Codes de Gray

Marc Lorenzi

17 avril 2025

1 Codes de Gray binaires

1.1 Introduction

Un *bit* est un élément de l'ensemble $\mathbb{B} = \{0, 1\}$. L'ensemble $\mathbb{B}$ peut être muni de l'opération $\oplus$ d'addition modulo 2. Cette opération fait de $(\mathbb{B}, \oplus)$ un groupe abélien. Pour tout $n \in \mathbb{N}$, un *mot de n bits* est un élément de $\mathbb{B}^n$. Si $x = (b_0, \ldots, b_{n-1})$ est un mot de n bits, nous noterons plus simplement $x = b_0 \ldots b_{n-1}$. Par exemple, $x = 01001$ est un mot de 5 bits. Les bits 0, 2 et 3 de x valent 0. Les bits 1 et 4 valent 1.

Soit $n \in \mathbb{N}$. Un *code de Gray binaire* sur n bits est un 2^n-uplet $G = (G_0, \ldots, G_{2^n-1})$ contenant tous les mots de n bits et vérifiant la propriété suivante : pour tout $i \in [\![0, 2^n - 1]\!]$, G_i et G_{i+1} diffèrent d'un seul bit. Si, de plus, G_{2^n-1} et G_0 diffèrent d'un seul bit, le code G est *cyclique*.

Un code de Gray cyclique peut être interprété comme un *cycle hamiltonien* de l'hypercube. Pour tout $n \in \mathbb{N}$, l'hypercube Q_n est un graphe simple non orienté. L'ensemble de ses sommets est $\mathbb{B}^n$. Pour tous $x, y \in \mathbb{B}^n$, il existe une arête d'extrémités x et y si et seulement si x et y diffèrent d'un seul bit. Par exemple, Q_2 est un carré et Q_3 est un cube classique. La figure ci-dessous représente, en bleu, un cycle hamiltonien de Q_3. Ce cycle hamiltonien fournit 16 codes de Gray cycliques sur 3 bits (8 possibilités pour le choix de l'origine, puis 2 possibilités pour le choix du sens de parcours).

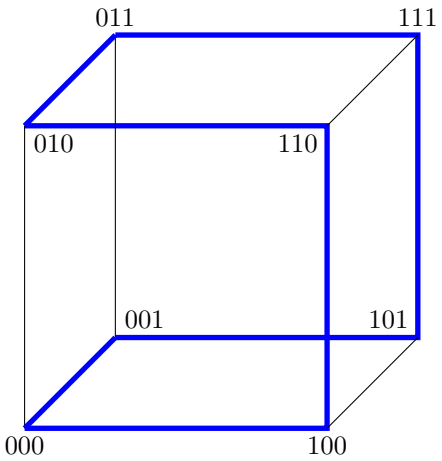

FIGURE 1 – Un parcours hamiltonien de Q_3.

1.2 Le code binaire réfléchi

Nous allons nous intéresser à une famille particulière de codes de Gray. Le *code de Gray binaire réfléchi*, G^n, est défini par récurrence sur n. Tout d'abord, $G^0 = (\varepsilon)$ où ε est le mot vide. Pour tout $n \in \mathbb{N}$, ayant défini G^n, on pose

$$G^{n+1} = (G^n_0 0, \ldots, G^n_{2^n-1} 0, G^n_{2^n-1} 1, \ldots, G^n_0 1)$$

Plus précisément,

- Pour tout $i \in [\![0, 2^n - 1]\!]$, $G^{n+1}_i = G^n_i 0$.
- Pour tout $i \in [\![2^n, 2^{n+1} - 1]\!]$, $G^{n+1}_i = G^n_{2^{n+1}-i-1} 1$.

Exemples. $G^0 = (\varepsilon)$ où ε est le mot vide. $G^1 = (0, 1)$, $G^2 = (00, 10, 11, 01)$ et

$$G^3 = (000, 100, 110, 010, 011, 111, 101, 001)$$

Pour revenir à l'exemple du cube Q_3, on effectue un parcours hamiltonien en suivant les arêtes bleues. Partant du sommet 000, on parcourt la face avant dans le sens inverse des aiguilles d'une montre. Arrivés en 010, on saute sur la face arrière, que l'on parcourt dans le sens des aiguilles d'une montre. Arrivés en 001, on peut revenir à la face avant sur le sommet 000.

