

## Codes de Gray

**Florent Hivert**

Mél : Florent.Hivert@lri.fr

Adresse universelle : <http://www.lri.fr/~hivert>

## Objectifs : Complexité optimale pour iter

L'ordre lexicographique est pratique pour faire de la génération combinatoire, **mais** ...

### Retenir

*Les algorithmes d'itérations vue en cours sont pour la plupart en temps constant amorti (CAT). On peut trouver des algorithmes réellement en temps constant ! Pour ceci, il faut changer l'ordre d'énumération.*

## Exemple : les chaines de bits

- Soit  $B_n$  le nombre de bits écrit lors de l'itération le long de la liste lexicographique des mots binaires de longueurs  $n$ .

$$B_0 = 0, \quad B_1 = 2, \quad B_n = 2B_{n-1} + 2$$

- On trouve :  $B_n = 2^{n+1} - 2$  pour  $2^n$  chaîne de bits.
- En moyenne  $\frac{2^{n+1}-2}{2^n} \approx 2$  bits par itération.

## Exemple : les chaines de bits

- Soit  $B_n$  le nombre de bits écrit lors de l'itération le long de la liste lexicographique des mots binaires de longueurs  $n$ .

$$B_0 = 0, \quad B_1 = 2, \quad B_n = 2B_{n-1} + 2$$

- On trouve :  $B_n = 2^{n+1} - 2$  pour  $2^n$  chaîne de bits.
- En moyenne  $\frac{2^{n+1}-2}{2^n} \approx 2$  bits par itération.

## Exemple : les chaines de bits

- Soit  $B_n$  le nombre de bits écrit lors de l'itération le long de la liste lexicographique des mots binaires de longueurs  $n$ .

$$B_0 = 0, \quad B_1 = 2, \quad B_n = 2B_{n-1} + 2$$

- On trouve :  $B_n = 2^{n+1} - 2$  pour  $2^n$  chaîne de bits.
- En moyenne  $\frac{2^{n+1}-2}{2^n} \approx 2$  bits par itération.

# Algorithme en temps constant !

## Retenir

*Il existe des algorithmes en temps constant (non amorti) ! Ce sont des algorithmes très simple et très courts : pas de boucle !*

## Problem

*D'un objet au suivant, on veut changer le minimum de chose.*

Exemple : 1 bits

# Algorithme en temps constant !

## Retenir

*Il existe des algorithmes en temps constant (non amorti) ! Ce sont des algorithmes très simple et très courts : pas de boucle !*

## Problem

*D'un objet au suivant, on veut changer le minimum de chose.*

Exemple : 1 bits

# Algorithme en temps constant !

## Retenir

*Il existe des algorithmes en temps constant (non amorti) ! Ce sont des algorithmes très simple et très courts : pas de boucle !*

## Problem

*D'un objet au suivant, on veut changer le minimum de chose.*

Exemple : 1 bits



# Code de Gray combinatoire

## Définition

*Soit  $E$  un ensemble à énumérer. Soit  $P$  une relation sur  $E$ , dite de **proximité**. Un code de code de Gray combinatoire pour le système  $(E, P)$  est une liste  $S = s_1, s_2, \dots, s_N$ , où  $N = |E|$ , tel que chaque élément de  $E$  apparaît une fois et une seule dans  $S$  et  $(s_i, s_{i+1}) \in P$  pour tout  $i = 1, \dots, N - 1$ .*

- Dans la plupart des cas la relation  $P$  est symétrique.
- Problème de théorie des graphes (chemin Hamiltonien).
- On dit que le code est cyclique si de plus  $(S_n, s_0) \in P$ .

# Code de Gray combinatoire

## Définition

*Soit  $E$  un ensemble à énumérer. Soit  $P$  une relation sur  $E$ , dite de **proximité**. Un code de code de Gray combinatoire pour le système  $(E, P)$  est une liste  $S = s_1, s_2, \dots, s_N$ , où  $N = |E|$ , tel que chaque élément de  $E$  apparaît une fois et une seule dans  $S$  et  $(s_i, s_{i+1}) \in P$  pour tout  $i = 1, \dots, N - 1$ .*

- Dans la plupart des cas la relation  $P$  est symétrique.
- Problème de théorie des graphes (chemin Hamiltonien).
- On dit que le code est cyclique si de plus  $(s_n, s_0) \in P$ .

# Code de Gray combinatoire

## Définition

*Soit  $E$  un ensemble à énumérer. Soit  $P$  une relation sur  $E$ , dite de **proximité**. Un code de code de Gray combinatoire pour le système  $(E, P)$  est une liste  $S = s_1, s_2, \dots, s_N$ , où  $N = |E|$ , tel que chaque élément de  $E$  apparaît une fois et une seule dans  $S$  et  $(s_i, s_{i+1}) \in P$  pour tout  $i = 1, \dots, N - 1$ .*

- Dans la plupart des cas la relation  $P$  est symétrique.
- Problème de théorie des graphes (chemin Hamiltonien).
- On dit que le code est cyclique si de plus  $(S_n, s_0) \in P$ .

