Jurnal Ilmiah KOMPUTASI, Volume 16 No : 3, Desember 2017, p-ISSN 1412-9434/e-ISSN 2549-7227

Kode Gray dan Algoritma Pembangkit untuk Fungsi Tumbuh Terrestriksi Berbatas

Ahmad Sabri

Pusat Studi Komputasi Matematika
Universitas Gunadarma, Depok
sabri@staff.gunadarma.ac.id

Abstrak

Dalam penelitian terdahulu penulis beserta rekan mendefinisikan relasi urut yang menginduksi 3-kode Gray untuk kelas fungsi tumbuh terrestriksi berbatas. Sebagai penelitian lanjutan, paper ini memberikan rancangan 1-kode Gray untuk kelas tersebut beserta algoritma pembangkit dengan kompleksitas *constant amortized time* (CAT). Berbeda dengan penelitian terdahulu yang menggunakan relasi urut, pendekatan yang dilakukan pada penelitian ini adalah dengan memanfaatkan sifat defining sequence dari kelas fungsi tumbuh terrestriksi berbatas yang memiliki sedikitnya dua anggota tetap, yaitu 0 dan 1. Rancangan kode Gray dalam penelitian ini dibangun berdasarkan rancangan kode Gray Mansour-Vajnovszki, yang dimodifikasi untuk memperoleh urutan *definingsequence* yang lebih natural dan lebih mudah diimplementasikan pada bahasa pemrograman.

Kata Kunci : kode Gray, algoritma pembangkit, fungsi tumbuh terrestriksi, constant amortized time

Pendahuluan

Kode Gray merupakan susunan untai dengan panjang sama dari sebuah kelas kombinatorial, sedemikian sehingga sebarang dua untai yang berurutan hanya berbeda dalam satu posisi. Ide ini secara formal dikemukakan oleh Gray [2]. Definisi yang lebih longgar adalah kode Gray kombinatorial, dengan notasi d-kode Gray, di mana dua untai yang berurutan memiliki perbedaan maksimum dalam d posisi.

Dalam penelitian terdahulu, penulis bersama rekan membuktikan bahwa relasi urut *Reflected Gray Code* (RGC) order menginduksi 3-kode Gray untuk kelas fungsi tumbuh terrestriksi berbatas (bounded restricted growth functions) dengan panjang ganjil, dan relasi co-RGC untuk panjang genap [1]. Studi lanjutan yang disarankan dalam penelitian tersebut adalah menemukan kode Gray yang lebih restriktif, dalam arti mendefinisikan relasi urut yang menginduksi 2-kode Gray atau bahkan 1-kode Gray untuk kelas fungsi tumbuh terrestriksi berbatas.

Paper ini memberikan sebuah rancangan aturan penyusunan barisan dan algoritma pembangkit efisien yang menginduksi 1-kode Gray untuk kelas fungsi tumbuh terrestriksi berbatas. Aturan penyusunan ini terinspirasi dari aturan Mansour-Vajnovszki [3]. Perbedaannya adalah, aturan yang didefinisikan pada paper ini tidak didasari pada sebuah relasi urut sebagaimana pada [3], namun didasari pada karakteristik *defining sequence* (didefinisikan pada bagian setelah ini) dari kelas ini yang memiliki dua simbol tetap (dibuktikan pada bagian Hasil dan Pembahasan). Rancangan ini kemudian diterapkan pada algoritma pembangkit, yang memiliki kompleksitas *constant amortized time*.

Notasi dan Definisi

Dalam paper ini istilah barisan atau untai merujuk kepada n-tupel $s_1 s_2 ... s_n$, dan kedua istilah tersebut adalah setara dan digunakan secara bergantian sesuai dengan konteks.

