Du Dobble aux codes correcteurs d'erreurs

Bruno Grenet

Équipe ECO (Exact Computing) – LIRMM

24 novembre 2016

$\Longrightarrow$

Équipe ECO

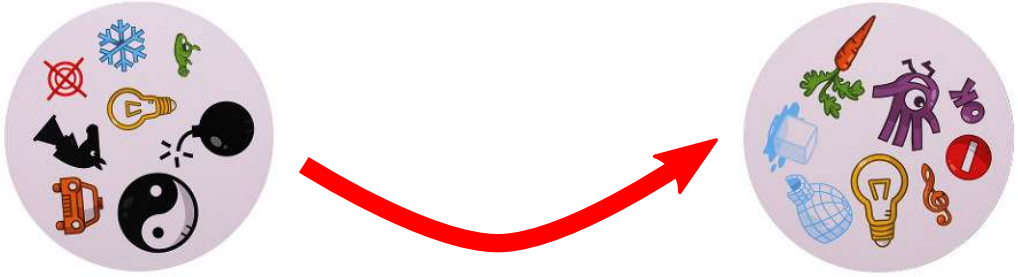
- À l'interface entre maths et info : calcul exact
- Thématiques :
  - Calcul formel (polynômes, algèbre linéaire, etc.)
  - Cryptographie
  - Codes correcteurs
- Membres permanents :
  - Pascal Giorgi (MCf UM)
  - Bruno Grenet (MCf UM)
  - Eleonora Guerrini (MCf UM)
  - Laurent Imbert (DR CNRS)
  - Romain Lebreton (MCf UM)
- Détails :
  `http://www.lirmm.fr/eco/`

Dobble : Qu'es aquò ?

Dobble : Qu'es aquò ?

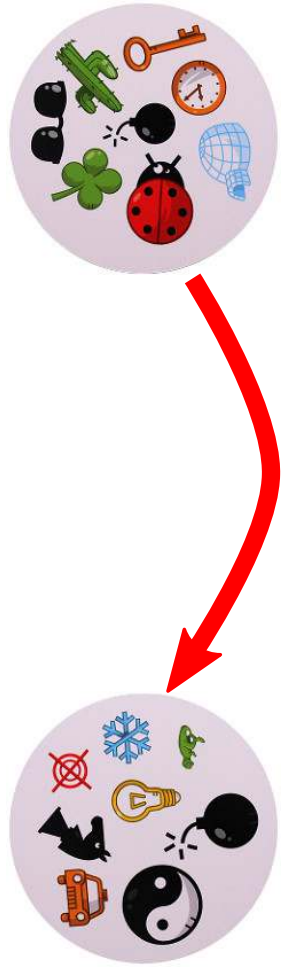
Dobble : Qu'es aquò ?

Dobble : Qu'es aquò ?



Dobble : Qu'es aquò ?

Dobble : Qu'es aquò ?

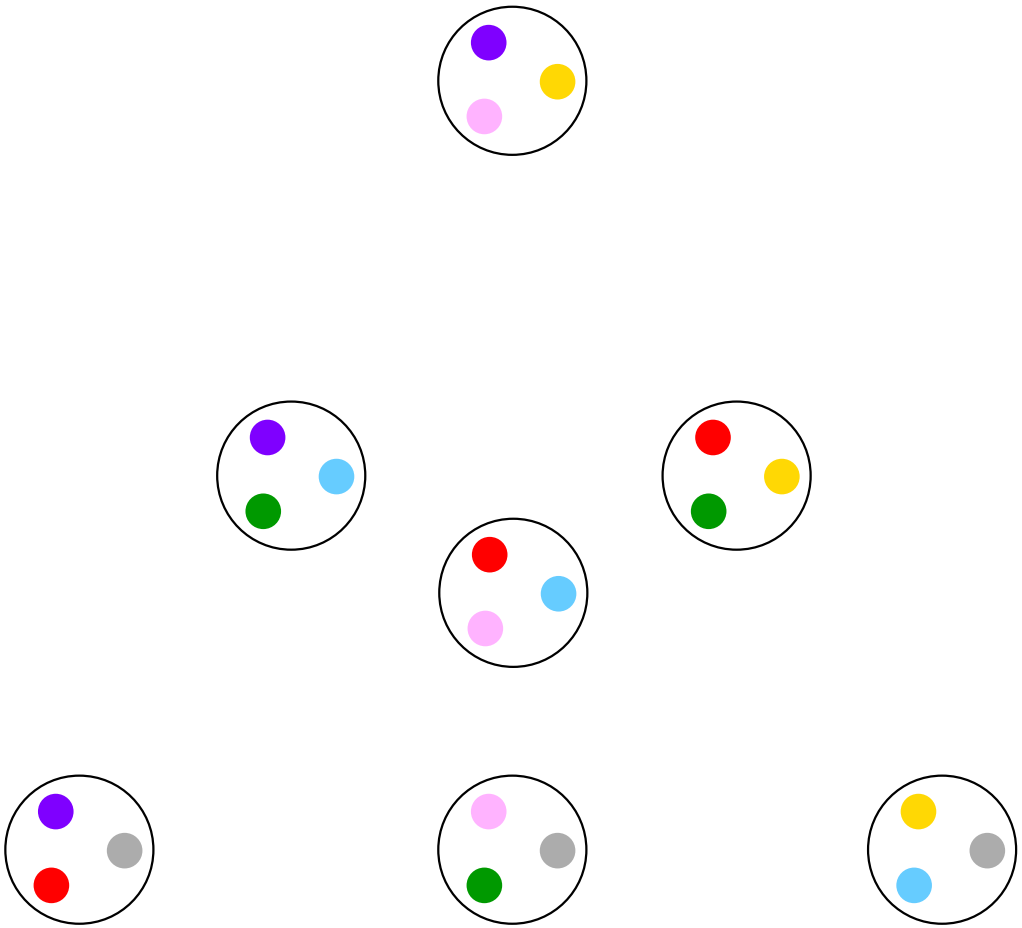


Dobble : Qu'es aquò ?

Dobble : Qu'es aquò ?