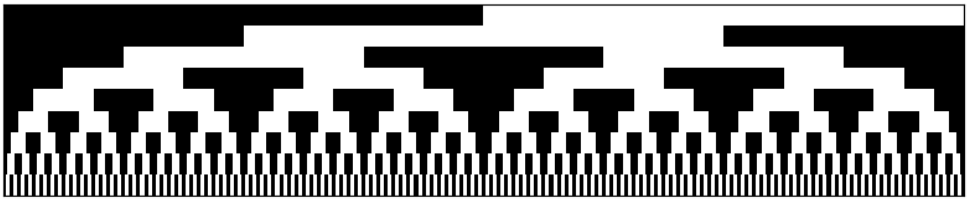

FIGURE 2 – Le code G^9. En noir, les bits égaux à 0. En blanc, les bits égaux à 1.

Proposition 1. *Pour tout $n \in \mathbb{N}$, G^n est un code de Gray cyclique.*

Démonstration. C'est une récurrence immédiate sur n. □

La fonction `code_gray0` renvoie un code de Gray réfléchi sur n bits.

```
def code_gray0(n):
    xs = [[]]
    for k in range(n):
        rxs = reversed(xs)
        xs = [x + [0] for x in xs] + [x + [1] for x in rxs]
    return xs
```

1.3 Une autre définition

Que se passe-t-il si, au lieu d'ajouter un 0 ou un 1 en fin de mot, nous les ajoutons au début ? Définissons par récurrence sur n le code G'^n en posant $G'^0 = (\varepsilon)$ puis, pour tout $n \in \mathbb{N}$,

$$G'^{n+1} = (0G'^n_0, \ldots, 0G'^n_{2^n-1}, 1G'^n_{2^n-1}, \ldots, 1G'^n_0)$$

Notation. Pour tout mot $x = b_0 \ldots b_{n-1}$, notons $\overline{x} = b_{n-1} \ldots b_0$. Remarquons que $\overline{\varepsilon} = \varepsilon$ et pour tous mots x et y, $\overline{xy} = \overline{y}\,\overline{x}$.

Proposition 2. *Pour tout $n \in \mathbb{N}$,*

$$G'_n = (\overline{G_0^n}, \dots, \overline{G_{2^n-1}^n})$$

Démonstration. Montrons ce résultat par récurrence sur n. C'est immédiat pour $n = 0$ puisque $\overline{\varepsilon} = \varepsilon$. Soit $n \in \mathbb{N}$. Supposons la propriété vraie pour n. Soit $k \in [\![0, 2^{n+1} - 1]\!]$.

- Si $k \leq 2^n - 1$, alors

$$G_k'^{n+1} = 0G_k'^n = 0\overline{G_k^n} = \overline{G_k^n 0} = \overline{G_k^{n+1}}$$

- Si $k \geq 2^n$, alors

$$G_k'^{n+1} = 1G'^n_{2^{n+1}-1-k} = 1\overline{G^n_{2^{n+1}-1-k}} = \overline{G^n_{2^{n+1}-1-k}1} = \overline{G_k^{n+1}}$$

□

2 Rang

2.1 Introduction

Soit $x \in \mathbb{B}^n$. Le *rang* de x est l'unique entier $k \in [\![0, 2^n - 1]\!]$ tel que $x = G_k^n$. Nous noterons $\mathrm{rg}_n(x)$ ce rang. Cette notation est un peu redondante, parce que nous pouvons retrouver l'entier n à partir de la longueur du mot x. Nous avons donc une bijection

$$\mathrm{rg}_n : \mathbb{B}^n \longrightarrow [\![0, 2^n - 1]\!]$$

La construction des codes G^n donne des relations de récurrence sur les fonctions rg_n. Tout d'abord, $\mathrm{rg}_0(\varepsilon) = 0$. Puis, pour tout $n \geq 1$, pour tout mot x de longueur $n - 1$,

- $\mathrm{rg}_n(x0) = \mathrm{rg}_{n-1}(x)$.
- $\mathrm{rg}_n(x1) = 2^n - 1 - \mathrm{rg}_{n-1}(x)$.

