Réseaux
Couche Liaison

E. Jeandel

Emmanuel.Jeandel at univ-lorraine.fr

Résumé des épisodes précédents

La couche physique nous permet une transmission brute de bits par un medium reliant une machine A à une machine B.

La couche liaison

Transmission de *trames* d'un point A à un point B relié par un medium

Plusieurs problématiques :

- Découpage en trames
- Detection/Correction des erreurs
- Partage du medium

Dans ce cours : les deux premières

Contenu

Trames

Une donnée échangée au niveau liaison s'appelle une *trame*.

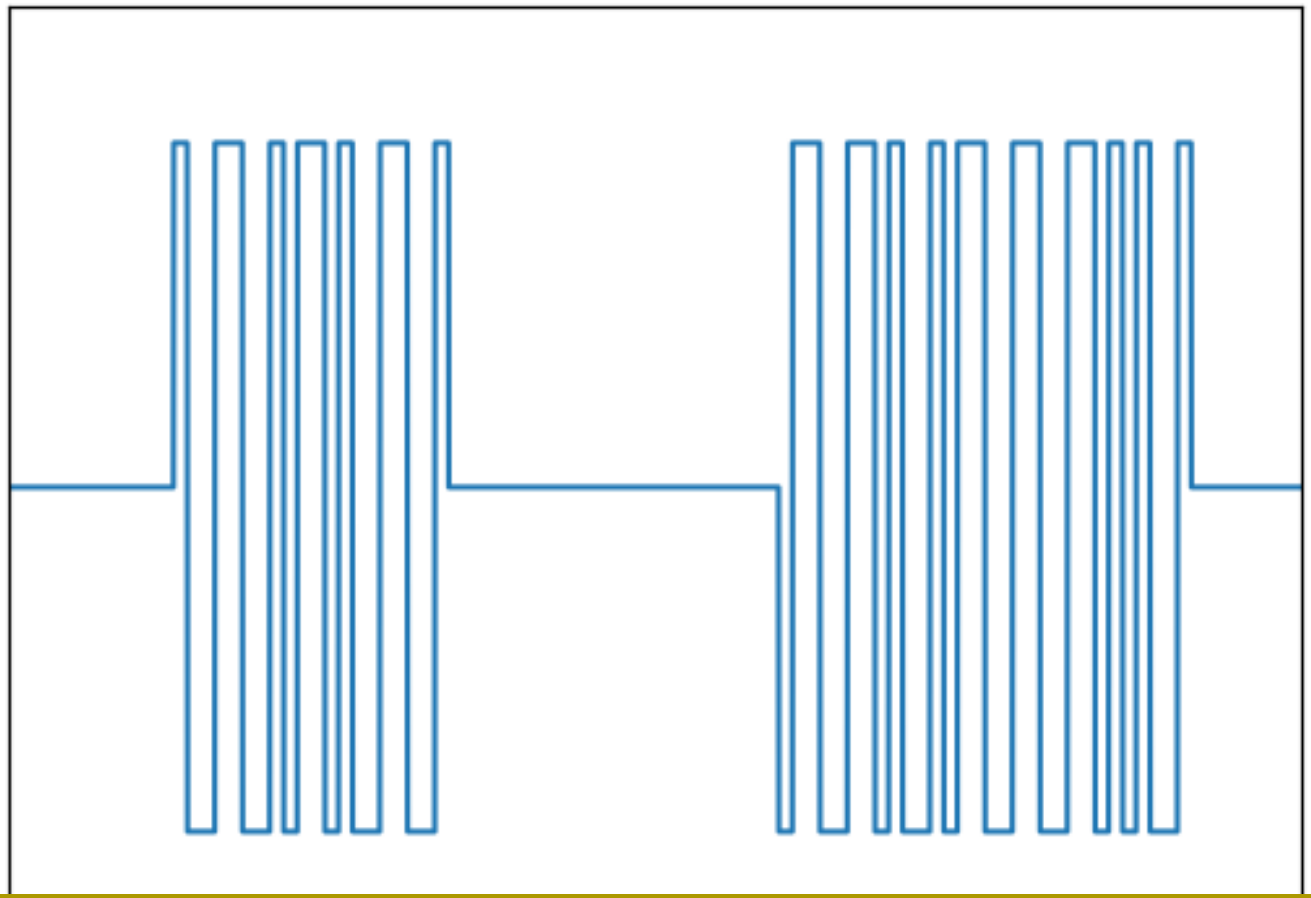
Problème : comment délimiter les trames ?

Comment "séparer" les messages ?

Solution 1

Reporter le problème à la couche physique

Exemple : Ethernet avec codage Manchester.



Ethernet

IEEE 802.3

Ethernet :

- Le récepteur se réveille quand il voit que quelque chose se passe
- Il peut rater le début

Solution

- 7 octets 10101010
- 1 octet 10101011

Avantages :

- Pas grave si on rate le début
- Permet de synchroniser l'horloge

Cas général

Si la couche physique fournit une suite ininterrompue de 0 et de 1...

- Il faut indiquer le début et la fin de chaque trame

Codage de début et de fin

Mode octet

P	l	a	t	y	p	u	s

#	P	l	a	t	y	p	u	s	#

On repère le début (resp. fin) de la trame par le caractère #.

Codage de début et de fin

Mode octet

R	o	o	m	:	#	1	3	4	9

#	R	o	o	m	:	#	1	3	4	9	#

Codage de début et de fin

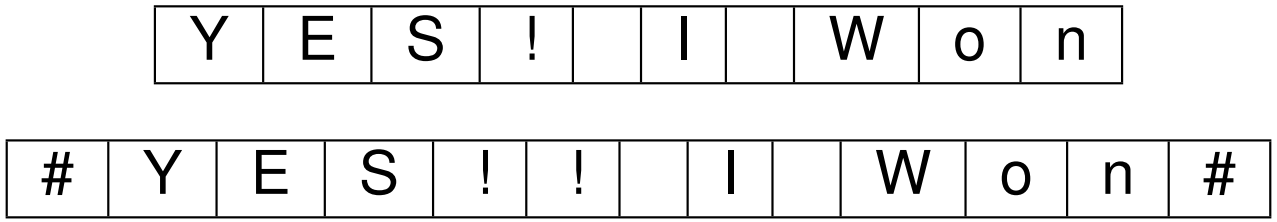
Mode octet

R	o	o	m	:	#	1	3	4	9

#	R	o	o	m	:	!	#	1	3	4	9	#

Codage de début et de fin

Mode octet



Codage de début et de fin

HDLC/PPP (RFC 1662)

- On code le début et la fin par le fanion `01111110`
- Quand il y a deux messages consécutifs, on a un seul `01111110` au milieu
- Dans tout le texte, une suite de 5 caractères `1` consécutifs est remplacée par `111110`

Contenu

Problématique

- La transmission dans le medium n'est pas parfaite.
- La couche physique n'offre aucune garantie.

Exemple

- Cable Ethernet : proba d'erreur 10^{-6} : 1 bit sur un million
- Fibre optique : proba d'erreur 10^{-14}
- Wifi : proba d'erreur 10^{-5}
- Pigeons : proba d'erreur 0.55

Erreurs

Plusieurs types d'erreurs possibles :

- Message (ou bits) perdu

1011011001

101101001

- Message (ou bits) ajouté/dupliqué
- Message (ou bits) illisible
- Message (ou bits) erroné
- Tout en même temps

En pratique, on se concentrera sur les messages erronés.

Erreurs

Plusieurs types d'erreurs possibles :

- Message (ou bits) perdu
- Message (ou bits) ajouté/dupliqué

1011011001

10101011001

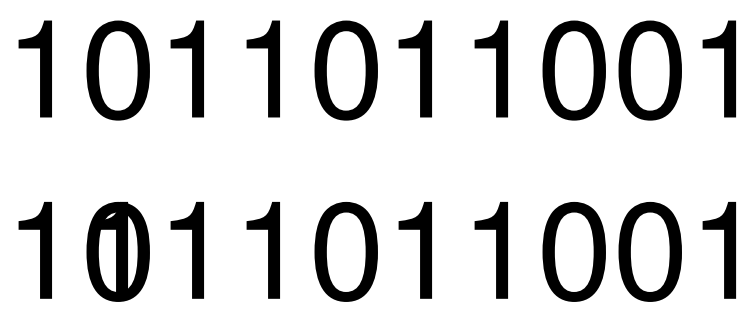
- Message (ou bits) illisible
- Message (ou bits) erroné
- Tout en même temps

En pratique, on se concentrera sur les messages erronés.

Erreurs

Plusieurs types d'erreurs possibles :

- Message (ou bits) perdu
- Message (ou bits) ajouté/dupliqué
- Message (ou bits) illisible



- Message (ou bits) erroné
- Tout en même temps

En pratique, on se concentrera sur les messages erronés.

Erreurs

Plusieurs types d'erreurs possibles :

- Message (ou bits) perdu
- Message (ou bits) ajouté/dupliqué
- Message (ou bits) illisible
- Message (ou bits) erroné

$$10110111001$$

$$10010111001$$

- Tout en même temps

En pratique, on se concentrera sur les messages erronés.

Erreurs

Plusieurs types d'erreurs possibles :

- Message (ou bits) perdu
- Message (ou bits) ajouté/dupliqué
- Message (ou bits) illisible
- Message (ou bits) erroné
- Tout en même temps

1011011001

1001011011̸0

En pratique, on se concentrera sur les messages erronés.

Erreurs

Que faut-il faire ?

- Coder le message de façon à s'apercevoir qu'il y a une erreur
- ...éventuellement la corriger
- ...ou redemander/détruire le message

Les protocoles pour redemander les messages sont très compliqués : on les verra en L3.

Attention

! Attention !

Un code qui corrige trois erreurs ne sert à rien si le medium produit 15 erreurs en moyenne.

! Attention !

On doit partir d'un codage physique qui produit peu d'erreurs

Code

Définition

Un code de paramètre (n, m) *est un procédé qui transforme n bits en* $m > n$ *bits. On note* $f(x)$ *la fonction (le code)*

Les codes s'obtiennent en général en *rajoutant* des bits, mais ce n'est pas toujours le cas : on peut aussi modifier ceux qui sont là.
(Ex déjà vu : le code 4 bits vers 5 bits)

Erreurs

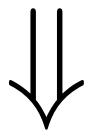
- Un mot x est transmis avec k erreurs si le résultat de la transmission contient k bits différents de leur valeur initiale.
- Un code *détecte* k erreurs si, lorsqu'un mot $f(x)$ est transmis avec moins de k erreurs, on peut se rendre compte que le message est erroné.
- Un code *corrige* k erreurs si, lorsqu'un mot $f(x)$ est transmis avec moins de k erreurs, on est capable de retrouver $f(x)$.

Premier exemple

Doubler les bits

Code (4,8) qui duplique tous les bits :

1011



11001111

Premier exemple

Doubler les bits

Code (4,8) qui duplique tous les bits :

Peut-on détecter 1 erreur ?

00110111

Premier exemple

Doubler les bits

Code (4,8) qui duplique tous les bits :

Peut-on corriger 1 erreur ?

00110111

Premier exemple

Doubler les bits

Code (4,8) qui duplique tous les bits :

Peut-on détecter 2 erreurs ?

01110111

Premier exemple

Doubler les bits

Code (4,8) qui duplique tous les bits :

Peut-on détecter 2 erreurs ?

11110011

Doubler les bits

Théorème

Le code (n,2n) qui double les bits :

- *Détecte une erreur, mais pas deux*
- *Ne corrige pas d'erreurs*

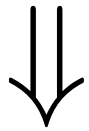
Pas super efficace : on divise le débit par deux, et on n'est même pas capable de détecter plus qu'une erreur.

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

1011



111000111111

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

- Est-ce qu'il détecte une erreur ?
- Est-ce qu'il corrige une erreur ?
- Est-ce qu'il détecte deux erreurs ?
- Est-ce qu'il corrige deux erreurs ?
- Est-ce qu'il détecte trois erreurs ?

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

- Est-ce qu'il détecte une erreur ? Oui

000100111000

- Est-ce qu'il corrige une erreur ?
- Est-ce qu'il détecte deux erreurs ?
- Est-ce qu'il corrige deux erreurs ?
- Est-ce qu'il détecte trois erreurs ?

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

- Est-ce qu'il détecte une erreur ?
- Est-ce qu'il corrige une erreur ? Oui

000100111000

- Est-ce qu'il détecte deux erreurs ?
- Est-ce qu'il corrige deux erreurs ?
- Est-ce qu'il détecte trois erreurs ?

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

- Est-ce qu'il détecte une erreur ?
- Est-ce qu'il corrige une erreur ?
- Est-ce qu'il détecte deux erreurs ? Oui

000100111000

- Est-ce qu'il corrige deux erreurs ?
- Est-ce qu'il détecte trois erreurs ?