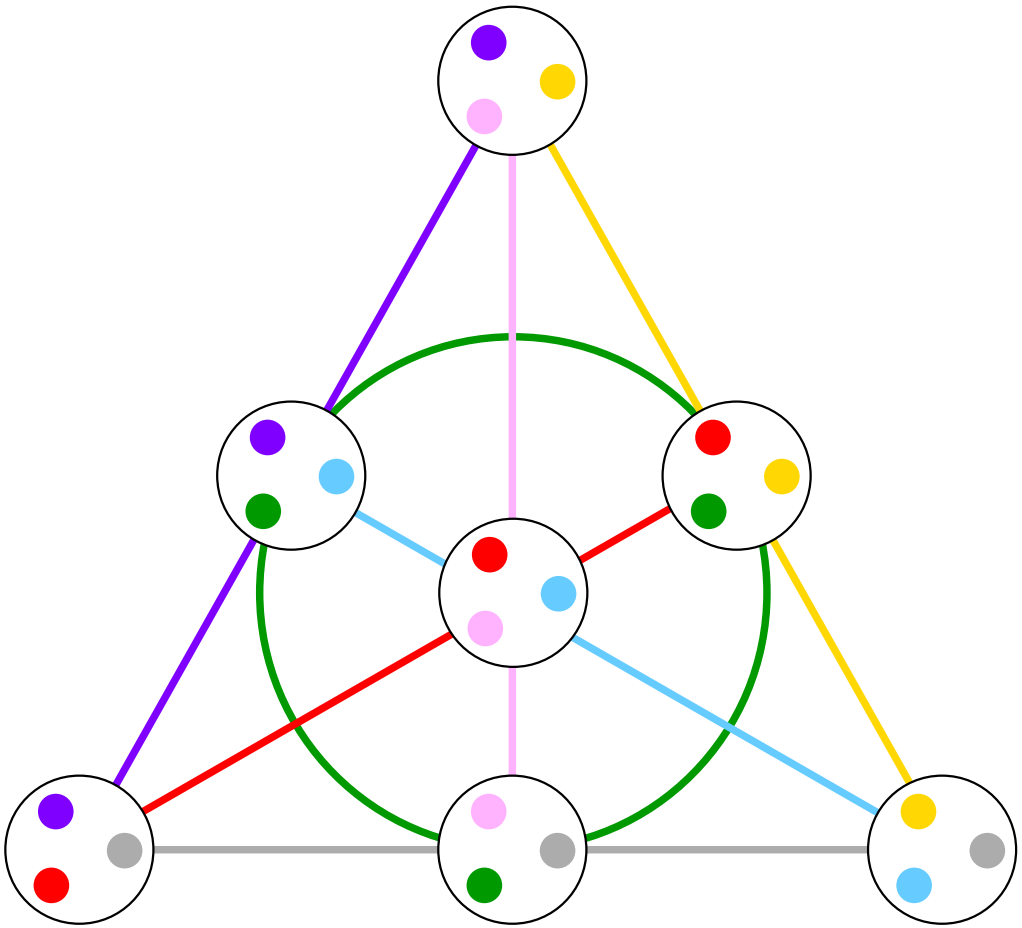
- Chaque joueur a une carte avec 8 symboles
- Chaque couple de cartes a exactement un symbole en commun
- Plusieurs jeux à partir de ce principe

Dobble et plan de Fano



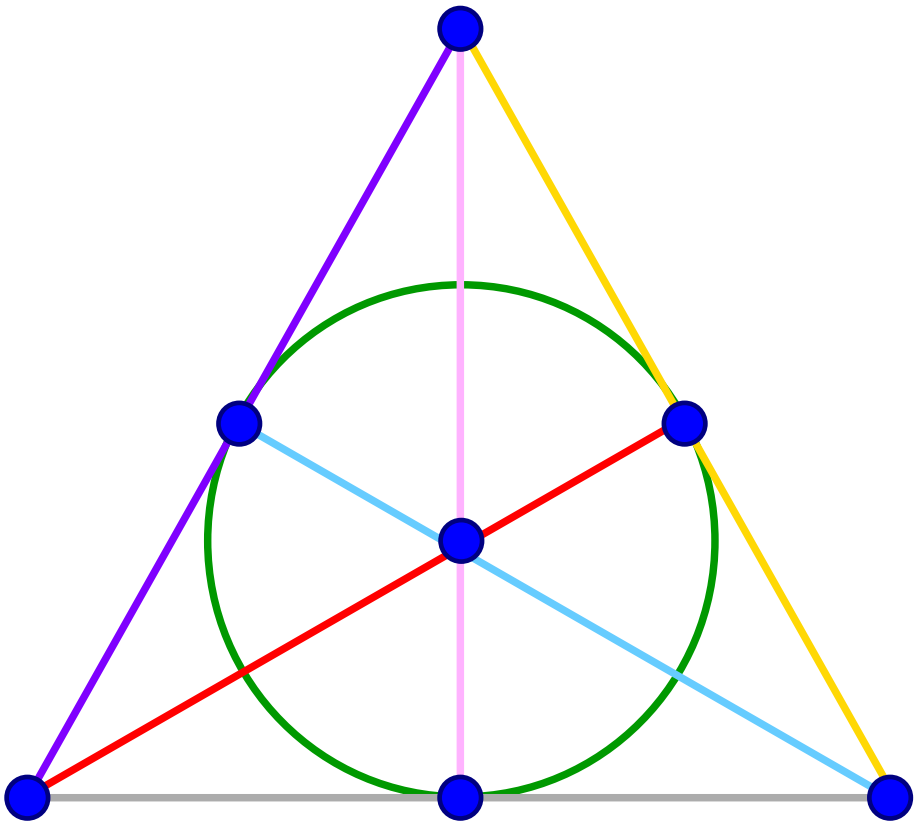
- 7 symboles, 7 cartes

Dobble et plan de Fano



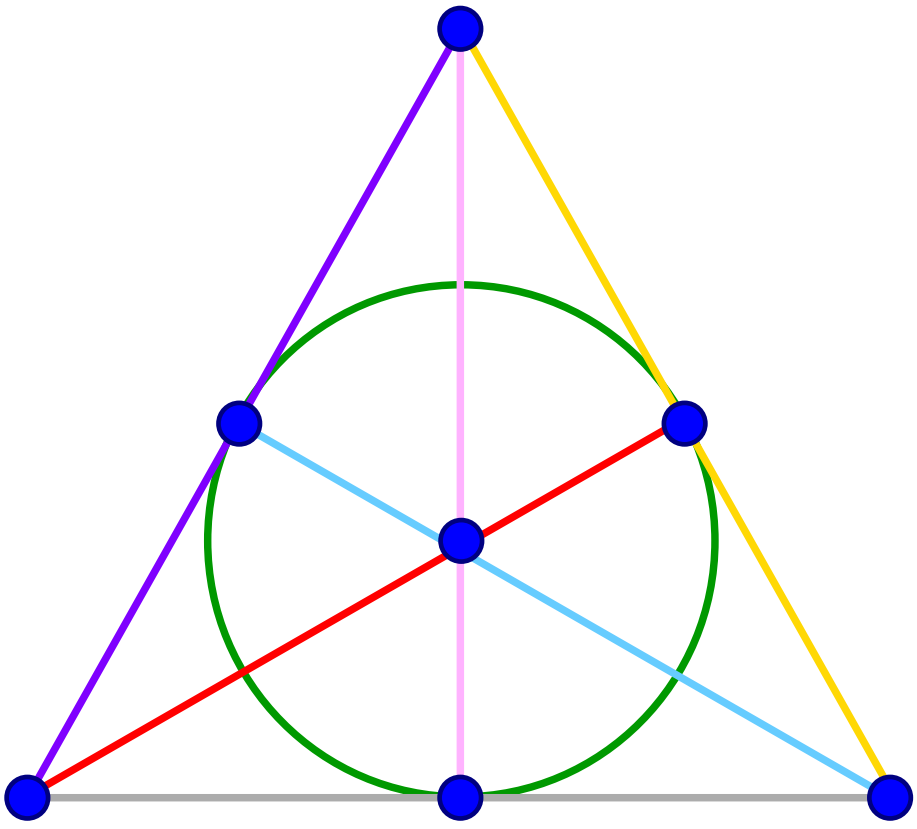
- 7 symboles, 7 cartes
- carte = point
- symbole = droite