2.2 Le rang d'un mot

Proposition 3. *Soit $n \in \mathbb{N}$. Soit $k \in [\![0, 2^n - 1]\!]$. Soit $b_0 \dots b_{n-1} b_n$ l'écriture de k en base 2, où $b_n = 0$. Posons $G_k^n = g_0 \dots g_{n-1}$. Alors, pour tout $i \in [\![0, n-1]\!]$, $g_i = b_i \oplus b_{i+1}$.*

Démonstration. Raisonnons par récurrence sur n. Pour $n = 0$, la propriété est trivialement vraie. Soit $n \in \mathbb{N}$. Supposons la propriété vraie pour n. Soit $k = b_0 \dots b_n 0 \in [\![0, 2^{n+1} - 1]\!]$. Soit $G_k^{n+1} = g_0 \dots g_n = y g_n$ où y est de longueur n. Si $k \leq 2^n - 1$, alors $b_n = 0$ et $g_n = 0$. On a donc $g_n = b_n \oplus 0 = b_n \oplus b_{n+1}$. De plus, $\mathrm{rg}_{n+1}(G_k^{n+1}) = \mathrm{rg}_n(y)$. Par l'hypothèse de récurrence, on a donc pour tout $i \in [\![0, n-1]\!]$, $g_i = b_i \oplus b_{i+1}$.

Si $k \geq 2^n$, alors $b_n = 1$ et $g_n = 1$. On a donc $g_n = b_n \oplus 0 = b_n \oplus b_{n+1}$. De plus, $k = \mathrm{rg}_{n+1}(G_k^{n+1}) = 2^{n+1} - 1 - \mathrm{rg}_n(y)$ et donc

$$\mathrm{rg}_n(y) = 2^{n+1} - 1 - k$$

L'écriture en base 2 de $\mathrm{rg}_n(y)$ est donc

$$\overline{b_0}\dots\overline{b_{n-1}}0$$

De là, par l'hypothèse de récurrence, on a pour tout $i \in [\![0, n-1]\!]$,

$$g_i = \overline{b_i} \oplus \overline{b_{i+1}} = b_i \oplus b_{i+1}$$

□

Connaissant le rang d'un mot $x = g_0 \dots g_{n-1}$ de G^n, il est donc facile de déterminer x. Par exemple, quel est le mot de G^{10} de rang 732 ? L'écriture binaire de 732 est 0011101101 (rappelons que le chiffre des unités est à gauche). On a donc

$$G^{10}_{732} = 0100110111$$

Supposons donnée une fonction `vers_bits` prenant en paramètres deux entiers r et n, où $r \in [\![0, 2^n - 1]\!]$. La fonction `unrank` renvoie le mot de G^n de rang r.

```
def unrank(r, n):
    bs = vers_bits(r, n)
    bs.append(0)
    xs = n * [0]
    for k in range(n):
        xs[k] = (bs[k] + bs[k+1]) % 2
    return xs
```

Inversement, Il est tout aussi facile de déterminer le rang d'un mot de G^n. Soit $k \in [\![0, 2^n - 1]\!]$. Pour tout $i \in [\![0, n-1]\!]$, l'égalité $g_i = b_i \oplus b_{i+1}$ s'écrit aussi

$$b_i = g_i \oplus b_{i+1}$$

Sachant que $b_n = 0$, on peut ainsi déterminer $b_{n-1}, \dots, b_0$. Par exemple, le rang de $1 \dots 1$ dans G^{10} est $0101010101 = 682$.

Remarquons que l'on a la formule plus explicite

$$b_i = \sum_{j=i}^{n-1} g_j$$

La fonction `rank` renvoie le rang d'un mot $x \in \mathbb{B}^n$.

```
def rank(x):
    n = len(x)
    bs = n * [0]
    b = 0
    for k in range(n - 1, -1, -1):
        b = (x[k] + b) % 2
        bs[k] = b
    return vers_entier(bs)
```

2.3 Successeur

Soient $n \geq 1$ et $k \in [\![0, 2^n - 2]\!]$. Connaissant le k-ème mot de G^n, ainsi que la valeur de k, on peut déterminer très simplement le mot suivant.

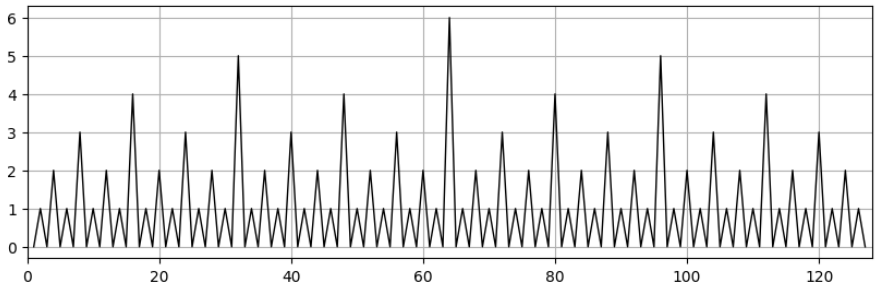

FIGURE 3 – La valuation dyadique des entiers entre 1 et 128.