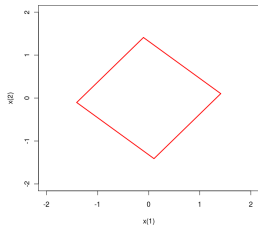
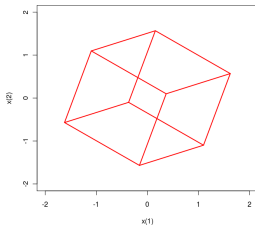
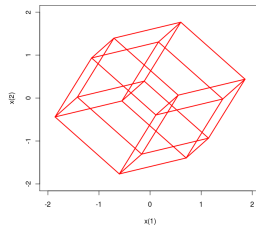
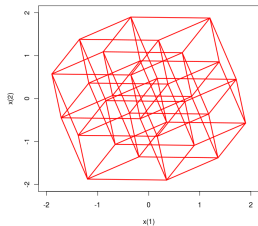
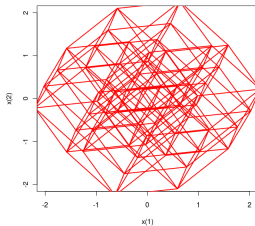
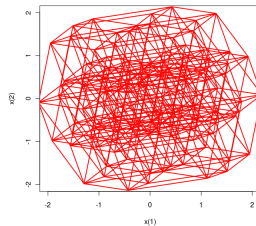
## Code de Gray combinatoire

### Définition

*Soit  $E$  un ensemble à énumérer. Soit  $P$  une relation sur  $E$ , dite de **proximité**. Un code de code de Gray combinatoire pour le système  $(E, P)$  est une liste  $S = s_1, s_2, \dots, s_N$ , où  $N = |E|$ , tel que chaque élément de  $E$  apparaît une fois et une seule dans  $S$  et  $(s_i, s_{i+1}) \in P$  pour tout  $i = 1, \dots, N - 1$ .*

- Dans la plupart des cas la relation  $P$  est symétrique.
- Problème de théorie des graphes (chemin Hamiltonien).
- On dit que le code est cyclique si de plus  $(S_n, s_0) \in P$ .

# Les hypercubes

**2-d hypercube****3-d hypercube****4-d hypercube****5-d hypercube****6-d hypercube****7-d hypercube**

# Code binaire réfléchi

Frank Gray 1953

## Définition

*Idée : on retourne la liste pour la deuxième partie :*

- $G_0 = [\epsilon]$
- $G_n = 0 \cdot G_{n-1} + 1 \cdot \overline{G_{n-1}}$

0, 1

00, 01, 11, 10

000, 001, 011, 010, 110, 111, 101, 100

## Codage vers le code Gray

### Retenir

*Une formule étonnamment simple*

$$G = i \oplus (i \gg 1)$$

*où  $\oplus$  désigne le Ou-Exclusif bit à bit.*

# Passage d'une liste de bits à la suivante en temps constant

Idée : noter la position du prochain zéro :

Retenir

$$\text{Pos}_i := \begin{cases} i & \text{si } b_i = 0 \text{ ou } b_{i-1} = 1 \\ \min\{k \mid b_k = 0 \text{ et } k > i\} & \text{si } b_i = 1 \text{ et } b_{i-1} = 0 \end{cases}$$

$i$	7	6	5	4	3	2	1	0
$b_i$	1	0	1	1	1	0	1	0
$\text{Pos}_i$	8	6	5	4	6	2	2	0



## Passage d'une liste de bits à la suivante en temps constant

### Retenir

*D'un nombre au suivant, au plus trois valeurs du tableau Pos changent.*

$n$	Bin	Pos	Pos <sub>0</sub>	Gray
0	000	210	0	000
1	001	211	1	001
2	010	220	0	011
3	011	212	2	010
4	100	310	0	110
5	101	311	1	111
6	110	230	0	101
7	111	213	3	100

## Exemple : Les permutations

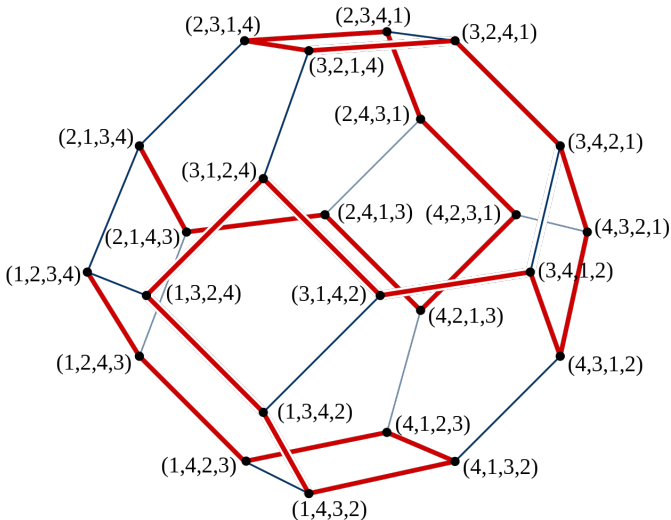
### Retenir

*Permutations par transpositions élémentaires*

- $S =$  ensemble des permutations de  $[1, 2, \dots, n]$
- Relation de proximité : soit  $S = (s_1, s_2, \dots, s_n)$ . Alors  $T$  est proche de  $S$  si on peut trouver  $i$  tels que

$$T = (s_1, s_2, \dots, s_{i+1}, s_i, \dots, s_n)$$

## Le graphe des permutations : le permutohedre



## Algorithme de Steinhaus-Johnson-Trotter

### Retenir

*Idée : récursivement insérer  $n$  à toute les places possible en alternant le sens de déplacement.*

123	1234	1243	1423	4123
132	4132	1432	1342	1324
312	3124	3142	3412	4312
321	4321	3421	3241	3214
231	2314	2341	2431	4231
213	4213	2413	2143	2134

Voir code Python...