Dobble et plan de Fano



- 7 symboles, 7 cartes
- carte = point
- symbole = droite

Dobble et plan de Fano

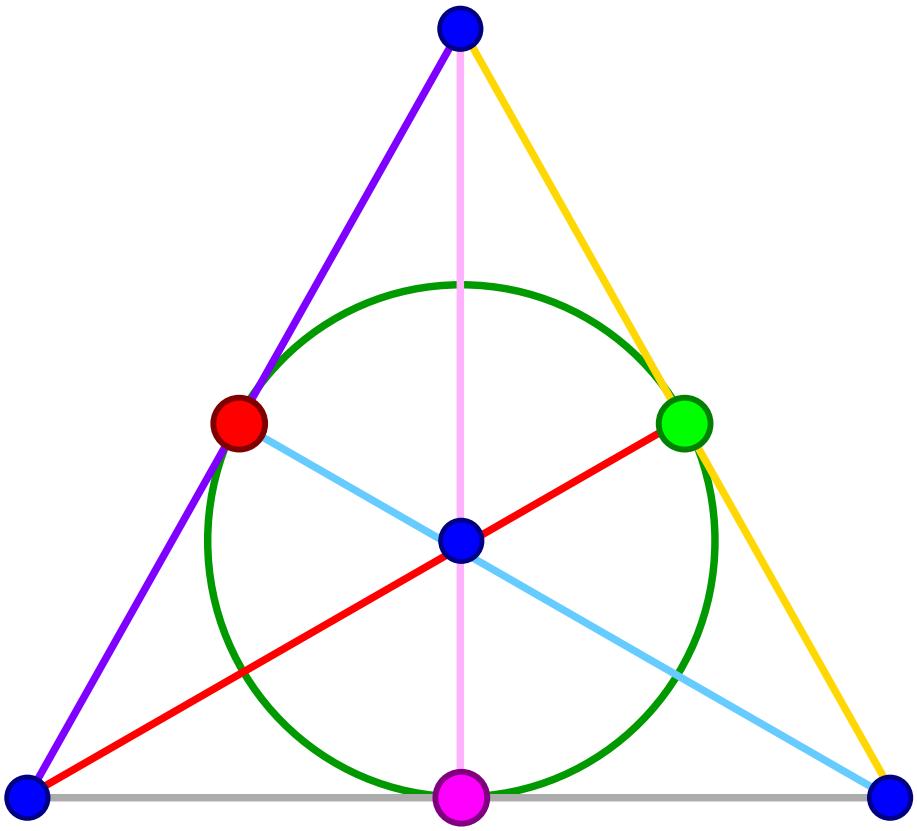


- 7 symboles, 7 cartes
- carte = point
- symbole = droite

- loi de groupe

$$\forall \bullet, \quad \bullet + \bullet = \varnothing$$

Dobble et plan de Fano



- 7 symboles, 7 cartes
- carte = point
- symbole = droite

- loi de groupe

$\forall$ ●, ● + ● = ∅

● + ● + ● = ∅

Codes correcteurs : Qu'es aquò ?

Codes correcteurs : Qu'es aquò ?

Codes correcteurs : Qu'es aquò ?

But : ajouter de la redondance pour repérer et corriger les erreurs

Paramètres des codes correcteurs

$$\begin{aligned} \Sigma^k &\longrightarrow \Gamma^n \\ m &\longmapsto c \end{aligned}$$

Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

Paramètres des codes correcteurs

$$\begin{aligned} \{0,1\}^k &\longrightarrow \{0,1\}^n \\ m &\longmapsto c \end{aligned}$$

But : corriger une erreur, où qu'elle soit

Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

But : corriger une erreur, où qu'elle soit

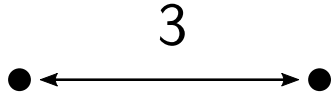
- Tripler chaque bit :
  - $0 \mapsto 000$, $1 \mapsto 111$
  - $k = 1$, $n = 3$ (ratio $1/3$)

Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

But : corriger une erreur, où qu'elle soit

- Tripler chaque bit :
  - $0 \mapsto 000$, $1 \mapsto 111$
  - $k = 1$, $n = 3$ (ratio $1/3$)
- Distance entre deux mots de code ≥ 3

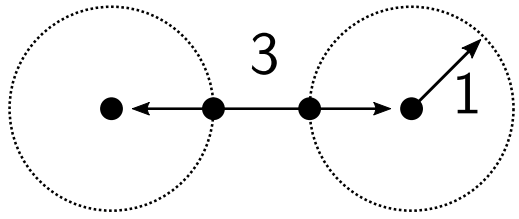


Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

But : corriger une erreur, où qu'elle soit

- Tripler chaque bit :
  - $0 \mapsto 000$, $1 \mapsto 111$
  - $k = 1$, $n = 3$ (ratio $1/3$)
- Distance entre deux mots de code ≥ 3

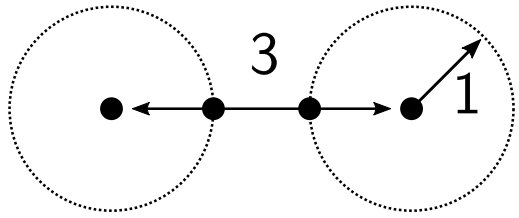


Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

But : corriger une erreur, où qu'elle soit

- Tripler chaque bit :
  - $0 \mapsto 000$, $1 \mapsto 111$
  - $k = 1$, $n = 3$ (ratio $1/3$)
- Distance entre deux mots de code ≥ 3



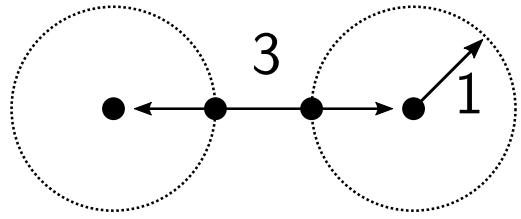
$$2^k(n+1) \leq 2^n$$

Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

But : corriger une erreur, où qu'elle soit

- Tripler chaque bit :
  - $0 \mapsto 000$, $1 \mapsto 111$
  - $k = 1$, $n = 3$ (ratio $1/3$)
- Distance entre deux mots de code ≥ 3



$$2^k(n+1) \leq 2^n$$

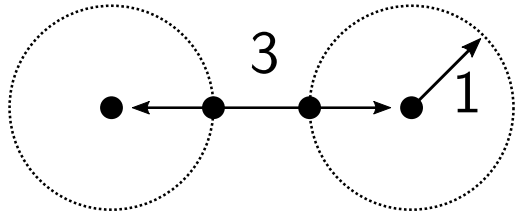
k	1	2	3	4	5	6	7	8	9	10	11	12
n	3	5	6	7	9	10	11	12	13	14	15	17