Tout entier naturel non nul n s'écrit de façon unique $n = 2^\nu m$ où $\nu = \nu(n) \in \mathbb{N}$ et m est un entier impair. L'entier $\nu(n)$ est la *valuation dyadique* de n.

Proposition 4. *Soit $n \in \mathbb{N}^*$. Pour tout $k \in [\![0, 2^n - 2]\!]$, G_k^n et G_{k+1}^n diffèrent par leur bit numéro $\nu(k+1)$.*

Démonstration. Soit $i = \nu(k+1)$. On a

$$k + 1 = 2^i(2m + 1)$$

où $m \in \mathbb{N}$. L'écriture binaire de $k+1$ est donc $k + 1 = 0 \dots 01 b_{i+1} \dots b_{n-1}$ où le 1 se trouve en position i. De là, l'écriture binaire de k est $k = 1 \dots 10 b_{i+1} \dots b_{n-1}$ où le 0 se trouve en position i. Le bit numéro i de G_k^n est donc $1 \oplus b_{i+1}$ alors que le bit numéro i de G_{k+1}^n est $0 \oplus b_{i+1}$. Ces deux bits sont différents. □

Nous pouvons maintenant donner une fonction qui énumère les mots de G^n. L'avantage de cette fonction sur notre première fonction est sa complexité spatiale : si l'on excepte l'espace occupé par les mots G_k^n (qui est de l'ordre de $n2^n$), celle-ci est en $O(1)$.

```
def code_gray(n):
    x = n * [0]
    for k in range(2 ** n):
        yield x.copy()
        if k < 2 ** n - 1:
            i = nu(k + 1)
            x[i] = 1 - x[i]
```

2.4 Complexité temporelle

Quelle est la complexité temporelle de notre fonction ? Hormis les opérations en $O(1)$ effectuées dans la boucle indexée par k, cette complexité est essentiellement due au calcul de $\nu(k+1)$. Une fonction effectuant ce travail est la suivante.

```
def nu(n):
    k = 0
    while n % 2 == 0:
        k += 1
        n = n // 2
    return k
```

Le coût du calcul de $\nu(n)$ par cette fonction est de l'ordre de ... $\nu(n)$. Ainsi, la complexité temporelle de `code_gray` appelée sur un entier n est de l'ordre de

$$C_n = \sum_{i=1}^{2^n-1} \nu(i)$$

Proposition 5. *Pour tout $n \in \mathbb{N}$,*

$$C_n = 2^n - n - 1$$

Démonstration. Passons le cas évident $n = 0$ et prenons $n \geq 1$. Les valuations possibles des entiers de $[\![1, 2^n - 1]\!]$ sont 0,1, ..., $n-1$. Pour tout $j \in [\![0, n-1]\!]$, les entiers de valuation j sont de la forme $m = 2^j(2k+1)$ où $k \in \mathbb{N}$ vérifie $2^j(2k+1) < 2^n$, ou encore $2k + 1 < 2^{n-j}$, c'est à dire

$$k \leq 2^{n-1-j} - 1$$

Il y a donc 2^{n-1-j} tels entiers. De là,

$$C_n = \sum_{j=0}^{n-1} j2^{n-1-j} = 2^{n-1} \sum_{j=0}^{n-1} j2^{-j}$$

Considérons la fonction φ définie pour $x \in]0, 1[$ par

$$\varphi(x) = \sum_{j=0}^{n-1} x^j$$

On a pour tout $x \in]0, 1[$,

$$\varphi'(x) = \frac{1}{x} \sum_{j=0}^{n-1} jx^j$$

De là,

$$C_n = 2^{n-2}\varphi'\left(\frac{1}{2}\right)$$

Remarquons que pour tout $x \in]0, 1[$,

$$\varphi(x) = \frac{x^n - 1}{x - 1}$$

On en déduit

$$\varphi'(x) = \frac{(n-1)x^n - nx^{n-1} + 1}{(x-1)^2}$$

et donc

$$\begin{aligned} \varphi'\left(\frac{1}{2}\right) &= 4\left(\frac{n-1}{2^n} - \frac{n}{2^{n-1}} + 1\right) \\ &= \frac{2^n - n - 1}{2^{n-2}} \end{aligned}$$

d'où la valeur annoncée pour C_n. □