Himpunan fungsi tumbuh terrestriksi dengan panjang n, dinotasikan sebagai R_n, didefiniskan sebagai [3,4]:

$$R_n = \{s_1 s_2 ... s_n | s_1 = 0 \, dan \\ 0 \leq s_{k+1} \leq max\,(s_1 s_2 ... s_n) + 1, \\ untuk\, 1 \leq k < n\} \quad (1)$$

di mana max $(s_1 s_2 ... s_n)$ adalah nilai terbesar pada $s_1 s_2 ... s_n$.Berdasarkan definisi dalam [1], untuk sebuah $b \geq 0$, himpunan barisan fungsi tumbuh terrestriksi berbatas b dalam R_n, dinotasikan sebagai $R_n(b)$, didefinisikan sebagai:

$$R_n(b) = \{s_1 s_2 ... s_n | max\{s_i\}_{i=1}^{n} \leq b\} \quad (2)$$

Tabel 1 menampilkan senarai (daftar/*list*) untuk semua barisan anggota $R_5(3)$ sebanyak 51 barisan, yang disusun secara leksikografis. Susunan secara leksikografis tidak menginduksi kode Gray untuk $R_n(b)$, karena perbedaan antara dua barisan yang berurutan adalah tidak berbatas (*unbounded*). Sebagai contoh, barisan 00111 dan 01000 pada urutan 15 dan 16 berbeda dalam 4 posisi. Perbedaan ini akan semakin besar untuk n yang lebih besar.

Tabel 1: Senarai untuk $R_5(3)$, ditampilkan secara leksikografis

1.	0 0 0 0 0	27.	0 1 2 0 2
2.	0 0 0 0 1	28.	0 1 2 0 3
3.	0 0 0 1 0	29.	0 1 2 0 1
4.	0 0 0 1 2	30.	0 1 2 2 0
5.	0 0 0 1 1	31.	0 1 2 2 2
6.	0 0 1 0 0	32.	0 1 2 2 3
7.	0 0 1 0 2	33.	0 1 2 2 1
8.	0 0 1 0 1	34.	0 1 2 3 0
9.	0 0 1 2 0	35.	0 1 2 3 2
10.	0 0 1 2 2	36.	0 1 2 3 3
11.	0 0 1 2 2	37.	0 1 2 3 1
12.	0 0 1 2 1	38.	0 1 2 1 0
13.	0 0 1 1 0	39.	0 1 2 0 2
14.	0 0 1 1 2	40.	0 1 2 1 3
15.	0 0 1 1 1	41.	0 1 2 1 1
16.	0 1 0 0 0	42.	0 1 1 0 0
17.	0 1 0 0 2	43.	0 1 1 0 2
18.	0 1 0 0 1	44.	0 1 1 0 1
19.	0 1 0 2 0	45.	0 1 1 2 0
20.	0 1 0 2 2	46.	0 1 1 2 2
21.	0 1 0 2 3	47.	0 1 1 2 3
22.	0 1 0 2 1	48.	0 1 1 2 1
23.	0 1 0 1 0	49.	0 1 1 1 0
24.	0 1 0 1 2	50.	0 1 1 1 2
25.	0 1 0 1 1	51.	0 1 1 1 1
26.	0 1 2 0 0		

Definingsequence adalah barisan yang terdiri dari nilai-nilai yang dapat mengekspansi sebuah barisan dari sebuah kelas, sehingga ekspansinya juga merupakan anggota kelas tersebut [5]. Sebagai contoh, sebuah fungsi tumbuh terrestriksi 01020 (panjang 5) memiliki *defining sequence* 0123, karena ekspansi dari 01020 adalah 010200, 010201, 010202, dan 010203.

Algoritma pembangkit yang diberikan dalam penelitian ini adalah algoritma rekursif. Kompleksitasnya dihitung berdasarkan rata-rata komputasi per objek (dalam hal ini adalah untai/barisan) yang dihasilkan. Analisa kompleksitas terhadap algoritma pembangkit yang dikemukakan dalam paper ini dilakukan dengan mengikuti prinsip *constant amortized time* (prinsip CAT) berdasarkan [6], sebagai berikut. Sebuah algoritma rekursif memiliki kompleksitas CAT jika berlaku:

1. Setiap pemanggilan rekursif menghasilkan minimal sebuah untai.

2. Banyak komputasi pada sebuah pemanggilan rekursif berbanding lurus terhadap derajat pemanggilan (yaitu, banyaknya pemanggilan rekursif lanjutan yang dihasilkan oleh pemanggilan rekursif terkini).

3. Banyaknya pemanggilan rekursif berderajat satu, jika ada, adalah $O(N)$, di mana N adalah banyak untai yang dibangkitkan.