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

- Est-ce qu'il détecte une erreur ?
- Est-ce qu'il corrige une erreur ?
- Est-ce qu'il détecte deux erreurs ?
- Est-ce qu'il corrige deux erreurs ? Non

000100111000

000100111000

- Est-ce qu'il détecte trois erreurs ?

Deuxième exemple

Tripler les bits

Code (4,12) qui duplique tous les bits :

- Est-ce qu'il détecte une erreur ?
- Est-ce qu'il corrige une erreur ?
- Est-ce qu'il détecte deux erreurs ?
- Est-ce qu'il corrige deux erreurs ?
- Est-ce qu'il détecte trois erreurs ? Non

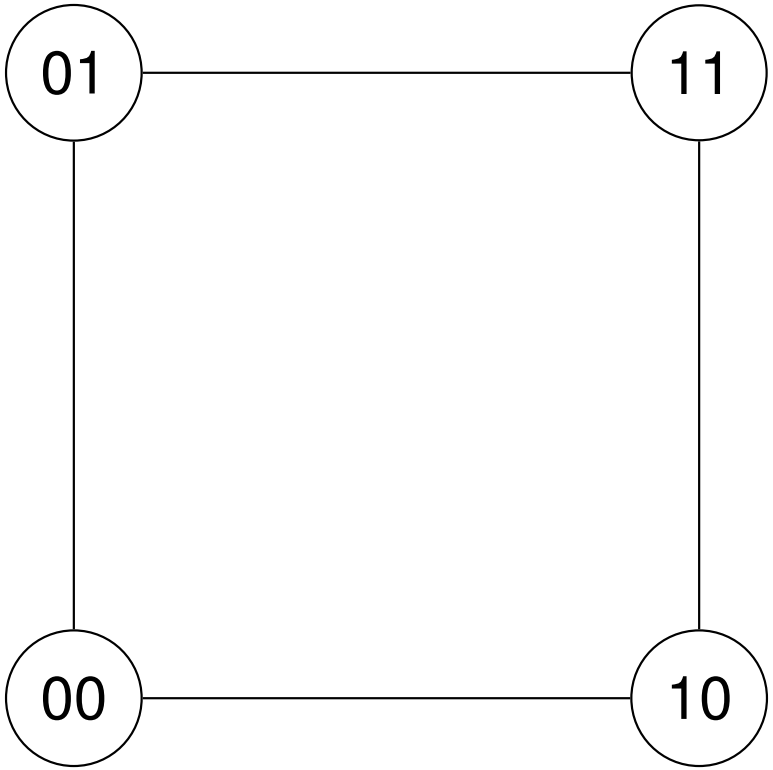
000000111000

Un peu de théorie

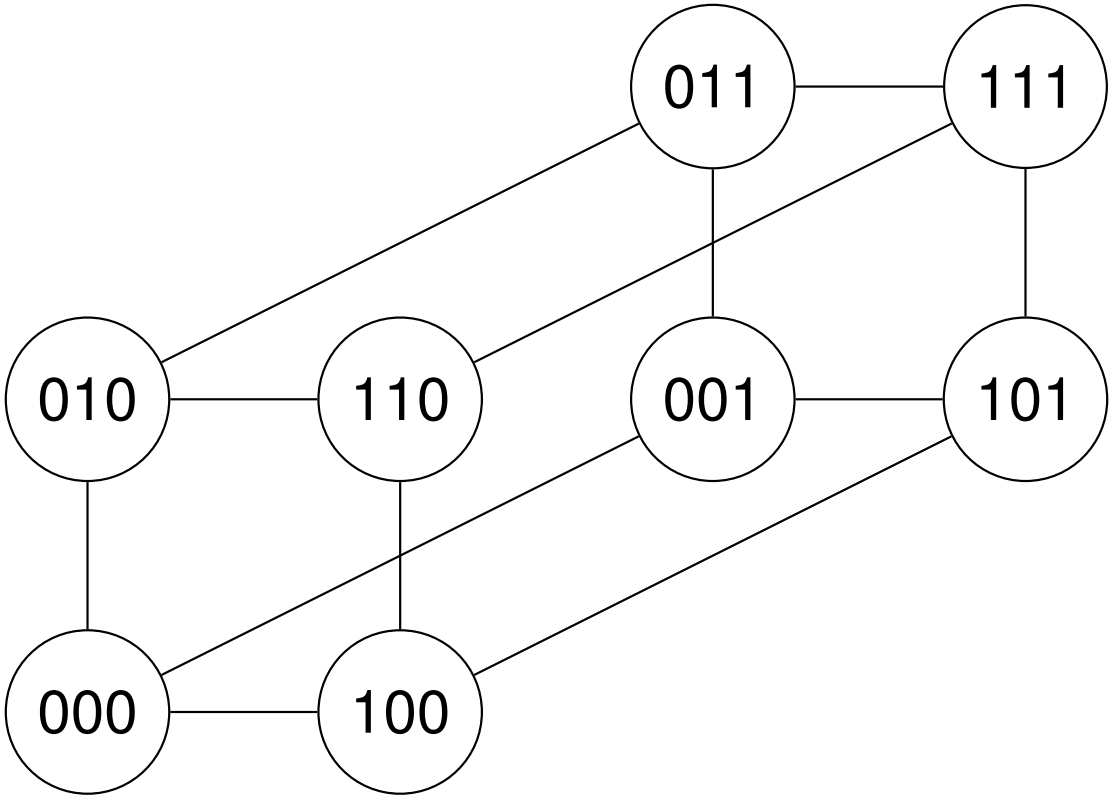
On peut représenter tous les mots sur n bits comme un graphe :

- Sommets : tous les mots possibles
- Arêtes : on relie deux mots s'ils n'ont qu'un bit de différent.