Paramètres des codes correcteurs

$$\{0,1\}^k \longrightarrow \{0,1\}^n$$
$$m \longmapsto c$$

But : corriger une erreur, où qu'elle soit

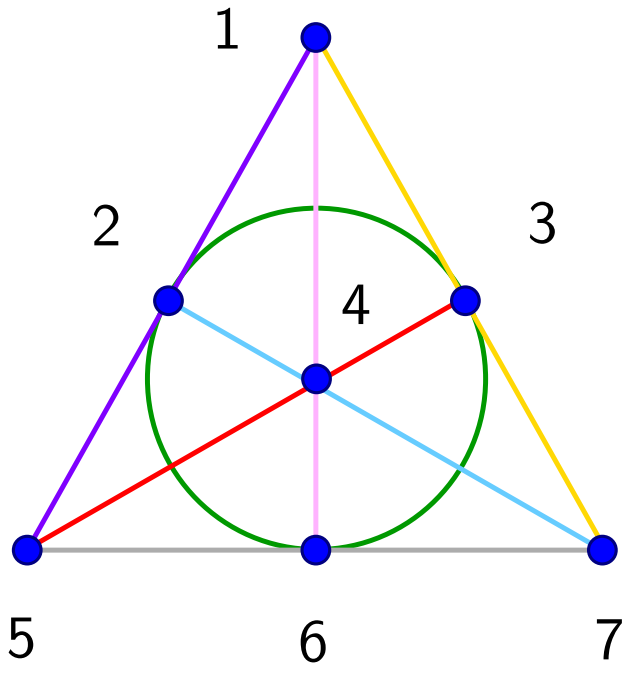
- Tripler chaque bit :
  - $0 \mapsto 000$, $1 \mapsto 111$
  - $k = 1$, $n = 3$ (ratio $1/3$)
- Distance entre deux mots de code ≥ 3