Jika sebuah algoritma pembangkit memiliki kompleksitas CAT, maka algoritma tersebut membangkitkan keseluruhan objek dengan kompleksitas rata-rata $O(1)$ per objeknya.

Hasil dan Pembahasan

Eksistensi kode Gray

Porposis 1.

Jika $s_1 s_2 ... s_k \in R_k(b)$, , maka $s_1 s_2 ... s_k\ 0 \in R_{k+1}(b)$ dan $s_1 s_2 ... s_k\ 1 \in R_{k+1}(b)$

Bukti.

Berdasarkan definisi untuk $R_n$pada (1), ekspansi sebarang $s_1 s_2 ... s_k \in R_k$, , menjadi $s_1 s_2 ... s_k s_{k+1} \in R_k$ memberikan nilai untuk s_{k+1}pada range $0 \leq s_{k+1} \leq max(s_1 s_2 ... s_k)+1$. Berdasarkan (2), pada $R_n(b)$ berlaku $max(s_1 s_2 ... s_k) \leq b$. Karena $b \geq 1$maka jelas bahwa 0 dan 1 berada di dalam range $0 \leq s_{k+1} \leq max(s_1 s_2 ... s_k)+1$.

Dalam [6] dibuktikan bahwa jika untuk setiap barisan dalam kelas yang sama dan dengan panjang yang sama memiliki *defining sequence* yang terdiri dari sedikitnya dua nilai yang tetap, maka terdapat 1-kode Gray untuk kelas tersebut. Proposisi 1 menjamin bahwa *defining sequence* untuk $R_n(b)$ sedikitnya terdiri dari dua nilai tetap, yaitu 0 dan 1, sehingga akibat berikut berlaku untuk $R_n(b)$.

Akibat 1.

Terdapat 1-kode Gray untuk $R_n(b)$

Rancangan kode Gray dan Algoritme Pembangkit

Sebagaimana telah disebutkan pada bagian sebelumnya, Proposisi 1 memastikan bahwa 0 dan 1 adalah dua nilai yang selalu terdapat pada *defining sequence* sebarang barisan pada $R_n(b)$. Berdasarkan sifat ini, untuk memperoleh 1-kode Gray, 0 dan 1 ditetapkan menjadi elemen pertama dan elemen terakhir dari *defining sequence*, sedangkan elemen yang lebih besar dari 1 berada di antara keduanya. Berikut adalah definisi yang diterapkan untuk aturan ekspansi pada kelas fungsi tumbuh terrestriksi berbatas:

Definisi 1

1. *Defining sequence* urutan naik untuk $s_1 s_2 ... s_k \in R_k(b)$,didefinisikan sebagai: $0, 2, 3, 4, ...M, 1$, dan *defining sequence* urutan turun didefinisikan sebagai: $1, M, M-1, ..., 3.2.0$, di mana $M = max\,(s_1 s_2 ... s_k)$.

2. Untuk $s_1 s_2 ... s_{k-1} p \in R_k(b)$,dan $s_1 s_2 ... s_{k-1} q \in R_k(b)$,dua barisan yang berurutan pada seranai (*list*) berlaku: jika $s_1 s_2 ... s_{k-1} p$ berekspansi secara urutan naik [turun], maka $s_1 s_2 ... s_{k-1} q$ berekspansi dengan urutan turun [naik].

Berdasarkan Definisi 1, maka dapat dikonstruksi 1-kode Gray untuk fungsi tumbuh terrestriksi berbatas, dan Proposisi 2 berikut berlaku.

Proposisi 2.

Aturan ekspansi pada Definisi 1 menginduksi 1-kode Gray untuk fungsi tumbuh terrestriksi.

Aturan ekspansi yang dikemukakan pada Definisi 1 di atas memiliki perbedaan dengan aturan ekspansi pada [3]. Perbedaan terdapat pada butir pertama, di mana *defining sequence* urutan naik pada [3] didefinisikan dalam urutan $0, 2, 4, ..., M, M-1, M-2, ..., 1$ untuk M genap, dan $0, 2, 4, ..., M, M-1, M-4, ..., 1$ untuk M ganjil.

Walaupun terdapat perbedaan urutan *defining sequence*, hal ini tidak mengubah 0 dan 1 sebagai elemen pertama dan terakhir pada urutan. Oleh karena itu, pembuktian untuk Proposisi 2 serupa dengan yang telah diberikan oleh [3]. Penerapan urutan yang natural pada paper ini bertujuan untuk memberikan kemudahan dalam menerapkan algoritma ke dalam bahasa pemrograman.