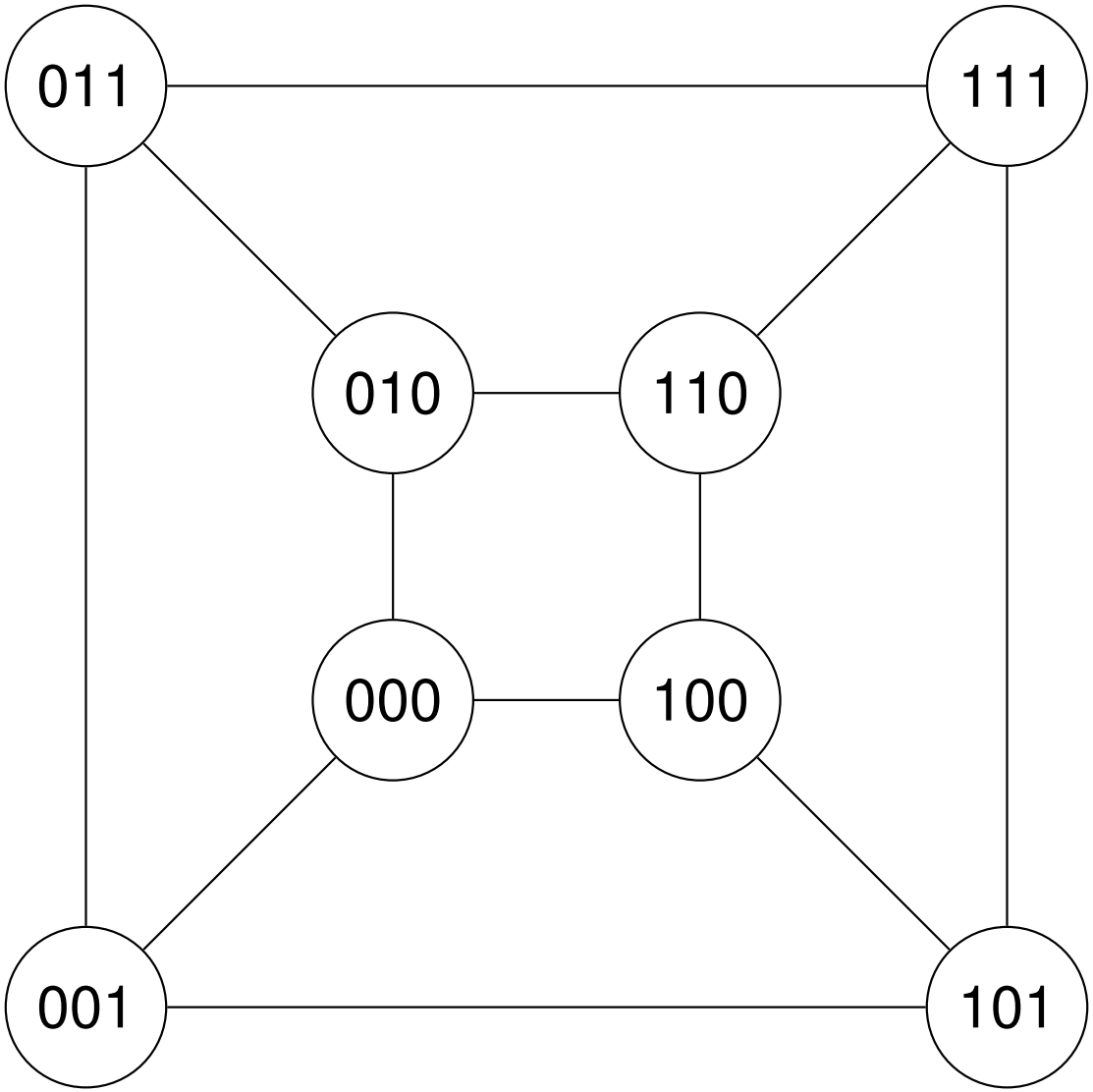
$n = 2$



$n = 3$



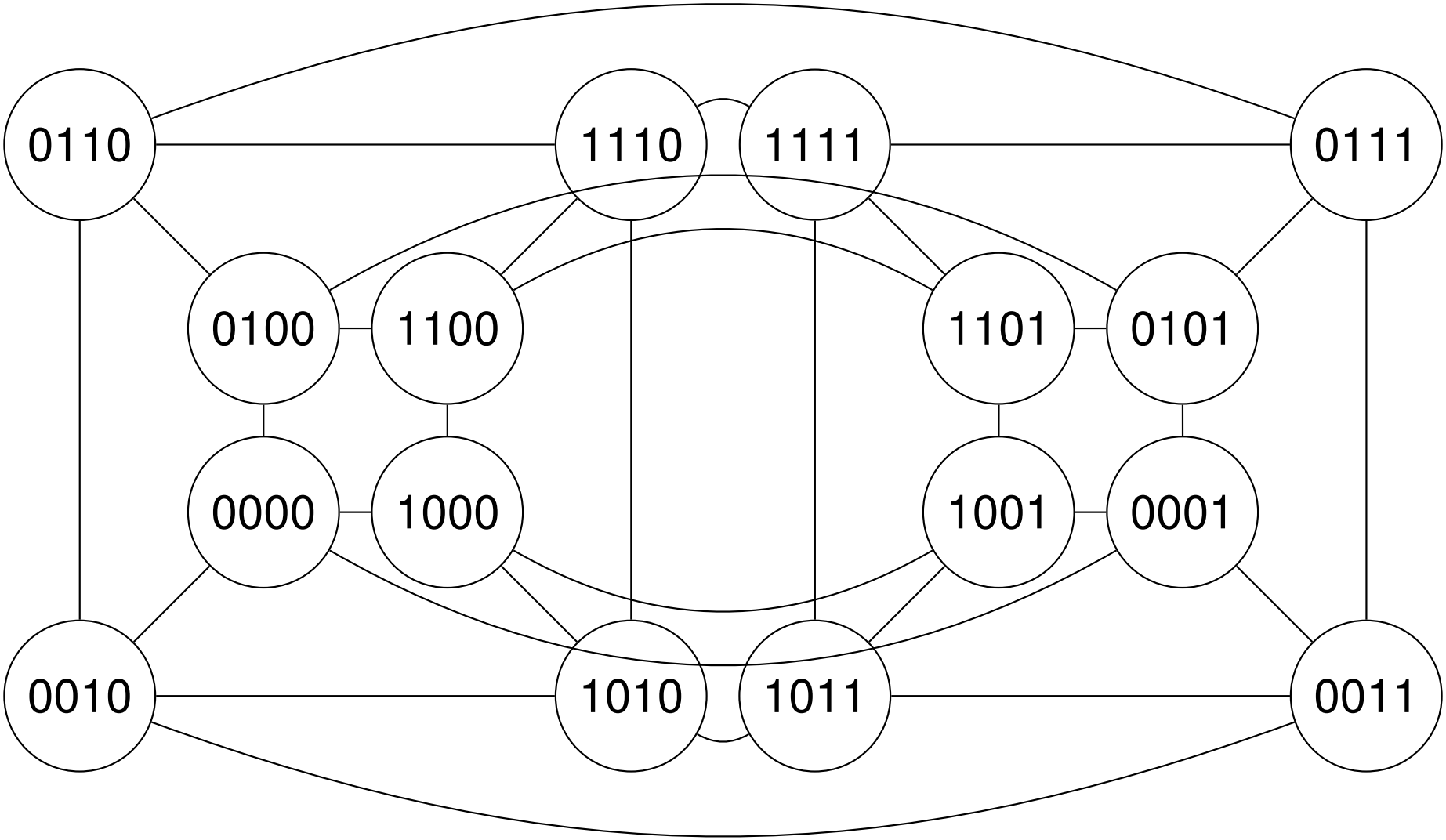
$n = 3$



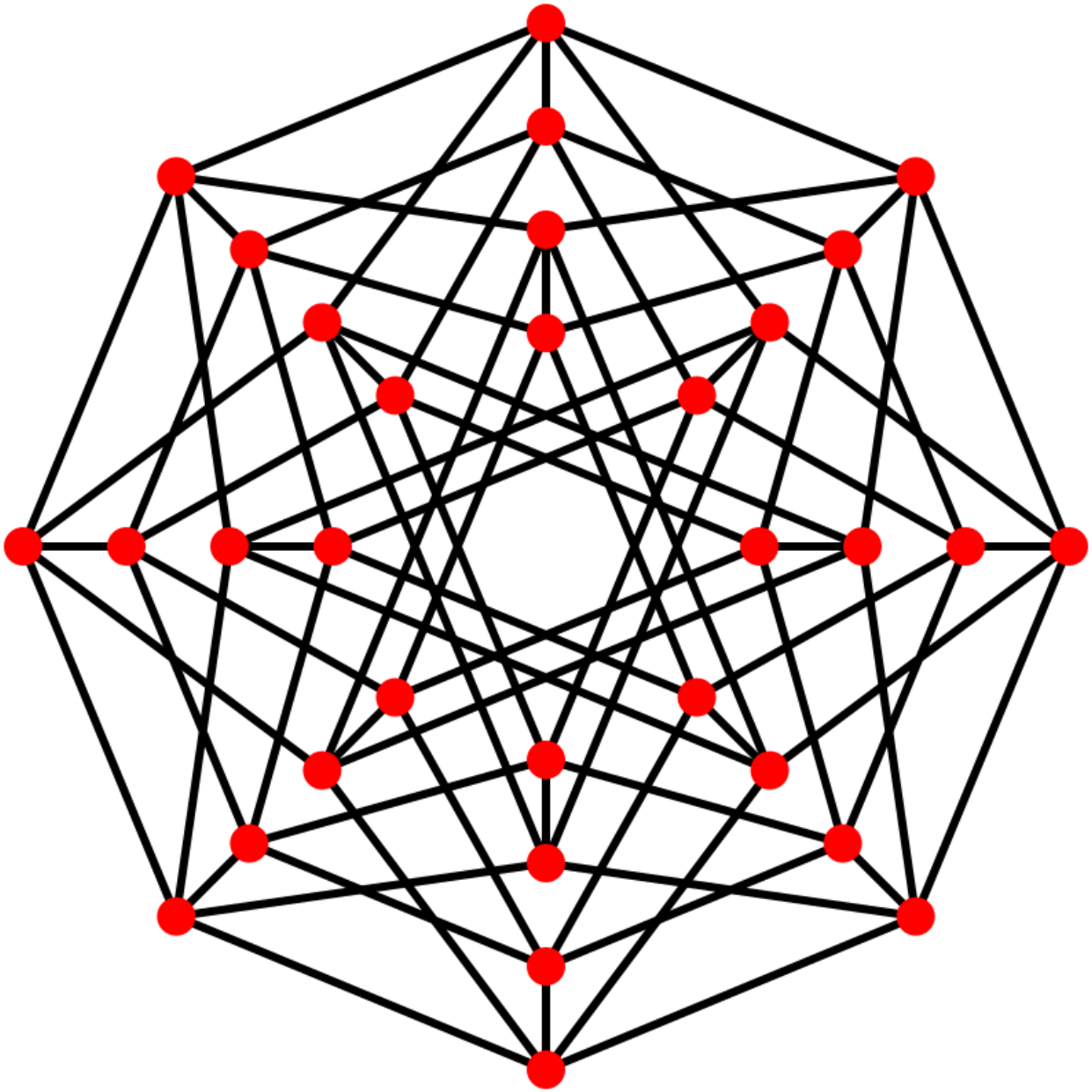
$n = 4$

TODO

$n = 4$


0110
1110
1111
0111
0100
1100
1101
0101
0000
1000
1001
0001
0010
1010
1011
0011

$n = 5$ (from wikipedia)

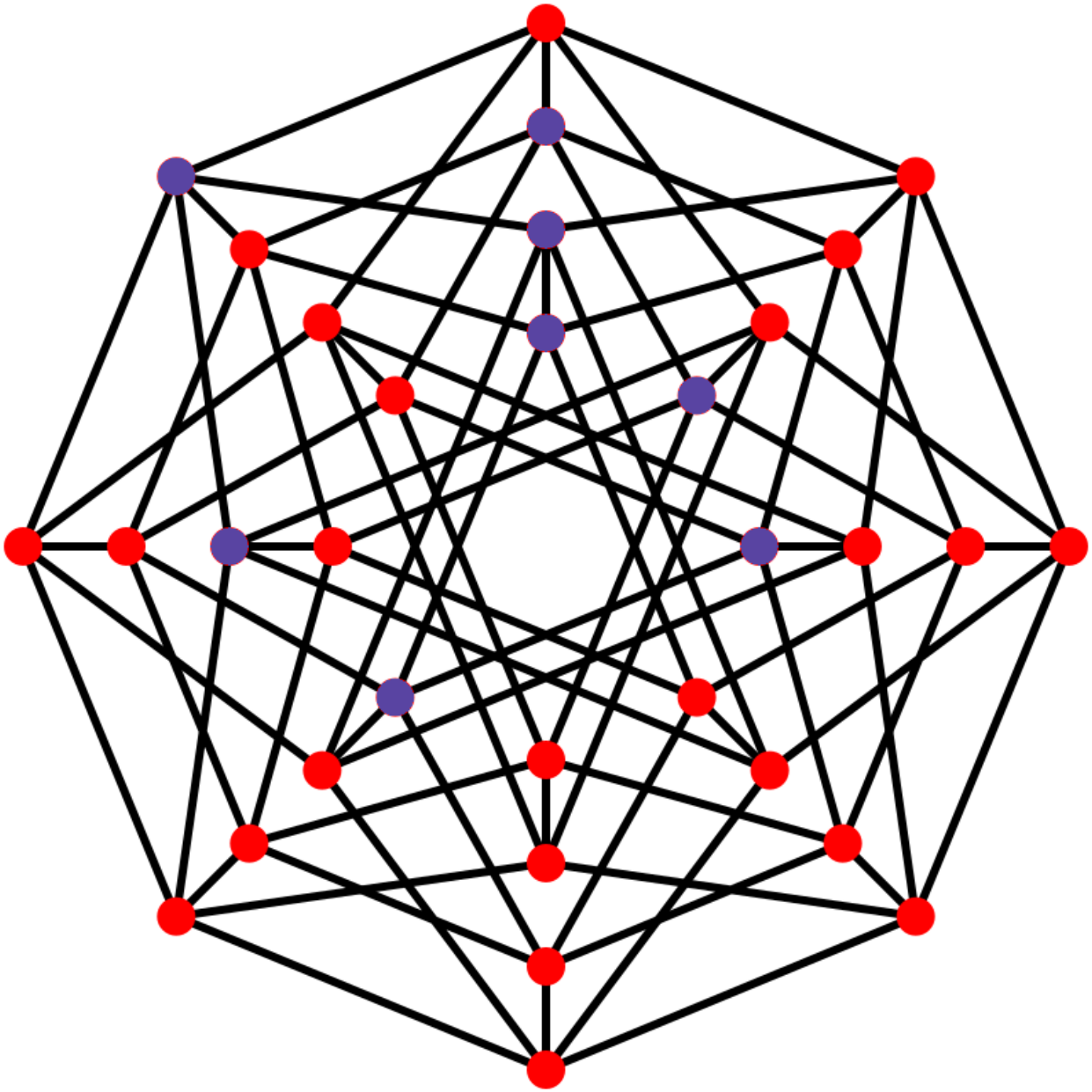


Théorie

Code (3,5) :

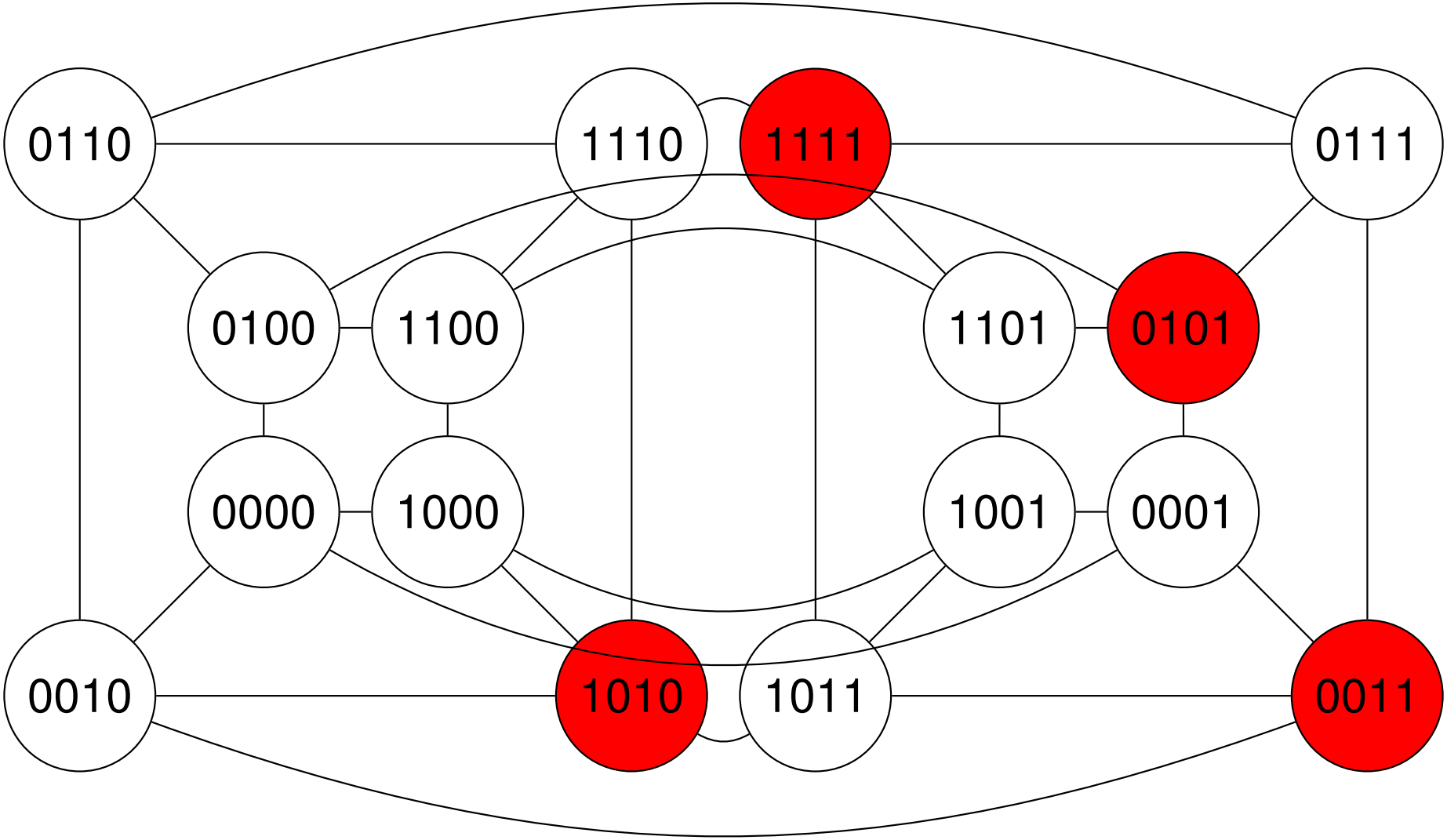
- On code 3 bits en 5 bits
- Cela revient à choisir $8 (= 2^3)$ sommets dans le graphe pour $n = 5$.