$$2^k(n+1) \leq 2^n$$

k	1	2	3	4	5	6	7	8	9	10	11	12
n	3	5	6	7	9	10	11	12	13	14	15	17

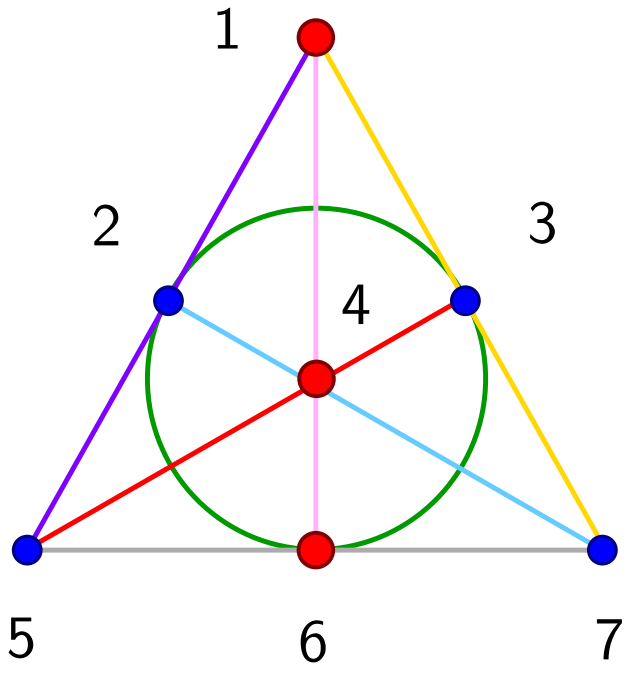
Plan de Fano et code de Hamming



Mots = ensembles de points

0000000

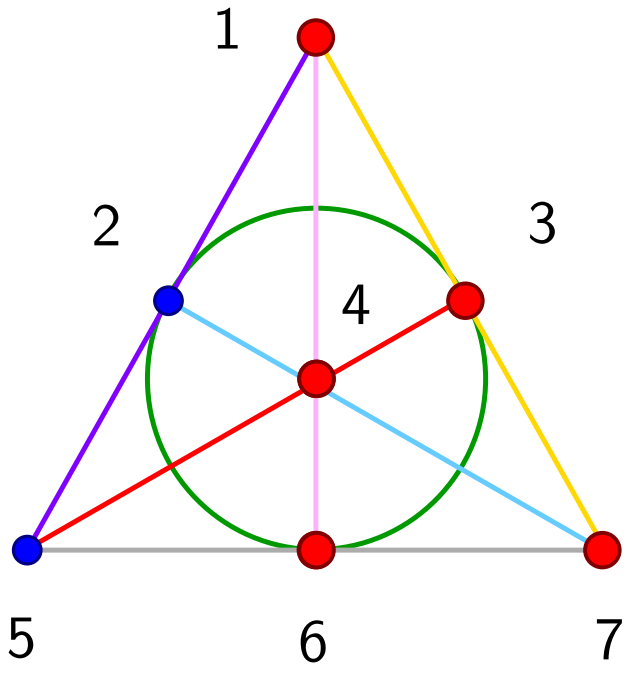
Plan de Fano et code de Hamming



Mots = ensembles de points

1001010

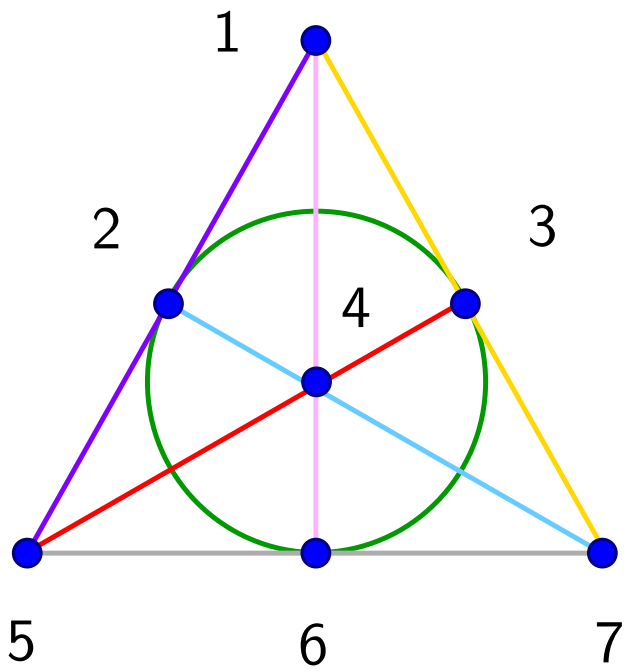
Plan de Fano et code de Hamming



Mots = ensembles de points

1011011

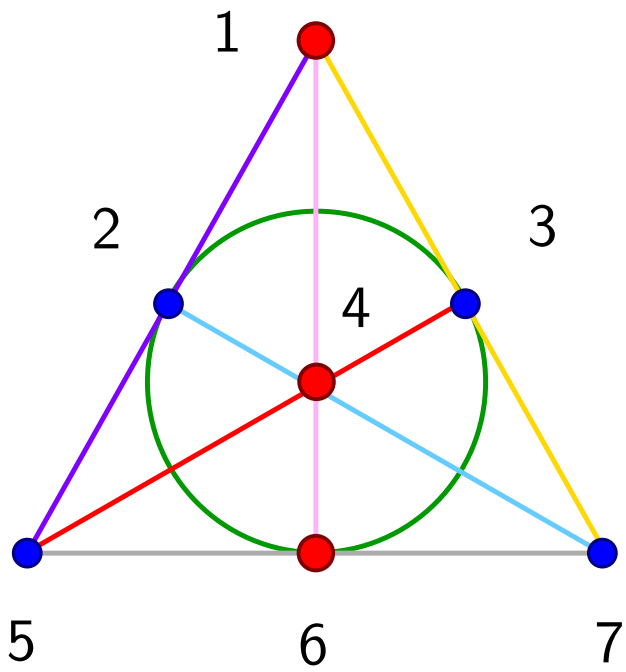
Plan de Fano et code de Hamming



Mots = ensembles de points

Mots de code = somme nulle

Plan de Fano et code de Hamming

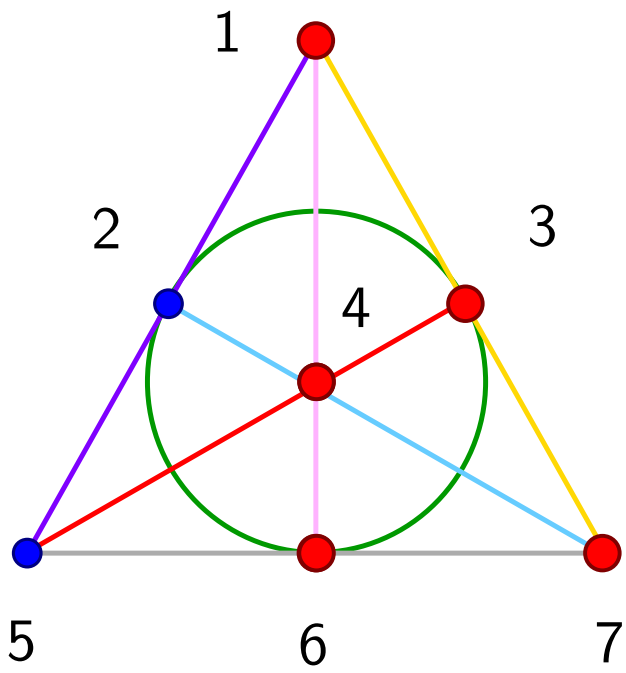


Mots = ensembles de points

Mots de code = somme nulle

1001010

Plan de Fano et code de Hamming

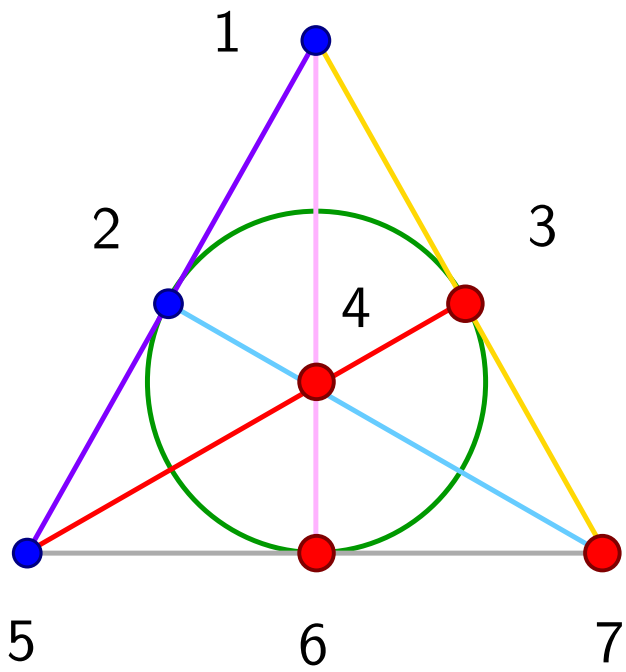


Mots = ensembles de points

Mots de code = somme nulle

1011011

Plan de Fano et code de Hamming

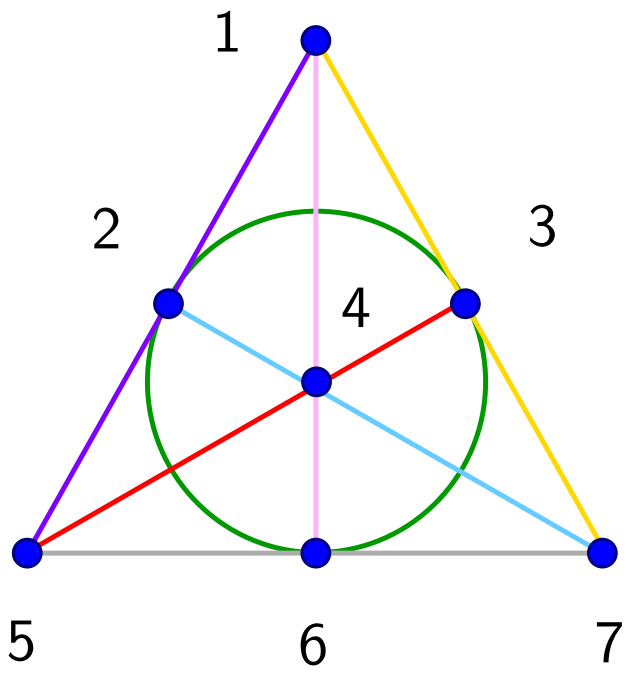


Mots = ensembles de points

Mots de code = somme nulle

0011011

Plan de Fano et code de Hamming



Mots = ensembles de points
Mots de code = somme nulle

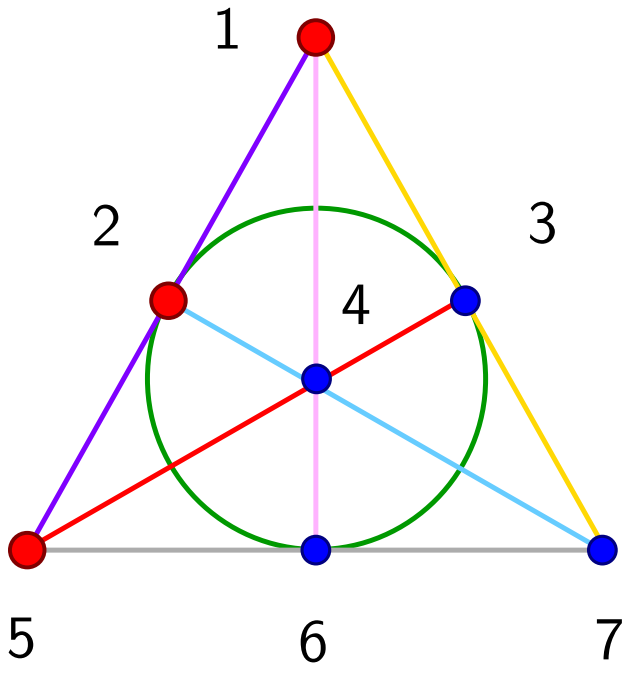
0011011

$\rightsquigarrow 16 = 2^4$ mots de code

Distance 3

Passage d'un mot à un autre : inversion d'une (ou $+$) droites

Plan de Fano et code de Hamming



Mots = ensembles de points
Mots de code = somme nulle

0011011

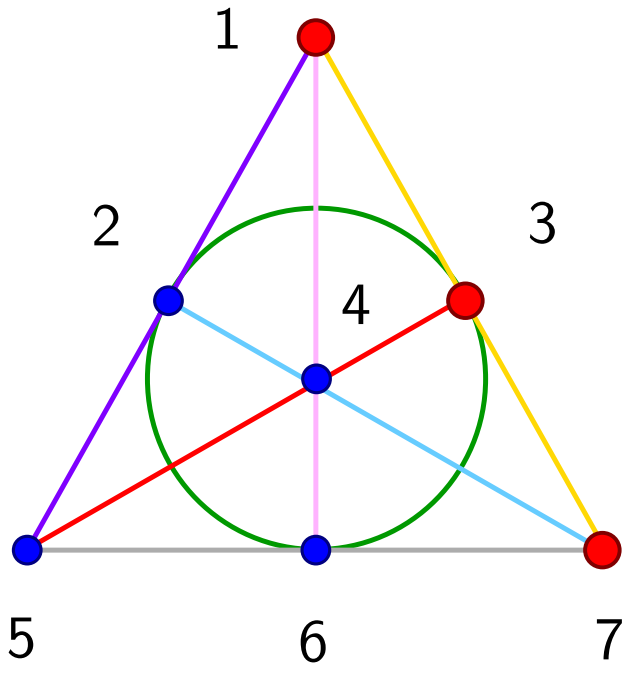