Tabel 2 menampilkan senarai untuk $R_5(3)$ berdasarkan aturan ekspansi pada Definisi 1. Senarai tersebut merupakan 1-kode Gray, karena setiap dua barisan yang berurutan memiliki perbedaan pada satu posisi saja (dalam tabel tersebut, simbol yang berbeda dengan barisan sebelumnya diberi garis bawah).

Tabel 2: Senarai 1-kode Gray untuk $R_5(3)$

1.	0 0 0 0 0	27.	0 1 2 1 $\underline{2}$
2.	0 0 0 0 $\underline{1}$	28.	0 1 2 1 $\underline{3}$
3.	0 0 0 $\underline{1}$ 1	29.	0 1 2 1 $\underline{1}$
4.	0 0 0 1 $\underline{2}$	30.	0 1 2 $\underline{3}$ 1
5.	0 0 0 1 $\underline{0}$	31.	0 1 2 3 $\underline{3}$
6.	0 0 $\underline{1}$ 1 0	32.	0 1 2 3 $\underline{2}$
7.	0 0 1 1 $\underline{2}$	33.	0 1 2 3 $\underline{0}$
8.	0 0 1 1 $\underline{1}$	34.	0 1 2 $\underline{2}$ 0
9.	0 0 1 $\underline{2}$ 1	35.	0 1 2 2 $\underline{2}$
10.	0 0 1 2 $\underline{3}$	36.	0 1 2 2 $\underline{3}$
11.	0 0 1 2 $\underline{2}$	37.	0 1 2 2 $\underline{1}$
12.	0 0 1 2 $\underline{0}$	38.	0 1 2 $\underline{0}$ 1
13.	0 0 1 $\underline{0}$ 0	39.	0 1 2 0 $\underline{3}$
14.	0 0 1 0 $\underline{2}$	40.	0 1 2 0 $\underline{2}$
15.	0 0 1 0 $\underline{1}$	41.	0 1 2 0 $\underline{0}$
16.	0 $\underline{1}$ 1 0 1	42.	0 1 $\underline{0}$ 0 0
17.	0 1 1 0 $\underline{2}$	43.	0 1 0 0 $\underline{2}$
18.	0 1 1 0 $\underline{0}$	44.	0 1 0 0 $\underline{1}$
19.	0 1 1$\underline{2}$ 0	45.	0 1 0 $\underline{2}$ 1
20.	0 1 0 2 $\underline{2}$	46.	0 1 0 2 $\underline{3}$
21.	0 1 0 2 $\underline{3}$	47.	0 1 0 2 $\underline{2}$
22.	0 1 1 2 $\underline{1}$	48.	0 1 0 2 $\underline{0}$
23.	0 1 1 $\underline{1}$ 1	49.	0 1 0 $\underline{1}$ 0
24.	0 1 1 1 $\underline{2}$	50.	0 1 0 1 $\underline{2}$
25.	0 1 1 1 $\underline{0}$	51.	0 1 0 1 $\underline{1}$
26.	0 1 $\underline{2}$ 1 0		

Penerapan kedua aturan ekspansi di atas menghasilkan algoritma Gen pada Gambar 1 dengan algoritma inisiasi pada Gambar 2.

Pada algoritma main() (Gambar 2), terjadi inisiasi variabel yaitu panjang barisan n, batas atas *bound*, array $s[n]$ (mewakili $s_1 s_2 ... s_n$), dan array $dir[n]$(mewakili ekspansi urutan naik pada posisi k jika $dir[k] = 0$, dan urutan turun jika $dir[k] = 1$, pada inisiasi *dir* bernilai -1 karena pada setiap pemanggilan rekursif nilainya akan bertambah 1, ini sebuah trik agar pemanggilan rekursif pertama nilai *dir* adalah 0. Pemanggilan rekursif pertama terjadi pada baris M10 untuk $k = 2$ (karena untuk k = 1 nilai yang mungkin untuk hanyalah 0).

```
G01   procedure Gen(k, M: integer)
G02   global n, s[n], dir[n]: integer
G03   local i,u: integer;
G04   if (M = bound) then M := bound - 1;
G05   if (k = n + 1) then display();
G06   else if (remainder of dir[k]/2 = 0) then
G07           for i in (0, 2, 3, ..., M, M + 1, 1)
G08               s[k] :=i;
G09               if (M < s[k]) then u := s[k]; else u := M;
G10               dir[k + 1]++;
G11               Gen(k + 1,u);
G12           else
G13               for i in (1, M + 1, M, ..., 3, 2, 0)
G14                   s[k] := i;
G15                   if (M < s[k]) then u := s[k]; else u := M;
G16                   dir[k + 1]++;
G17                   Gen(k + 1,u);
```