Exemple



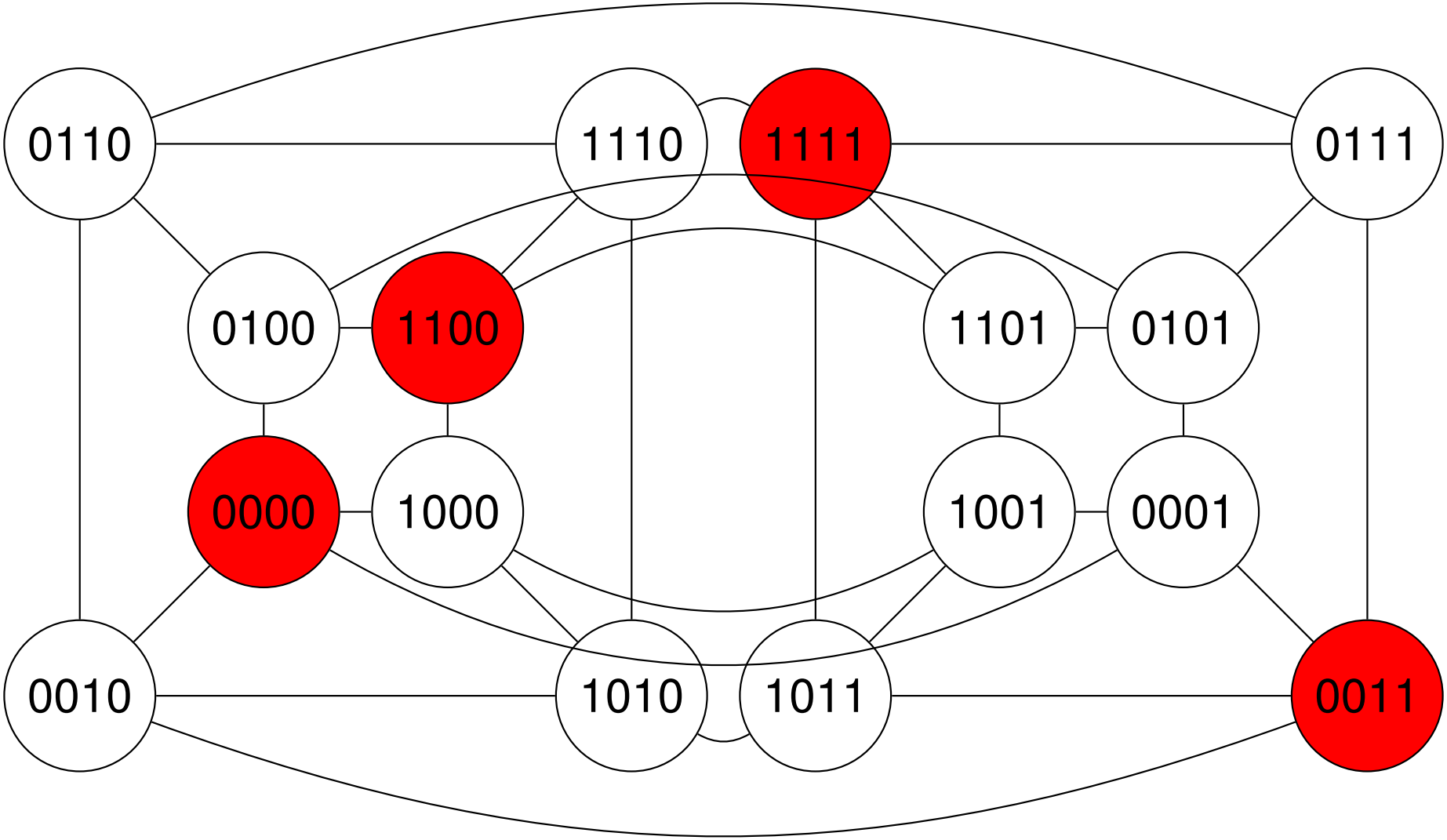
Exemple

Code (2,4) :



Exemple

Code (2,4) :



Théorie

Faire une erreur = se déplacer d'une arête

Théorème

Un code détecte une erreur si, lorsqu'on se place sur un mot de code et qu'on se déplace d'une arête, on ne retombe pas sur un mot de code.

Théorème

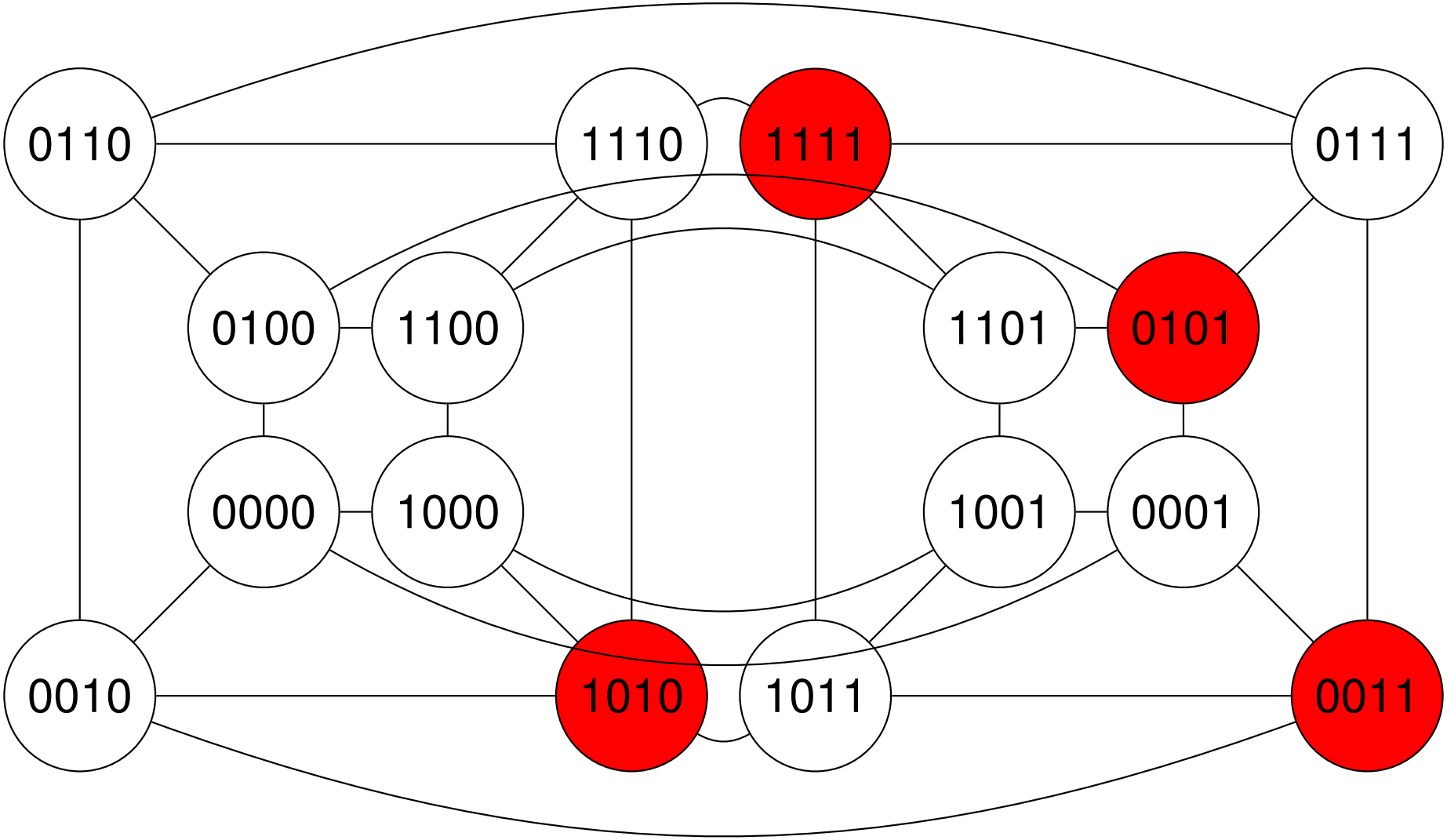
Un code détecte une erreur s'il n'y a pas deux mots de code reliés par une arête.

Théorie

Faire deux erreurs = se déplacer de deux arêtes

Exemple

Ce code (2,4) ne détecte pas deux erreurs.