$\rightsquigarrow 16 = 2^4$ mots de code

Distance 3

Passage d'un mot à un autre : inversion d'une (ou +) droites

(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,0,1)
↓	↓	↓	↓
1100100	1010001	0000111	1001101

Plan de Fano et code de Hamming



Mots = ensembles de points
Mots de code = somme nulle

0011011

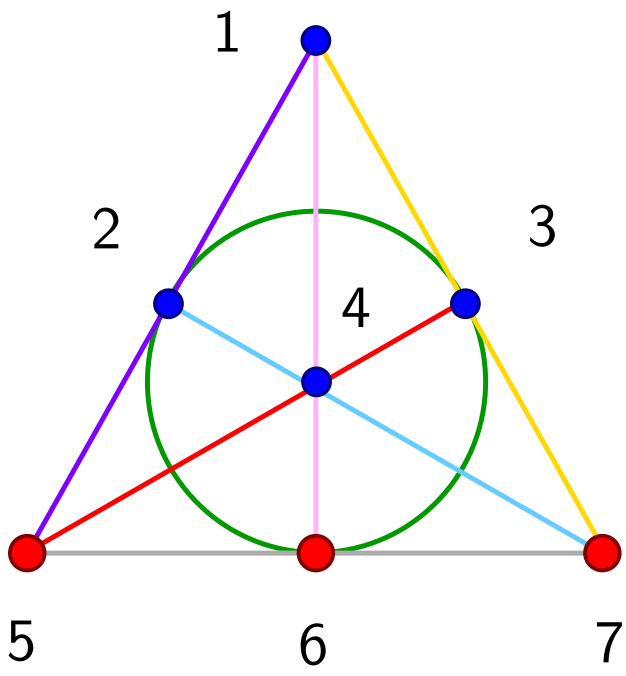
$\rightsquigarrow 16 = 2^4$ mots de code

Distance 3

Passage d'un mot à un autre : inversion d'une (ou +) droites

(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,0,1)
↓	↓	↓	↓
1100100	1010001	0000111	1001101

Plan de Fano et code de Hamming



Mots = ensembles de points
Mots de code = somme nulle

0011011

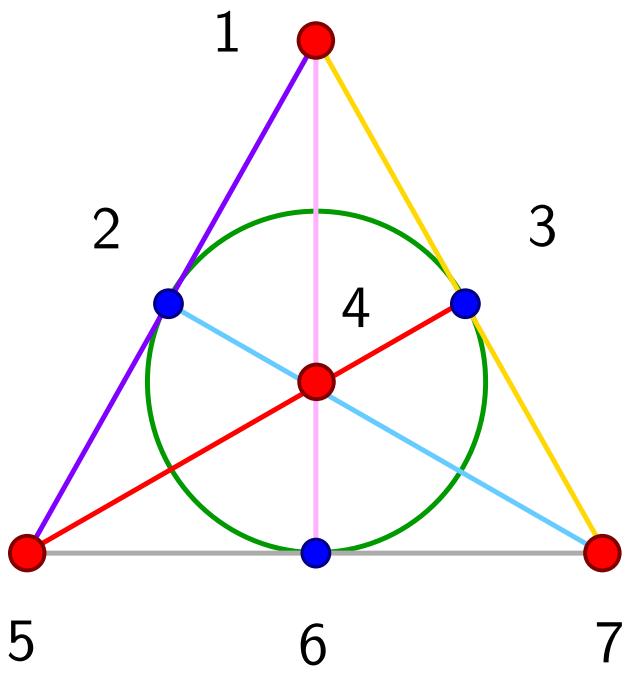
$\rightsquigarrow 16 = 2^4$ mots de code

Distance 3

Passage d'un mot à un autre : inversion d'une (ou +) droites

(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,0,1)
↓	↓	↓	↓
1100100	1010001	0000111	1001101

Plan de Fano et code de Hamming



Mots = ensembles de points
Mots de code = somme nulle

0011011

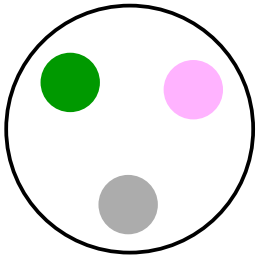
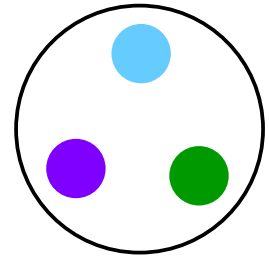
$\rightsquigarrow 16 = 2^4$ mots de code

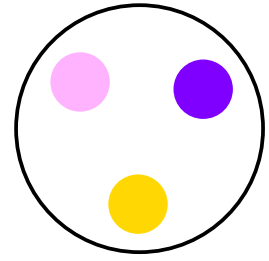
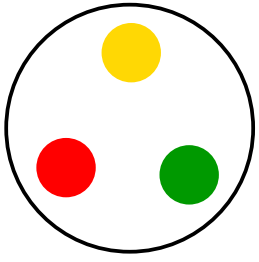
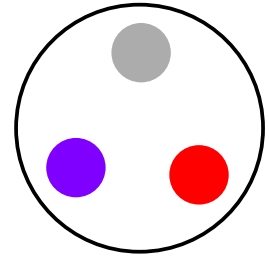
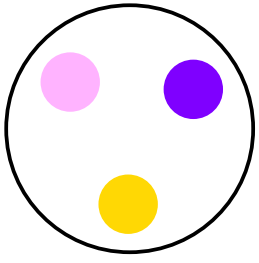
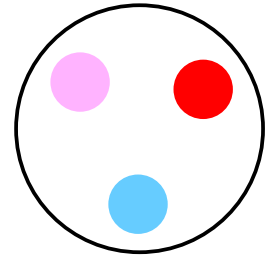
Distance 3