Gambar 1: Algoritme Gen

Pada algoritma Gen, baris G06 memeriksa apakah ekspansi dilakukan secara urutan naik (jika remainder of $dir[k]/2 = 0$, maka dilanjutkan ke baris G07-G11) atau turun (langsung menuju baris G13-G17). Berdasarkan for loop pada baris G07 dan G13, ekspansi dilakukan untuk sedikitnya dua nilai, yaitu 0 dan 1 (dengan kata lain, setiap pemanggilan rekursif berderajat minimal dua).

```
M01   main()
M02       global n, s[20], dir[20], bound: integer
M03       local i: integer;
M04       input n;
M05       input bound;
M06           for i := 1 to n
M07           s[i] := 0;
M08           dir[i] := -1;
M09       dir[1] := dir[2] := 0;
M10       Gen(2,0);
```

Gambar 2: Algoritme Inisiasi

Oleh karena tidak adanya pemanggilan rekursif berderajat satu, maka evaluasi berdasarkan sifat ketiga dari prinsip CAT tidak diterapkan untuk algoritma ini. Dapat ditelusuri pada algoritma Gen bahwa setiap pemanggilan rekursif akan mengekspansi untai dengan panjang k menjadi setidaknya dua untai dengan panjang k + 1. Keadaan ini memenuhi prinsip CAT yang pertama. Sifat kedua dari prinsip CAT dipenuhi oleh Gen karena setiap pemanggilan rekursif mengeksekusi prosedur komputasi yang sama (walaupun dengan parameter yang berbeda), dan pemanggilan rekursif ini terus berlangsung hingga seluruh fungsi tumbuh terrestriksi berbatas dengan panjang n dihasilkan secara ekshaustif. Oleh karena itu, algoritma Gen memiliki kompleksitas CAT. Hal ini berarti keseluruhan untai dibangkitkan dengan kompleksitas rata-rata O(1) per untai.

4. Penutup

Penelitian ini memberikan rancangan 1-kode Gray dan algoritma pembangkit efisien untuk kelas fungsi tumbuh terrestriksi berbatas. Hasil ini merupakan kode Gray paling optimal untuk kelas tersebut. Penelitian lanjutan disarankan untuk merumuskan rancangan kode Gray dengan pendekatan lain, yaitu kode Gray yang diinduksi oleh relasi urut, sehingga konsistensi urutan elemen dapat dipertahankan jika rancangan ini diterapkan untuk kelas yang lebih besar.

Daftar Pustaka

[1] A. Sabri, V. Vajnovszki, “More restricted growth functions: Gray codes and exhaustive generations”, Graphs and Combinatorics, 33(3), hal. 573-582, 2017.

[2] F. Gray, “Pulse code communication”, U.S. Patent No. 2632058, 1953.

[3] T. Mansour, V. Vajnovszki, “Efficient generation of restricted growth words”, Information Processing Letters , 113(7), hal. 613-616, 2013.

[4] A. Sabri, V. Vajnovszki, “Reflected Gray code based orders on some restricted growth sequences”, The Computer Journal , 58(5), hal. 1099-1111, 2015.

[5] T. Walsh, “Generating Gray codes in O(1) worst-case time per word”, dalam “4th Discrete Mathematics and Theoretical Com-

puter Science Conference", Dijon-France, 7-12 July 2003 (juga pada LNCS 2731, hal. 71-88).

[6] F. Ruskey, "Combinatorial generation", Monograf online pada www.1stworks.com/ref/ruskeycombgen.pdf, University of Victoria, Canada, 2003. Diakses 30 November 2017.