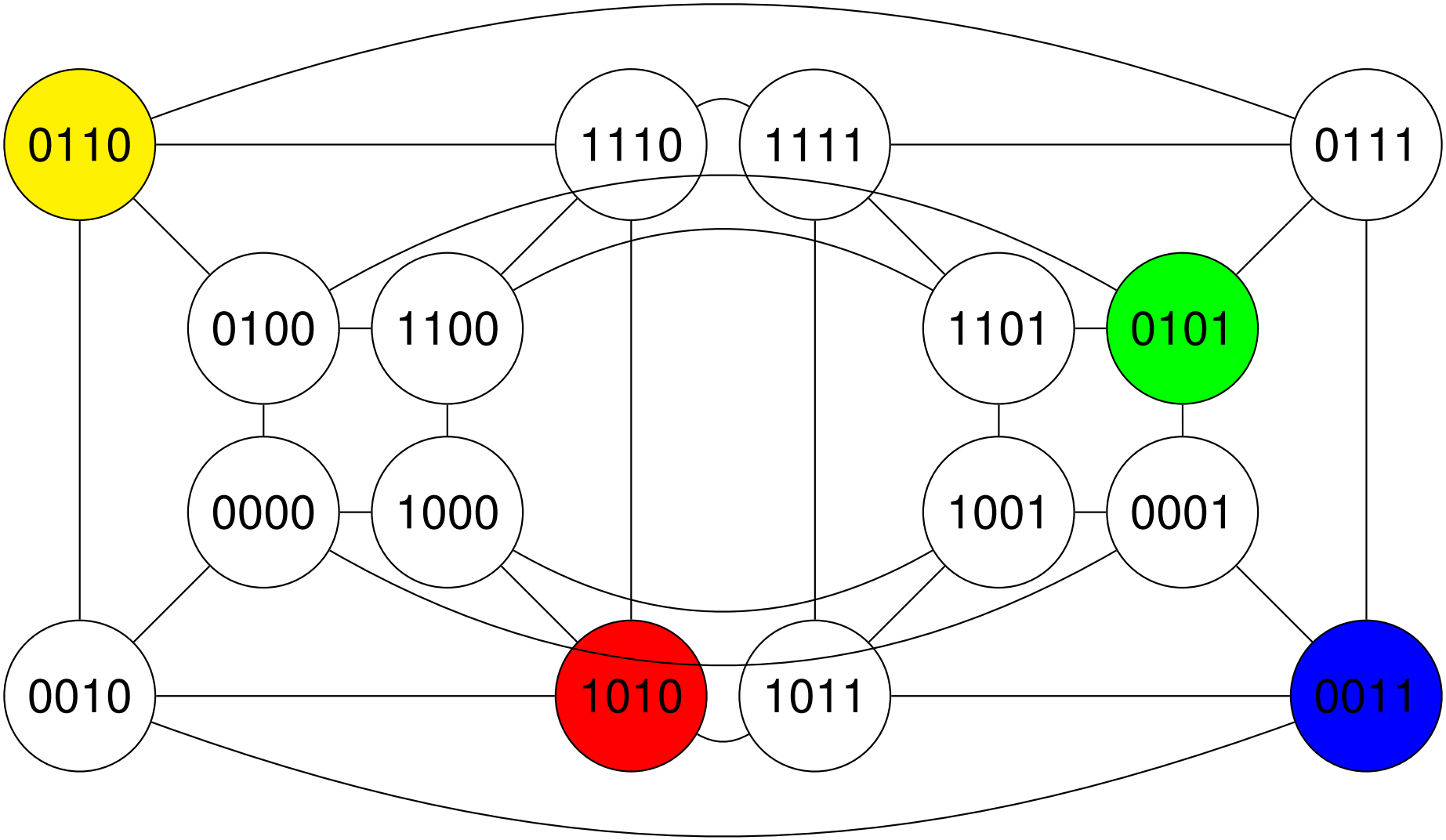
Un théorème

Théorème

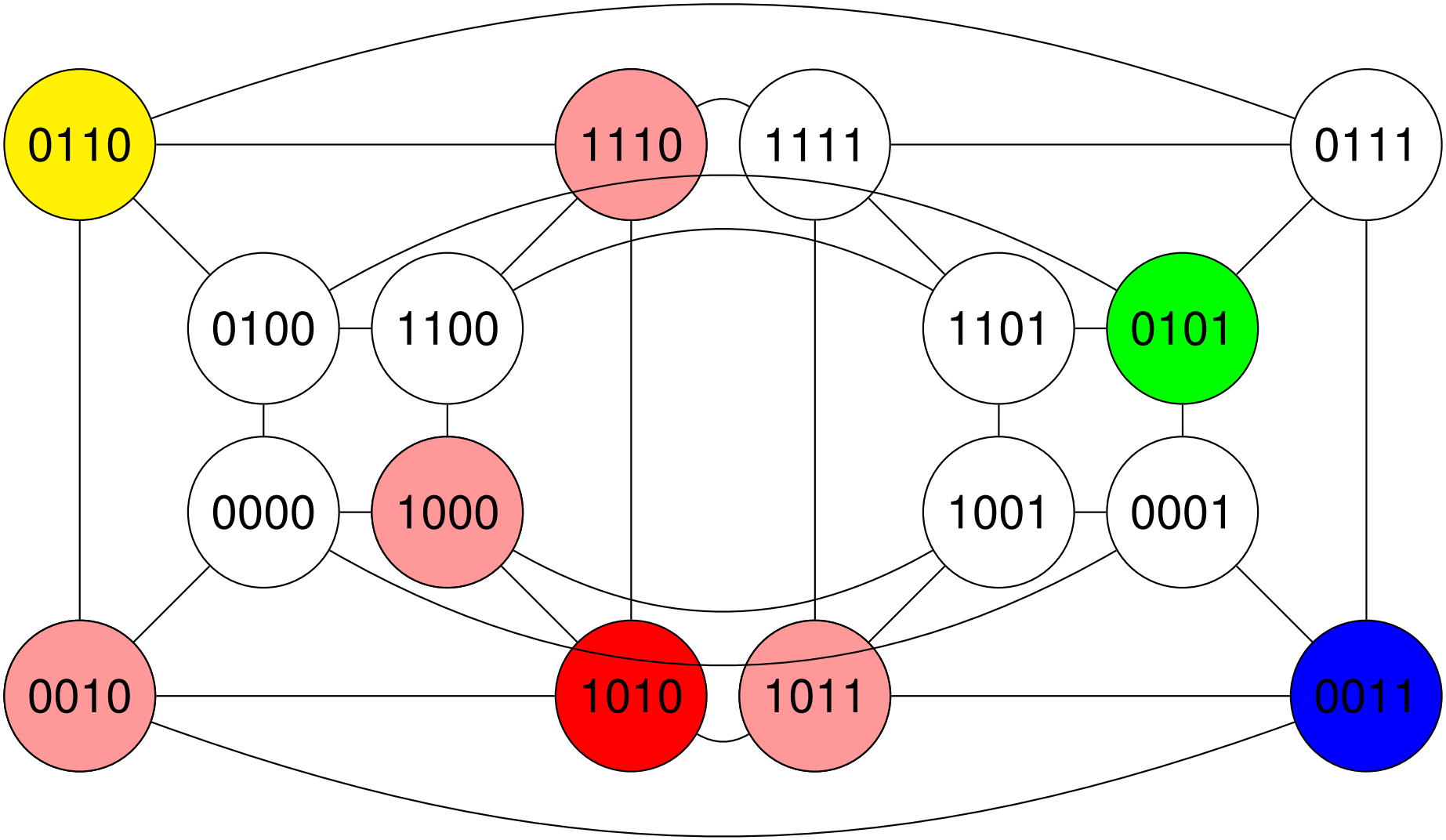
On ne peut pas trouver de code (2,4) qui détecte 2 erreurs.

- Deux sommets choisis ne peuvent pas être voisins, ni avoir de voisins en commun
- A chaque sommet choisi, on peut associer 5 sommets qui lui sont propres : lui-même, et ses 4 voisins