Passage d'un mot à un autre : inversion d'une (ou +) droites

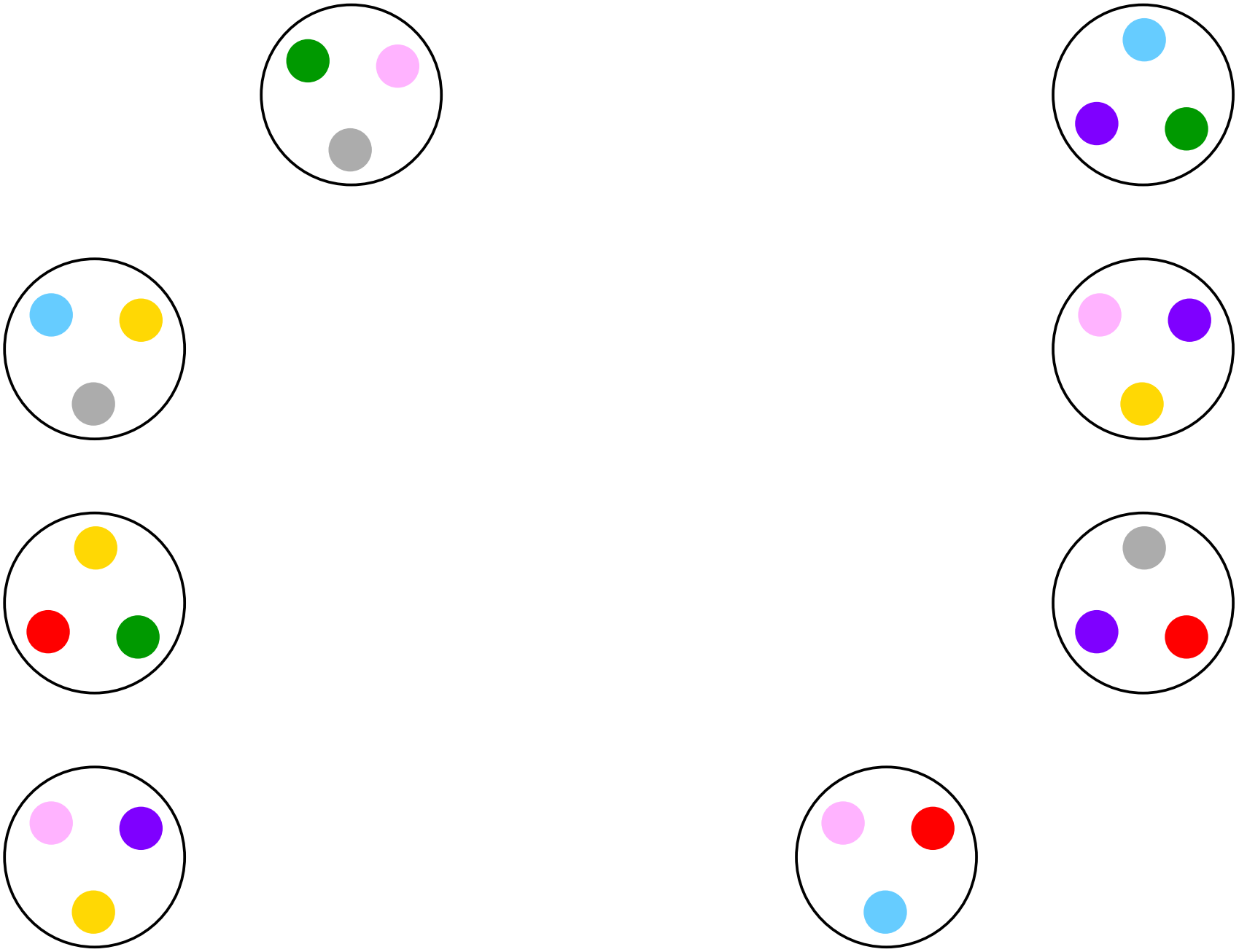
(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,0,1)
↓	↓	↓	↓
1100100	1010001	0000111	1001101

Dobble comme code de Hamming

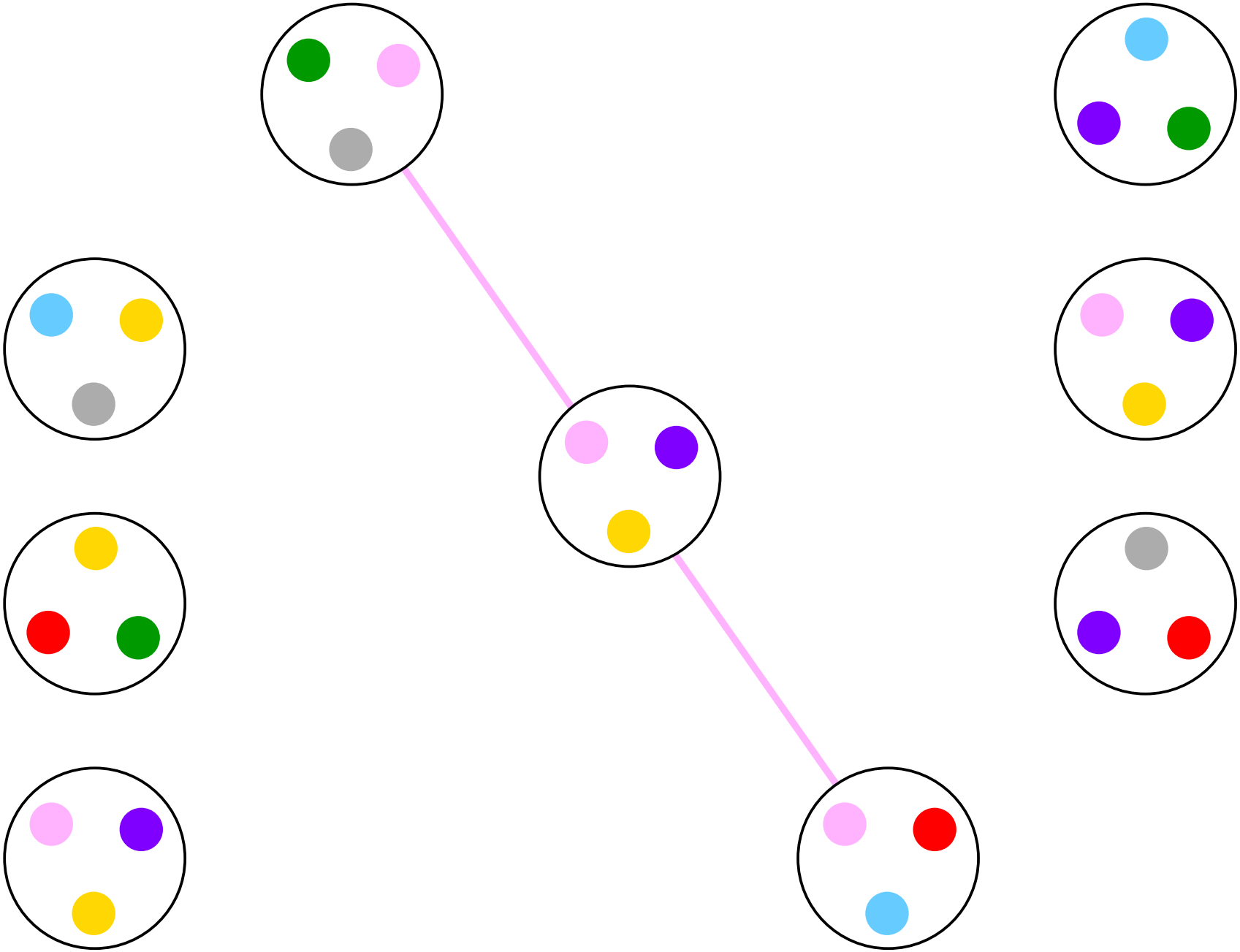



Dobble comme code de Hamming



Dobble comme code de Hamming



Et si on veut corriger plus d'une erreur ?

