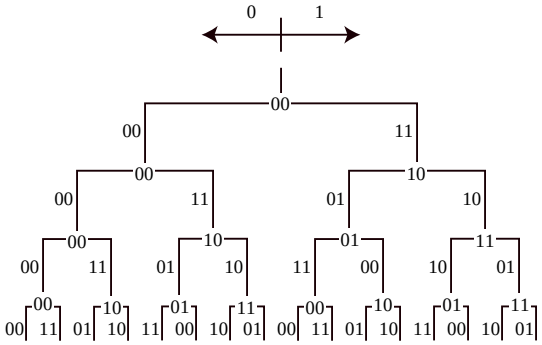
4 Décodage

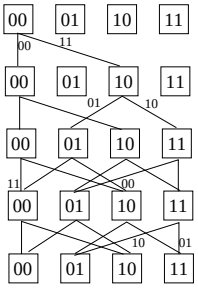
Afin de décoder plus facilement qu'en cherchant parmi tous les mots du code celui qui est le plus proche de la suite binaire correspondant au message reçu, on construit un *treillis*. Pour ce faire, on reprend l'arbre de codage défini en 2.3 et

- on étiquette les sommets de l'arbre par les états correspondants du diagramme de transition des états.
- on identifie les sommets de l'arbre qui portent la même étiquette lorsqu'ils sont à la même profondeur en conservant les arêtes et les étiquettes.

De cette manière, la suite des étiquettes des arêtes d'un chemin dans le treillis correspond à un mot du code.

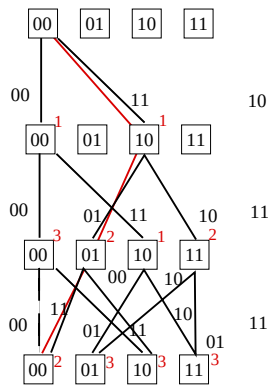
Question 19 : Construire le treillis correspondant à l'arbre de codage de la question 8.
Corrigé :





Question 20 : Expliquer comment utiliser le treillis précédent pour faire le décodage à distance minimum d'un message reçu.
Corrigé : On construit un mot du code c qui minimise la distance de Hamming entre r et c. On se déplace dans le treillis de haut en bas en effaçant tous les sous-chemins qui ne peuvent être préfixe du meilleur chemin dans le treillis. Pour ce faire, on calcule la distance de Hamming entre r et le c correspondant au chemin que l'on est en train de construire. Quand on atteint le niveau 2, on dispose de 4 sous-chemins –un par état du diagramme de transition. Au niveau suivant, on dispose 8 sous chemins –deux par état du diagramme de transition. Pour chacun des états du diagramme de transition de ce niveau, on ne conserve qu'un seul chemin entrant–celui qui est le plus proche de r en calculant la distance de Hamming entre chaque sous-chemin et r. On efface les autres car ils ne peuvent mener à un mot du code. On continue de cette manière jusqu'à arriver à la fin du mot reçu r à l'état 00, pour lequel on ne dispose que d'un chemin entrant. Il suffit alors de remonter les niveaux pour obtenir le mot du code le plus proche du mot reçu.

Question 21 : Appliquer votre résultat au décodage du message codé 10 11 11 au moyen du treillis construit à la question 18.
Corrigé :


00
01
10
11
00
11
10
00
01
10
11
00
01
11
10
11
00
01
10
11
00
11
10
10
00
01
11
11
01
00
01
10
11