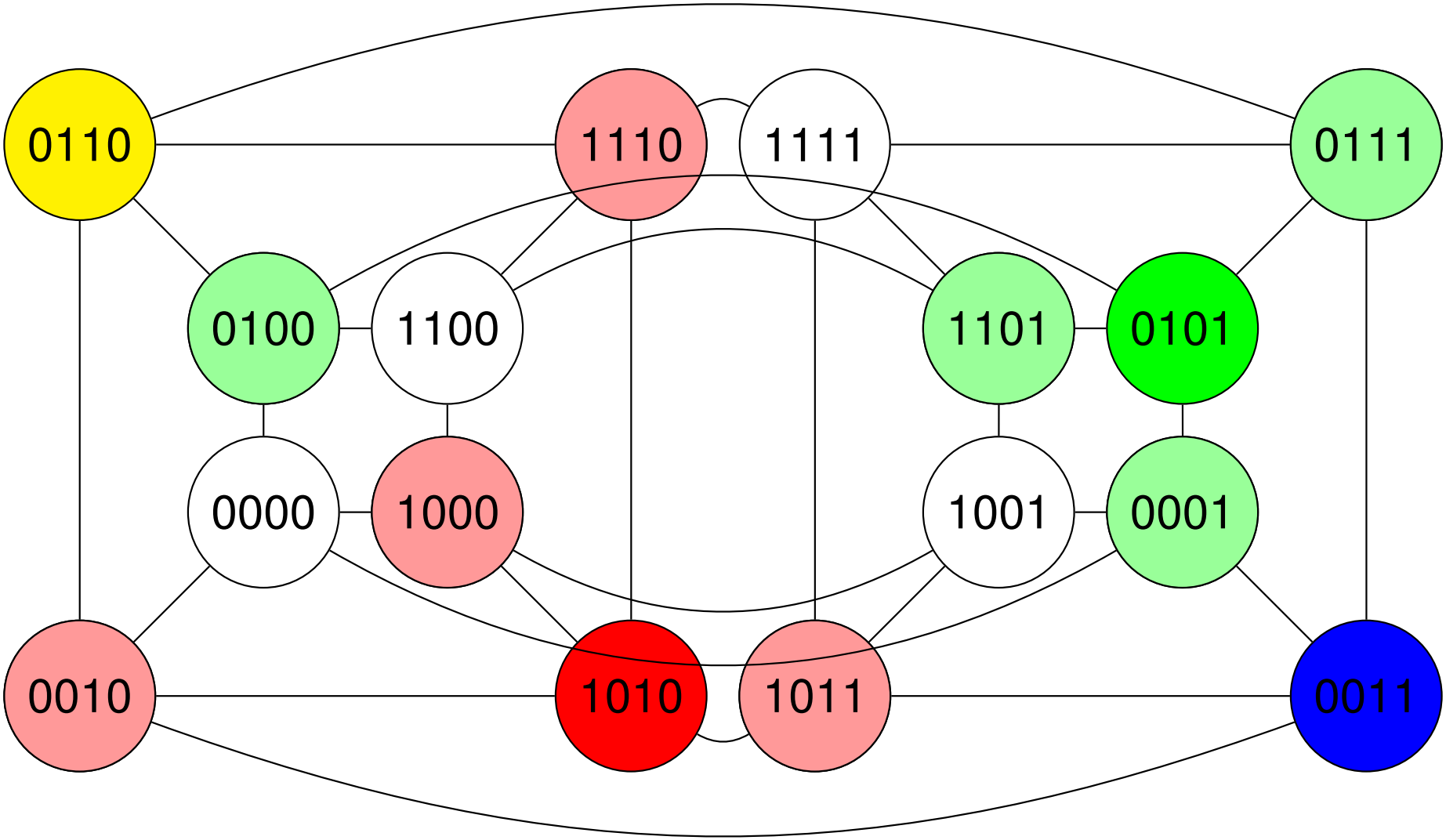
Exemple


0110
1110
1111
0111
0100
1100
1101
0101
0000
1000
1001
0001
0010
1010
1011
0011

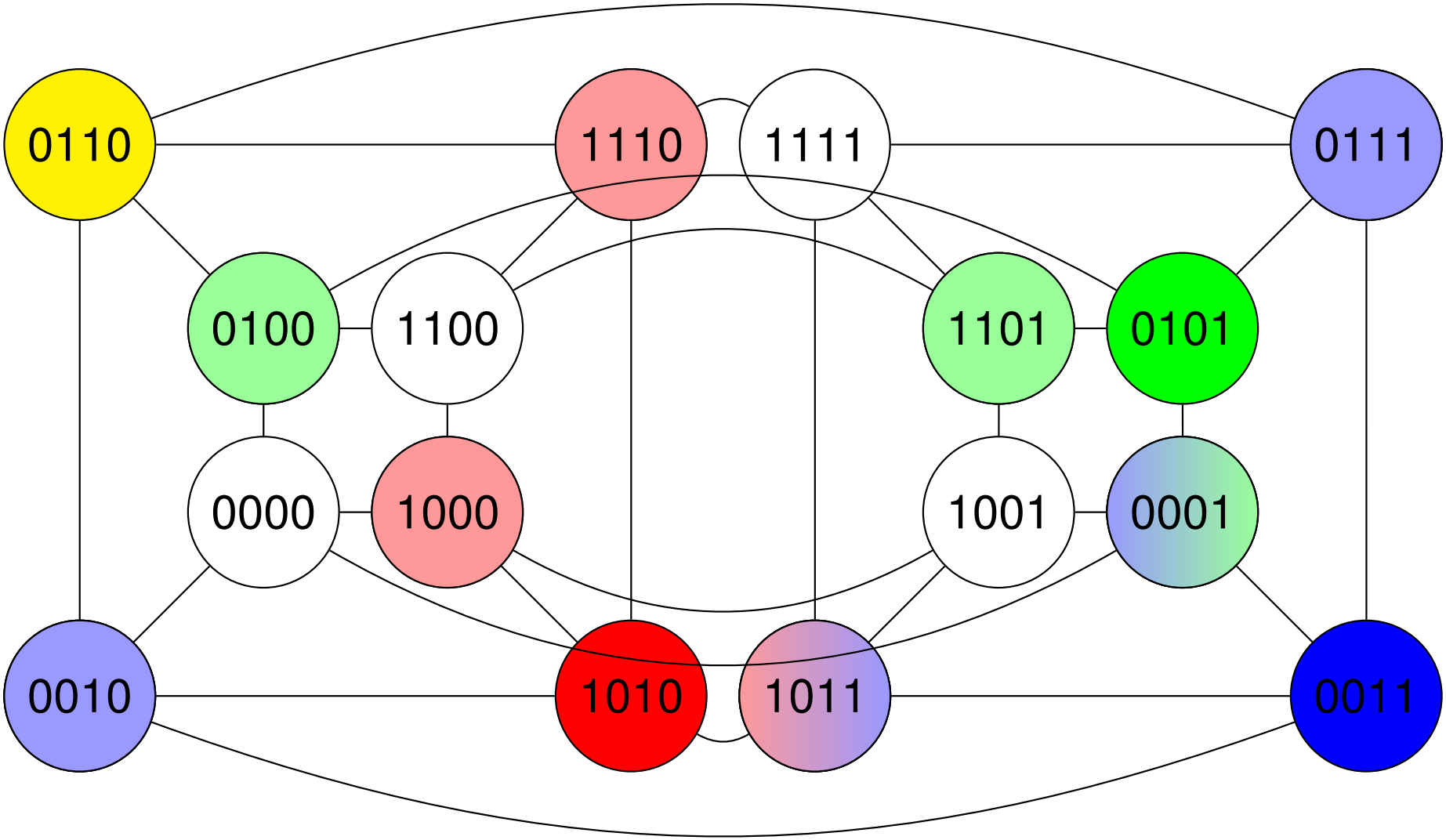
Exemple


0110
1110
1111
0111
0100
1100
1101
0101
0000
1000
1001
0001
0010
1010
1011
0011

Exemple


0110
1110
1111
0111
0100
1100
1101
0101
0000
1000
1001
0001
0010
1010
1011
0011

Exemple


0110
1110
1111
0111
0100
1100
1101
0101
0000
1000
1001
0001
0010
1010
1011
0011

Théorème

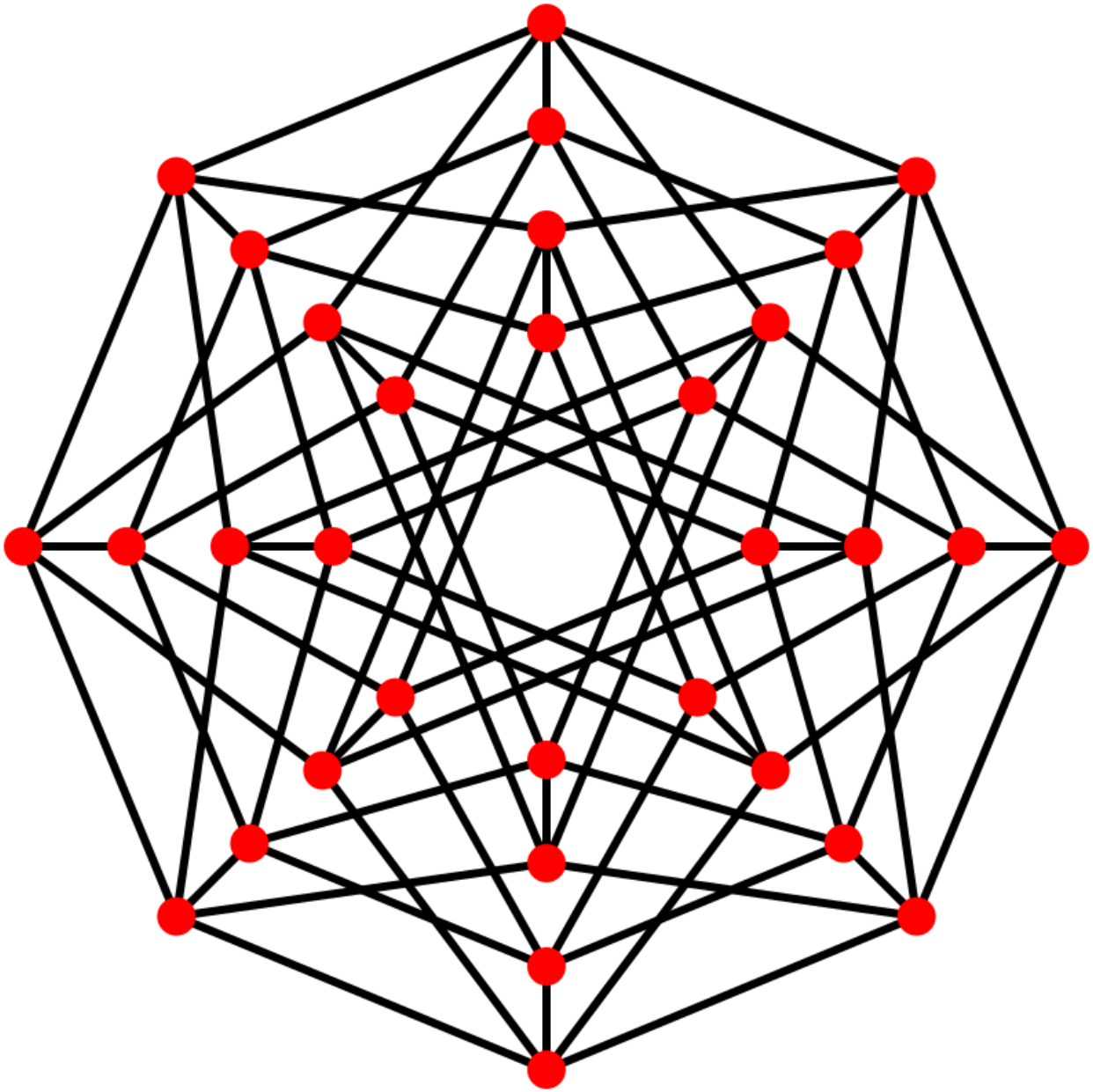
- A chaque sommet choisi, on peut associer 5 sommets qui lui sont propres : lui-même, et ses 4 voisins
- On a choisi 4 sommets, ca fait donc $4 \times 5 = 20$ sommets en tout. Ce n'est pas possible, on n'a que 16 sommets.

Code (2,5)

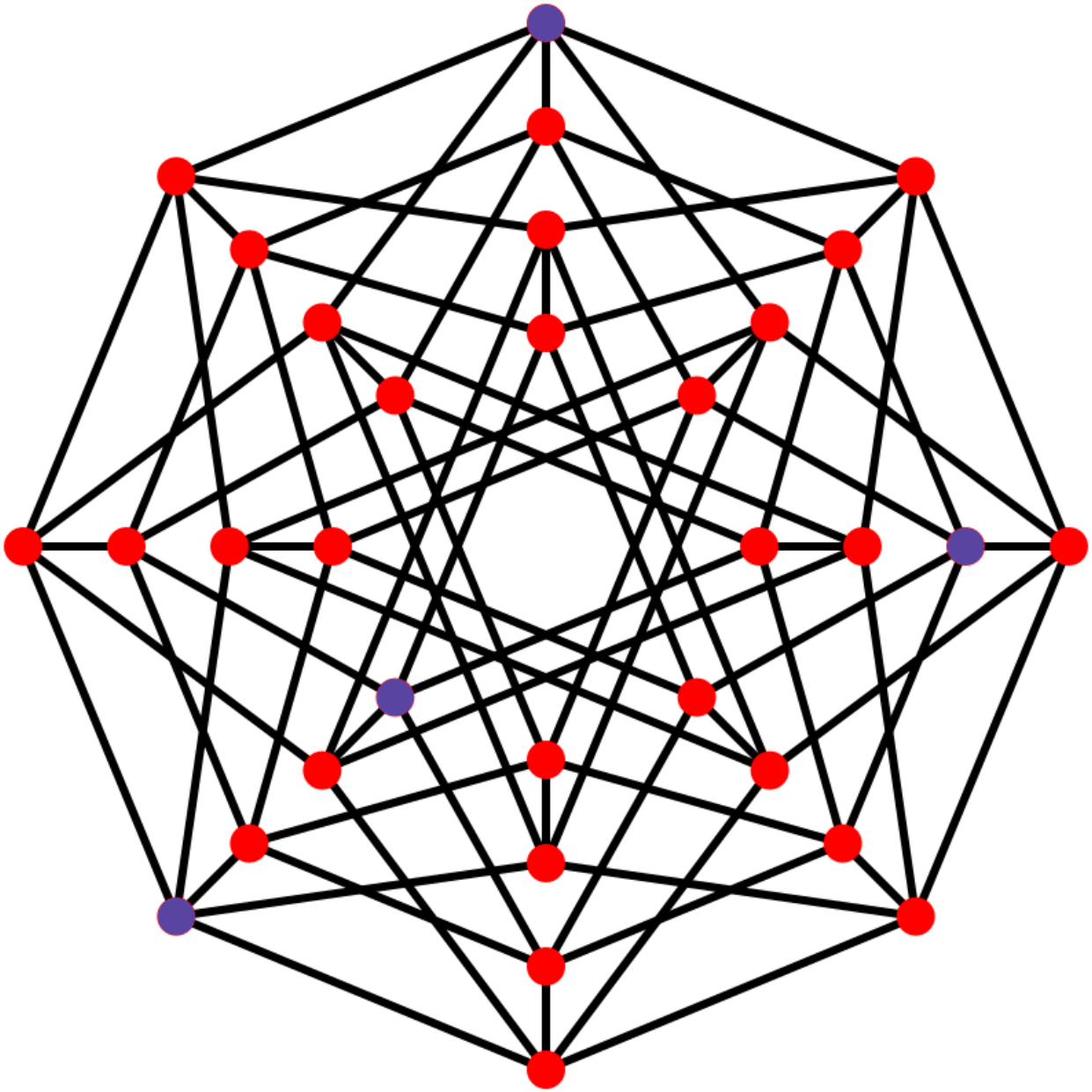
On peut trouver un code (2,5) qui détecte deux erreurs.

Cela revient à trouver 4 sommets dans le graphe pour $n = 5$ qui sont tous à distance 2 ou plus les uns des autres.

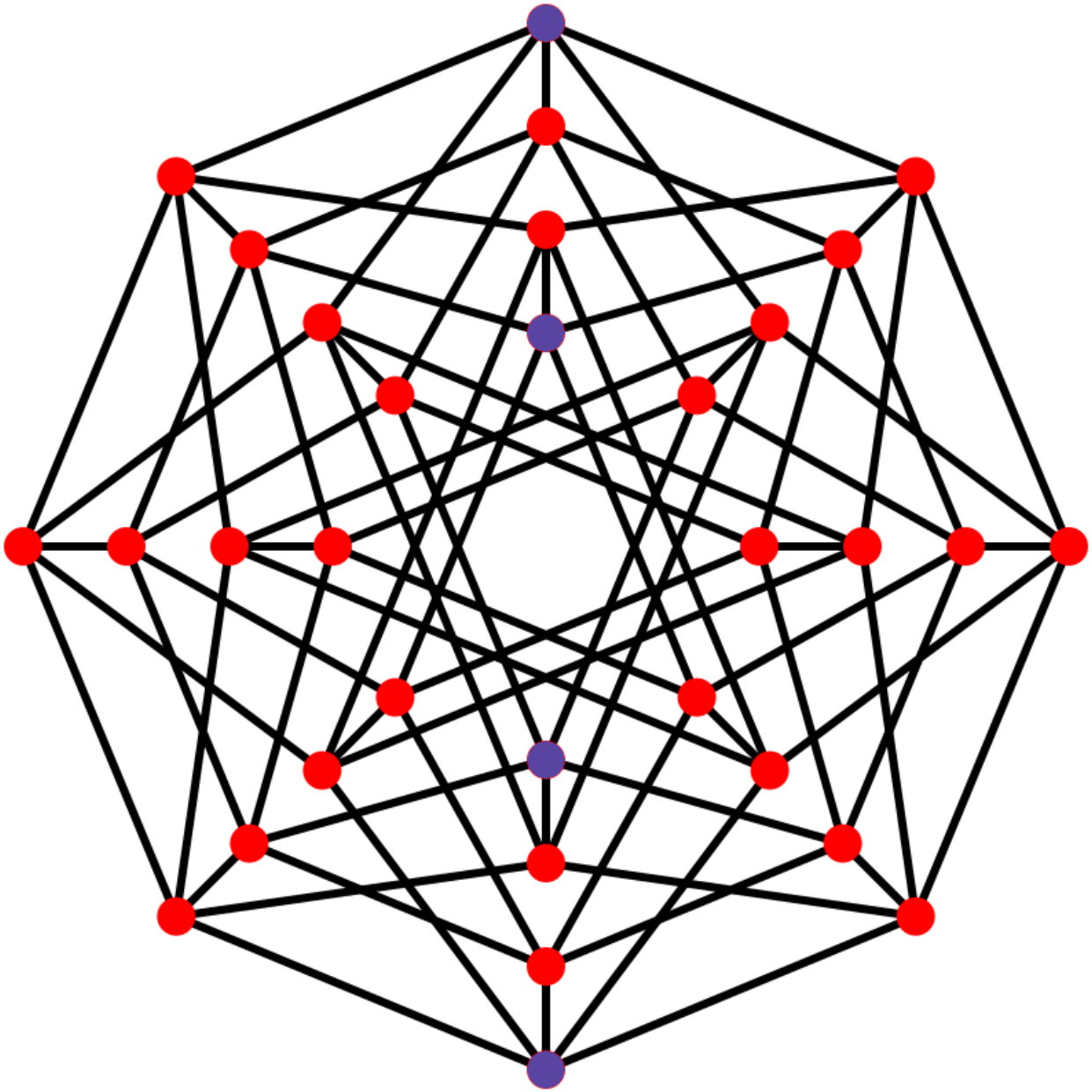
A vous !



A vous !



A vous !



Théorie (suite)

- Détecter k erreurs : les sommets sont à distance $\geq k+1$ les uns des autres.
- On appelle *distance de Hamming* la distance entre deux mots (entre deux sommets).