Code de Reed-Solomon : fixons $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$

$$\begin{aligned} \mathbb{F}_q[X]_{<k} &\longrightarrow \mathbb{F}_q^n \\ f &\longmapsto (f(\alpha_1), \ldots, f(\alpha_n)) \end{aligned}$$

Et si on veut corriger plus d'une erreur ?

Code de Reed-Solomon : fixons $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$

$$\begin{aligned} \mathbb{F}_q[X]_{<k} &\longrightarrow \mathbb{F}_q^n \\ f &\longmapsto (f(\alpha_1), \ldots, f(\alpha_n)) \end{aligned}$$

- Distance : $d = n - k + 1$

Et si on veut corriger plus d'une erreur ?

Code de Reed-Solomon : fixons $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$

$$\begin{aligned} \mathbb{F}_q[X]_{<k} &\longrightarrow \mathbb{F}_q^n \\ f &\longmapsto (f(\alpha_1), \ldots, f(\alpha_n)) \end{aligned}$$

- Distance : $d = n - k + 1$
- Erreurs corrigeables : $e = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor$

Et si on veut corriger plus d'une erreur ?

Code de Reed-Solomon : fixons $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$

$$\begin{aligned} \mathbb{F}_q[X]_{<k} &\longrightarrow \mathbb{F}_q^n \\ f &\longmapsto (f(\alpha_1), \ldots, f(\alpha_n)) \end{aligned}$$

- Distance : $d = n - k + 1$
- Erreurs corrigeables : $e = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor$

Comment corriger autant d'erreurs ?

Décodage des codes de Reed-Solomon I

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

Décodage des codes de Reed-Solomon I

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- Interpolation de Lagrange : $k = n$ et $e = 0$

Décodage des codes de Reed-Solomon I

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- Interpolation de Lagrange : $k = n$ et $e = 0$
- Pour certains n-uplets, aucune solution !

Décodage des codes de Reed-Solomon I

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- Interpolation de Lagrange : $k = n$ et $e = 0$
- Pour certains n-uplets, aucune solution !
- Si on connaît les bons indices : il suffit de k points ($e = n - k$)

Décodage des codes de Reed-Solomon I

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- Interpolation de Lagrange : $k = n$ et $e = 0$
- Pour certains n-uplets, aucune solution !
- Si on connaît les bons indices : il suffit de k points ($e = n - k$)
- Si f existe, on peut le retrouver dès que $e \leq \lfloor \frac{n-k}{2} \rfloor$

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- On pose $E(X) = \prod_{i : f(\alpha_i) \neq y_i} (X - \alpha_i)$ et $N(X) = f(X)E(X)$

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- On pose $E(X) = \prod_{i : f(\alpha_i) \neq y_i} (X - \alpha_i)$ et $N(X) = f(X)E(X)$
- Alors pour tout i, $y_i E(\alpha_i) = N(\alpha_i)$ $(\star)$

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- On pose $E(X) = \prod_{i : f(\alpha_i) \neq y_i} (X - \alpha_i)$ et $N(X) = f(X)E(X)$
- Alors pour tout i, $\qquad y_i E(\alpha_i) = N(\alpha_i) \qquad (\star)$
- Toute solution $(E^\star, N^\star)$ de $(\star)$ satisfait $N^\star / E^\star = f$

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- On pose $E(X) = \prod_{i : f(\alpha_i) \neq y_i} (X - \alpha_i)$ et $N(X) = f(X)E(X)$
- Alors pour tout i, $\qquad y_i E(\alpha_i) = N(\alpha_i) \qquad (\star)$
- Toute solution $(E^\star, N^\star)$ de $(\star)$ satisfait $N^\star / E^\star = f$
- Résolution par algèbre linéaire dès que $2e + k < n$:

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- On pose $E(X) = \prod_{i : f(\alpha_i) \neq y_i} (X - \alpha_i)$ et $N(X) = f(X)E(X)$
- Alors pour tout i, $y_i E(\alpha_i) = N(\alpha_i)$ $(\star)$
- Toute solution $(E^\star, N^\star)$ de $(\star)$ satisfait $N^\star / E^\star = f$
- Résolution par algèbre linéaire dès que $2e + k < n$:
  - $e + (e + k)$ inconnues

Décodage des codes de Reed-Solomon II

Problème
Entrée : $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$
Sortie : $f \in \mathbb{F}_q[X]_{<k}$ tel que $f(\alpha_i) = y_i$ pour $\geq n - e$ indices

- On pose $E(X) = \prod_{i : f(\alpha_i) \neq y_i} (X - \alpha_i)$ et $N(X) = f(X)E(X)$
- Alors pour tout i, $\qquad y_i E(\alpha_i) = N(\alpha_i) \qquad (\star)$
- Toute solution $(E^\star, N^\star)$ de $(\star)$ satisfait $N^\star / E^\star = f$
- Résolution par algèbre linéaire dès que $2e + k < n$:
  - $e + (e + k)$ inconnues
  - n équations

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.
- Plus d'erreurs : décodage en liste

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.
- Plus d'erreurs : décodage en liste
- Plus petits alphabets, meilleurs ratios $k/n \rightsquigarrow$ nouveaux codes

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.
- Plus d'erreurs : décodage en liste
- Plus petits alphabets, meilleurs ratios $k/n \rightsquigarrow$ nouveaux codes

Dans l'équipe ECO :

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.
- Plus d'erreurs : décodage en liste
- Plus petits alphabets, meilleurs ratios $k/n \rightsquigarrow$ nouveaux codes

Dans l'équipe ECO :

- Étude théorique des paramètres admissibles

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.
- Plus d'erreurs : décodage en liste
- Plus petits alphabets, meilleurs ratios $k/n \rightsquigarrow$ nouveaux codes

Dans l'équipe ECO :

- Étude théorique des paramètres admissibles
- Algèbre linéaire rapide

Aller plus loin : où va la recherche ?

- Calcul efficace : algèbre linéaire rapide, interpolation avec erreurs, etc.
- Plus d'erreurs : décodage en liste
- Plus petits alphabets, meilleurs ratios $k/n \rightsquigarrow$ nouveaux codes

Dans l'équipe ECO :

- Étude théorique des paramètres admissibles
- Algèbre linéaire rapide
- Algorithmique des polynômes

Merci de votre attention !