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 algoritmes 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'itéreration le long de la liste lexicographique des mots binaires de longueurs n.

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

- 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'itéreration le long de la liste lexicographique des mots binaires de longueurs n.

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

- 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'itéreration le long de la liste lexicographique des mots binaires de longueurs n.

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

- 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 cours : pas de boucle !

Problem

D'un objet au suivant, on veux 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 cours : pas de boucle !

Problem

D'un objet au suivant, on veux 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 cours : pas de boucle !

Problem

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

Exemple : 1 bits

Code de Gray combinatoire

Définition

Soit E un ensemble à enumérer. Soit P une relation sur E, dite de ***proximité****. Un code de code de Gray combinatoire pour le système (E, P) est une liste $S = s_1, s_2, \ldots, s_N$, 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 $i = 1, \ldots N - 1$.*

- 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 enumérer. 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 enumérer. Soit P une relation sur E, dite de **proximité**. Un code de code de Gray combinatoire pour le système (E, P) est une liste $S = s_1, s_2, \ldots, s_N$, 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 $i = 1, \ldots N - 1$.*

- 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 $(S_n, s_0) \in P$.

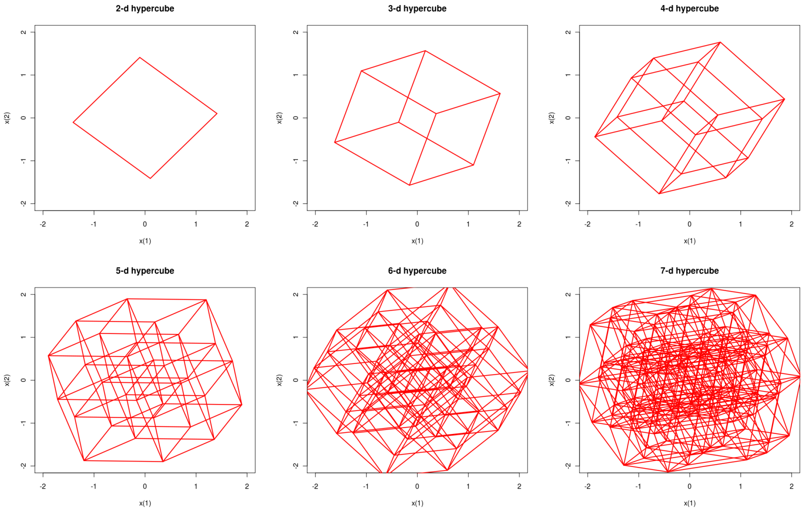
Code de Gray combinatoire

Définition

Soit E un ensemble à enumérer. Soit P une relation sur E, dite de ***proximité****. Un code de code de Gray combinatoire pour le système (E, P) est une liste $S = s_1, s_2, \ldots, s_N$, 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 $i = 1, \ldots N - 1$.*

- 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



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

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

i	7	6	5	4	3	2	1	0
b_i	1	0	1	1	1	0	1	0
Pos_i	8	6	5	4	6	2	2	0

Note : on complète par un 0 en tête si nécessaire.

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_0	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

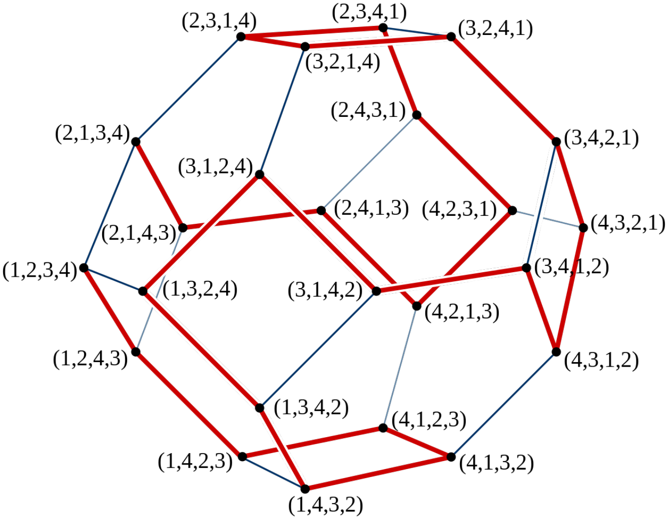
Permutations par transpositions élémentaires

- S = *ensemble des permutations de* $[1, 2, \ldots, n]$

- *Relation de proximité : soit* $S = (s_1, s_2, \ldots s_n)$. *Alors* T *est proche de* S *si on peut trouver* i *tels que*

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

Le graphe des permutations : le permutohedre



Algorithme de Steinhaus-Johnson-Trotter

Retenir

Idée : récursivement insérer n à toute les places possibles en alternant le sens de déplacement, d'une permutation à l'autre.

123	$<$	123**4**	12**4**3	1**4**23	**4**123
132	$>$	**4**132	1**4**32	13**4**2	132**4**
312	$<$	312**4**	31**4**2	3**4**12	**4**312
321	$>$	**4**321	3**4**21	32**4**1	321**4**
231	$<$	231**4**	23**4**1	2**4**31	**4**231
213	$>$	**4**213	2**4**13	21**4**3	213**4**

Voir code Python. . .