Correction

Distance 1 : on ne peut pas détecter d'erreurs

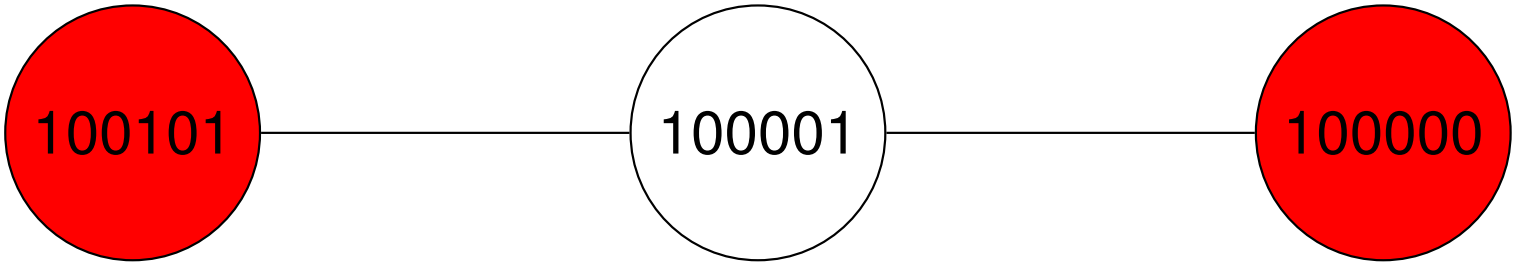
Correction

Distance 1 : on ne peut pas détecter d'erreurs

On envoie : 100101.
On reçoit : 100001.
On ne voit pas le problème

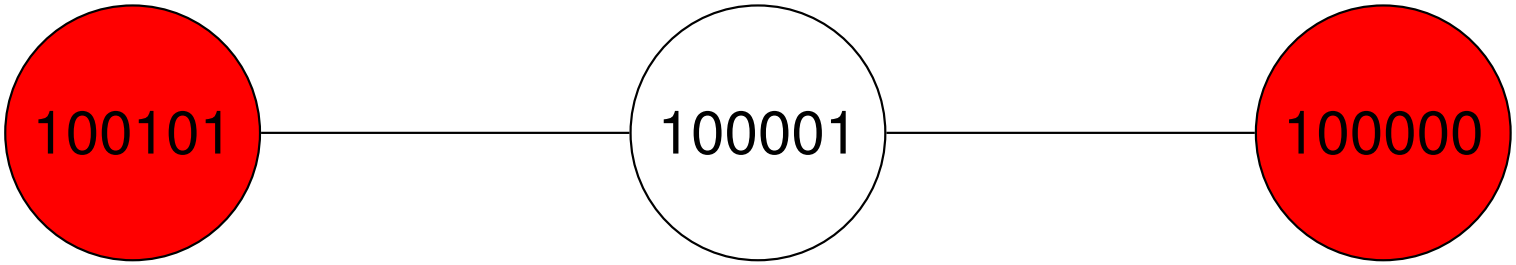
Correction

Distance 2 : on peut détecter une erreur



Correction

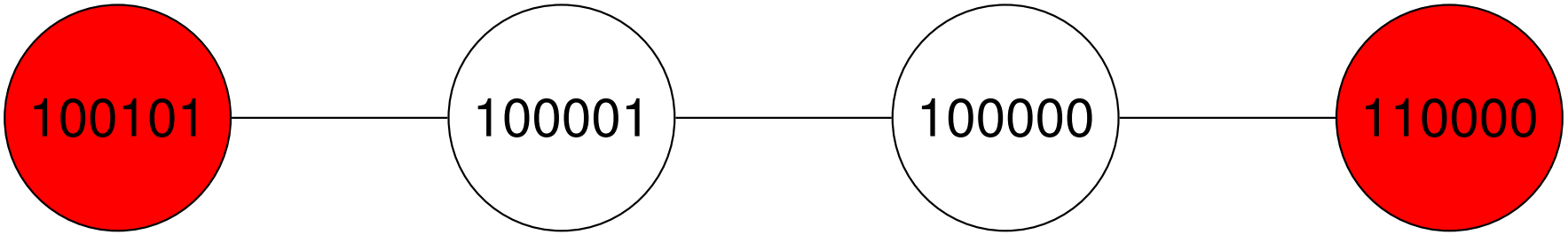
Distance 2 : on peut détecter une erreur



On envoie : 100101.
On reçoit : 100001.
On voit le problème, mais on ne sait pas le corriger

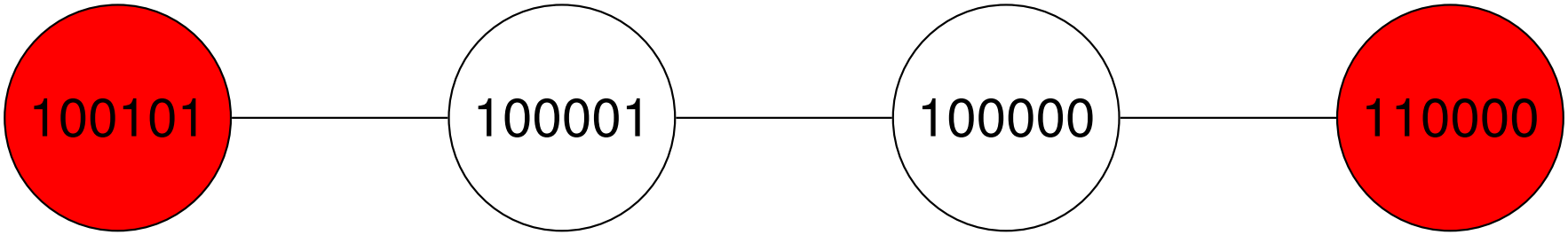
Correction

Distance 3 : on peut corriger une erreur



Correction

Distance 3 : on peut corriger une erreur



On envoie : 100101.
On reçoit : 100001.
On voit le problème, et on sait le corriger

Théorie (suite et fin)

Théorème

Détecter k erreurs = les sommets sont à distance $\geq k+1$ les uns des autres.

Théorème

Corriger k erreurs = les sommets sont à distance $\geq 2k+1$ les uns des autres.

Donc détecter 2 erreurs = corriger une erreur.

Contenu

Bit de parité

Code $(n, n+1)$ (on ajoute un seul bit).

- Le bit vaut 0 s'il y a un nombre pair de 1 dans le message, et 1 sinon.

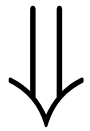
$$10110$$

$$\Downarrow$$

$$101101$$

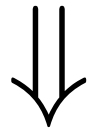
Bit de parité

00110



??????

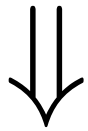
11010



??????

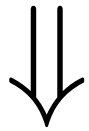
Bit de parité

00110



001100

11010



110101

Erreurs

Est-ce qu'il y a une erreur ?

- 001101 ?
- 100100 ?
- 100101 ?

Erreurs

Est-ce qu'il y a une erreur ?

- 001101 ? Oui
- 100100 ? Non
- 100101 ? Oui

Parité

Théorème

Le code de parité $(n, n+1)$ *détecte une erreur, mais pas deux (et donc ne corrige pas l'erreur)*

- Peu coûteux (un bit en plus)
- Utilisé un peu partout (ex : RS-232)

Codes en rectangle

On part de $n \times m$ bits, organisés en rectangle ($n = m = 3$ dans l'exemple)

$$\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{array}$$

Pour obtenir le code, on calcule la parité de chaque ligne et de chaque colonne

$$\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & \end{array}$$

Codes en rectangle

Est-ce qu'il y a une erreur ? Peut-on la corriger ?

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

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

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

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

Codes en rectangle

Est-ce qu'il y a une erreur ? Peut-on la corriger ?

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

Pas d'erreur

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

Une erreur

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

Une erreur

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

Une erreur

Cas particulier

Dans le cas $n = 1, m = 2$ on obtient :

x	x
y	y
$x + y$	

C'est à dire les 4 mots : 00000, 00111, 11001, 11110.
C'est le code (2,5) vu plus tôt !

CRC

Idée des CRC : Numéro de sécu

- On part d'un code à 13 chiffres :

  2 80 01 54 166 642

  (les espaces n'ont aucune importance)
- On rajoute deux chiffres de telle sorte que le résultat (le nombre à 15 chiffres) soit divisible par 97

  2 80 01 54 166 642 73

Numéro de Secu

Si on se trompe sur un chiffre, clairement le résultat ne sera plus divisible par 97. Pourquoi ?

- Si on se trompe sur le dernier chiffre (p.e. 4 à la place de 3), on a modifié le résultat de 1, qui n'est pas multiple de 97
- Si on se trompe sur l'avant dernier chiffre (p.e. 3 à la place de 7), on a modifié le résultat de 40, qui n'est pas multiple de 97
- Si on se trompe sur l'avant-avant-dernier chiffre (p.e. 8 à la place de 2), on a modifié le résultat de 600, qui n'est pas multiple de 97
- etc

Numéro de Secu

Comment trouver les 2 derniers chiffres ?

- On divise 2 80 01 54 166 642 00 par 97 et on regarde le reste
- On trouve 24
- $97 - 24 = 73$, donc si on remplace le 00 par 73, on a augmenté le reste de 73 et il vaut maintenant 97, donc 0.

Méthode :

- Ajouter deux 0 au nombre
- Calculer le reste de la division par 97, on l'appelle x
- Les deux chiffres à ajouter (à la place des 0) sont $97 - x$.

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

CRC = même principe sauf que :

- On ne joue pas avec des nombres mais avec des polynômes
- On raisonne modulo 2 : $(1 + 1 = 0)$.

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

Exemple avec le CRC : $G(x) = x^5 + x^2 + 1$ (CRC-5-USB).

- Ce polynôme $G(x)$ joue (quasi) le rôle du nombre 97.

Calculons le CRC pour la suite de bits 1000110

- On ajoute 5 zéros à la fin de la suite de bits (pourquoi 5 ? Car le polynôme $G(x)$ est de degré 5)

$$100011000000$$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

Exemple avec le CRC : $G(x) = x^5 + x^2 + 1$ (CRC-5-USB).

- Ce polynôme $G(x)$ joue (quasi) le rôle du nombre 97.

Calculons le CRC pour la suite de bits 1000110

- On ajoute 5 zéros à la fin de la suite de bits (pourquoi 5 ? Car le polynôme $G(x)$ est de degré 5)

$$100011000000$$

- On voit la suite de bits comme un polynôme :

$$x^{11} + x^7 + x^6$$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

Exemple avec le CRC : $G(x) = x^5 + x^2 + 1$ (CRC-5-USB).

- Ce polynôme $G(x)$ joue (quasi) le rôle du nombre 97.

Calculons le CRC pour la suite de bits 1000110

- On divise $x^{11} + x^7 + x^6$ par $G(x)$ et on prend le reste, qui vaut $x^4 + x^3 + 1$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

Exemple avec le CRC : $G(x) = x^5 + x^2 + 1$ (CRC-5-USB).

- Ce polynôme $G(x)$ joue (quasi) le rôle du nombre 97.

Calculons le CRC pour la suite de bits 1000110

- On reconvertit le reste $x^4 + x^3 + 1$ en 0 et 1, ce qui fait 11001 et on le met à la place des 5 zéros : 1000111011001

Comment faire la division ?

$$x^{11} \qquad +x^{7} \quad +x^{6} \qquad \Big| \; x^5 + x^2 + 1$$

Comment faire la division ?

$$x^{11} \qquad +x^7 \quad +x^6 \qquad \Big| \; x^5 + x^2 + 1$$
$$\overline{\qquad\qquad\qquad\qquad\quad}$$
$$\Big| \; x^6$$

Comment faire la division ?

$$
\begin{array}{l|l}
x^{11} + x^7 + x^6 & x^5 + x^2 + 1 \\
\cline{2-2}
x^{11} \quad x^8 \quad\quad x^6 & x^6
\end{array}
$$

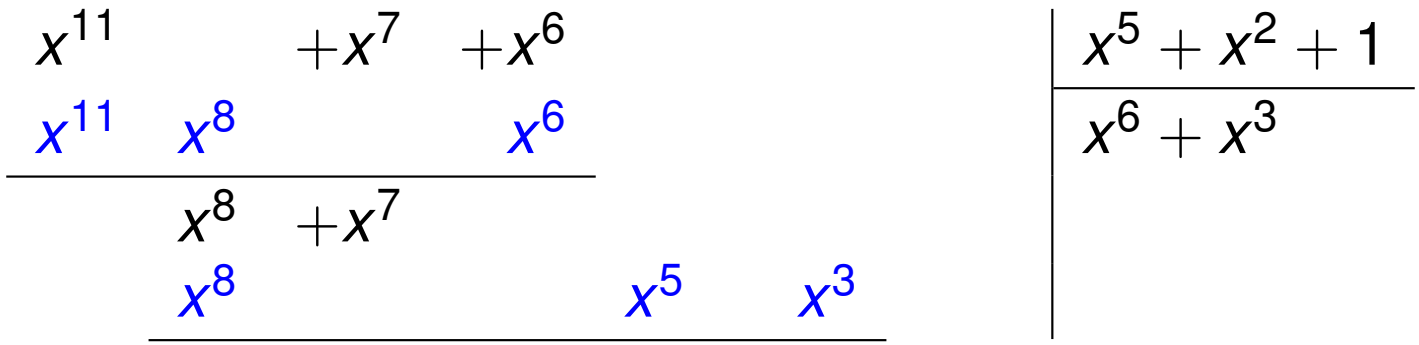
Comment faire la division ?

$$
\begin{array}{lll|l}
x^{11} & & +x^7 \quad +x^6 & x^5+x^2+1 \\
\cline{4-4}
x^{11} \quad x^8 & & \quad\quad\; x^6 & x^6 \\
\cline{1-3}
\quad\quad x^8 & +x^7 & &
\end{array}
$$

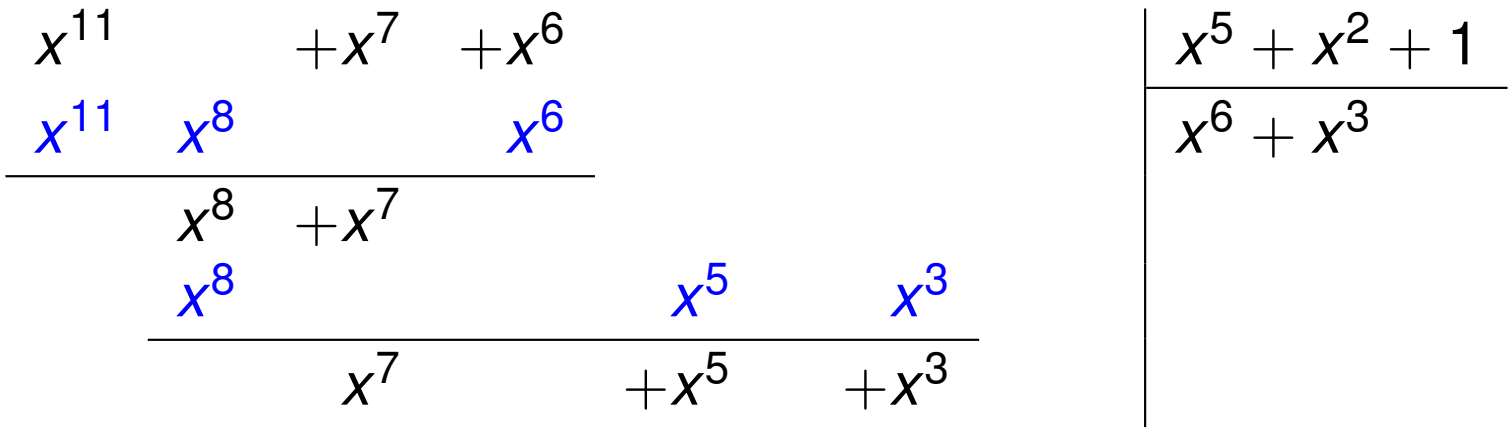
Comment faire la division ?

$$
\begin{array}{llll|l}
x^{11} & & +x^7 & +x^6 & x^5+x^2+1 \\
\cline{5-5}
x^{11} & x^8 & & x^6 & x^6+x^3 \\
\cline{1-4}
 & x^8 & +x^7 & & \\
\end{array}
$$

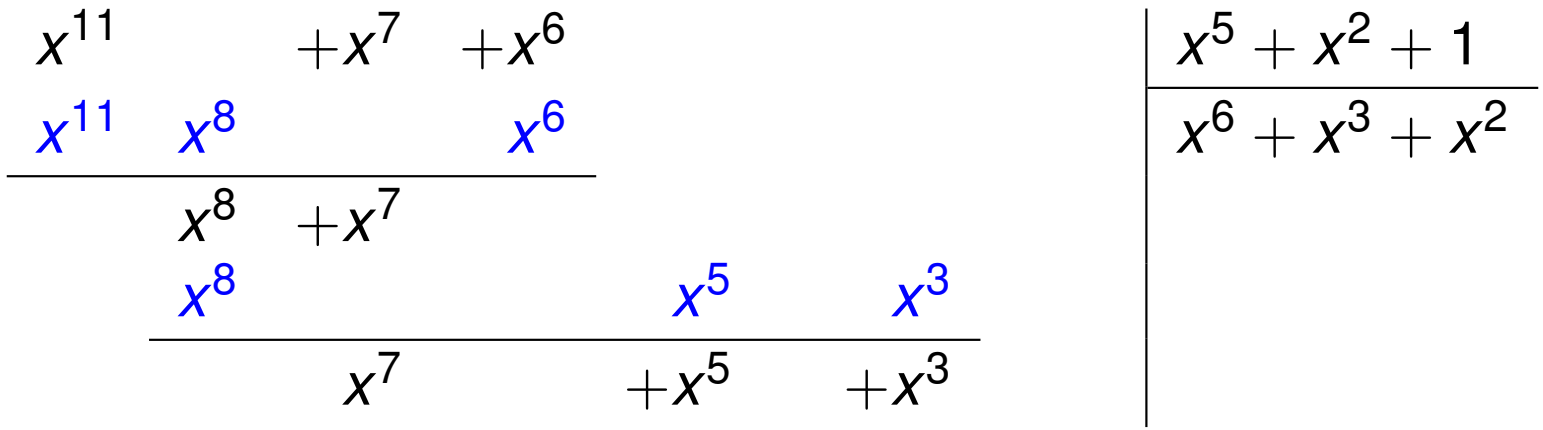
Comment faire la division ?



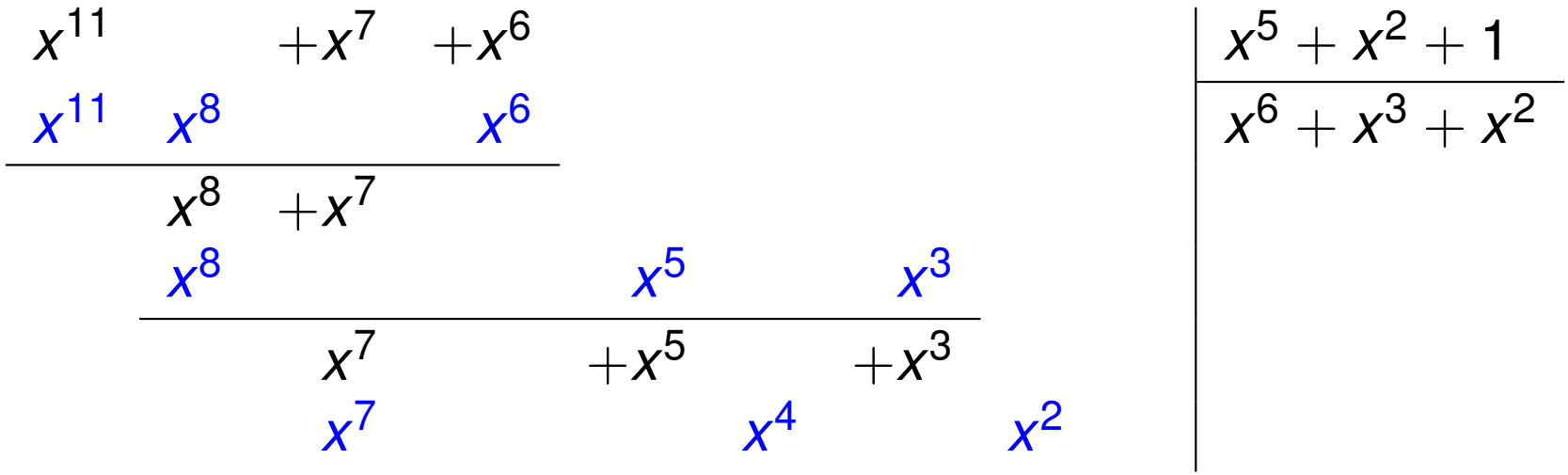
Comment faire la division ?



Comment faire la division ?



Comment faire la division ?



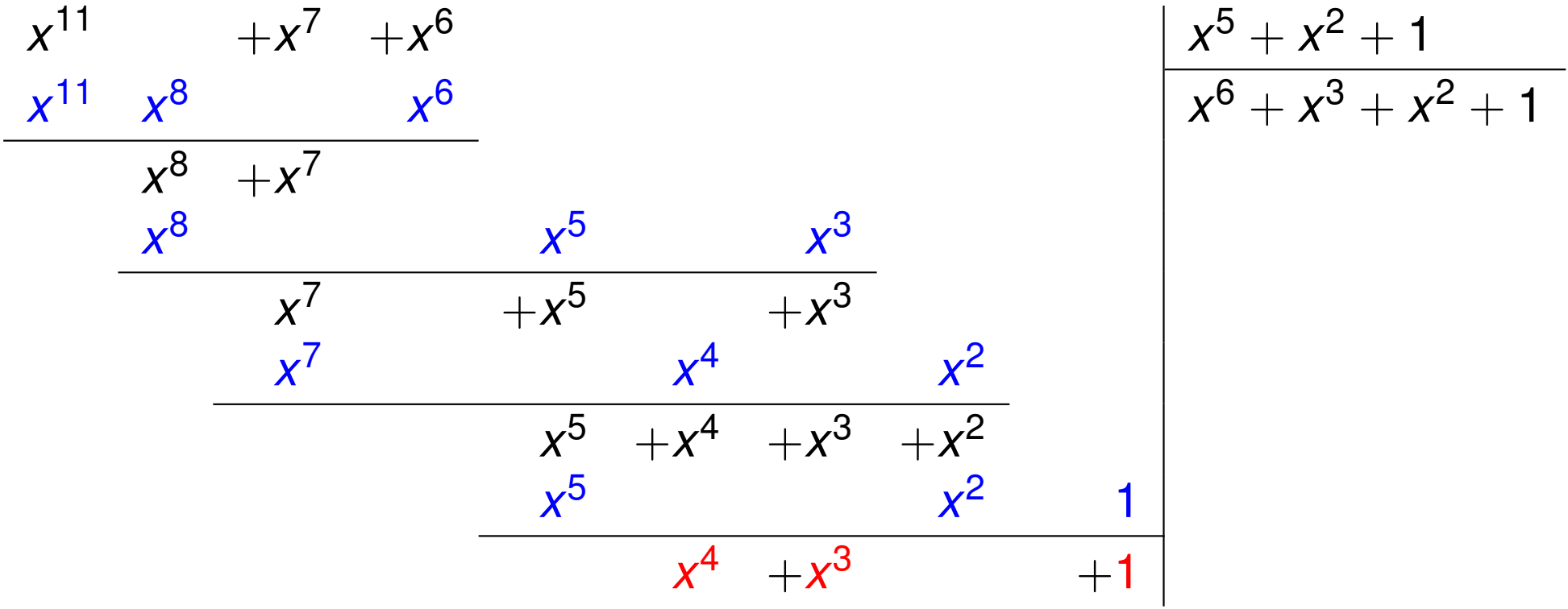
Comment faire la division ?

$$
\begin{array}{llllllll|l}
x^{11} & & & +x^7 & +x^6 & & & & x^5+x^2+1 \\
\cline{9-9}
x^{11} & x^8 & & & x^6 & & & & x^6+x^3+x^2 \\
\cline{1-5}
 & x^8 & +x^7 & & & & & & \\
 & x^8 & & & x^5 & & x^3 & & \\
\cline{2-7}
 & & x^7 & & +x^5 & & +x^3 & & \\
 & & x^7 & & & x^4 & & x^2 & \\
\cline{3-8}
 & & & & x^5 & +x^4 & +x^3 & +x^2 &
\end{array}
$$

Comment faire la division ?

$$
\begin{array}{llllll|l}
x^{11} & & +x^7 & +x^6 & & & x^5+x^2+1 \\
\cline{7-7}
x^{11} & x^8 & & x^6 & & & x^6+x^3+x^2+1 \\
\cline{1-4}
 & x^8 & +x^7 & & & & \\
 & x^8 & & x^5 & & x^3 & \\
\cline{2-5}
 & & x^7 & +x^5 & & +x^3 & \\
 & & x^7 & & x^4 & & x^2 \\
\cline{3-6}
 & & & x^5 & +x^4 & +x^3 & +x^2 \\
 & & & x^5 & & & x^2 \quad 1
\end{array}
$$

Comment faire la division ?



Un autre exemple

Un autre exemple : On reste avec le CRC : $G(x) = x^5 + x^2 + 1$ (CRC-5-USB).

...

Variante

Autre façon de faire, équivalent mais peut-être plus simple

- On voit $G(x)$ comme une suite de 0 et de 1. Pour $G(x) = x^5 + x^2 + 1$, on obtient donc 100101
- Pour calculer le CRC pour la suite de bits 1000110 :
  - On ajoute 5 zéros : 100011000000
  - On aligne 100101 avec le 1 le plus à gauche et on soustrait :

$$\begin{array}{rl} & 100011000000 \\ - & 100101 \\ \hline & 000110000000 \end{array}$$

    Attention, il n'y a pas de retenue !
  - On recommence jusqu'à que le nombre n'ait que 5 chiffres

Variante

$$
\begin{array}{rr}
 & 100011000000 \\
- & 100101 \\
\hline
 & 000110000000 \\
- & \quad 100101 \\
\hline
 & \quad 010101000 \\
- & \quad 100101 \\
\hline
 & \quad 0111100 \\
- & \quad 100101 \\
\hline
 & \quad 011001
\end{array}
$$

Propriétés

Si le polynôme $G(x)$ est bien choisi :

- Faire une erreur sur un bit revient à ajouter un x^{qqch} qui n'est pas divisible par $G(x)$: on la détecte
- Faire une erreur sur deux bits revient à ajouter un $x^{qqch} + x^{qqch}$ qui n'est sans doute pas divisible par $G(x)$: on la détecte aussi

Tout dépend du polynôme !

Ethernet

Polynôme pour Ethernet :

$$G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

- Ajoute donc 32 bits
- Distance de Hamming de 4 (détecte 3 erreurs, en corrige une)
- On aurait pu faire $d = 6$ avec un polynôme mieux pensé

Contenu

La semaine prochaine

- On reste sur la couche liaison
- On s'intéresse aux réseaux locaux, et réseaux à diffusion