AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **39** (2007), Pages 129–134

Equivalence classes of 5-bit gray codes

Clay Culver   David Leach

Department of Mathematics
University of West Georgia
Carrollton, GA 30118
U.S.A.

Abstract

An n-bit gray code can be represented as a hamilton cycle in the n-cube together with a direction. Two n-bit gray codes are equivalent if there is an automorphism of the n-cube that maps the edge set of one onto the other. In this paper we detail a method for counting the number of equivalence classes of 5-bit gray codes, and classify the equivalence classes according to the distribution of the bit-positions in their transition sequences.

1 Introduction

The n-cube Q_n is defined as the graph having the binary words of length n as its vertices, with two vertices being adjacent if and only if they differ in exactly one bit position. It is clear that $|V(Q_n)| = 2^n$. An n-bit gray code is a circular listing of all 2^n binary words of length n in such a way that two consecutive words differ in exactly one bit position. Thus an n-bit gray code is equivalent to a directed hamilton cycle in the n-cube for $n \geq 2$.

Since each pair of consecutive words in a gray code differ in exactly one position, a gray code can be completely described by listing the bit positions in which each pair of consecutive words differ. Such a listing is called a *transition sequence*. Let $C = w_1, w_2, \ldots, w_{2^n}$ be a gray code. Then the transition sequence of C is $T := t_1, t_2, \ldots, t_{2^n}$, where $t_i \in \{1, \ldots, n\}$ is the position where w_i and w_{i+1} differ.

Gray codes were named for Frank Gray, who patented the binary reflected gray code in 1953 for use in error-control in communication systems using pulse code modulation [5]. They have found applications in many areas including computer engineering, databases, puzzle solving, data compression, and a host of other combinatorial problems [1, 2, 4, 7]. For a more complete background on gray codes and their applications, see [8].

Sloane's sequence A003042 [10] gives the total number of gray codes of length n for $1 \leq n \leq 5$ is 1, 2, 12, 2688, and 1,813,091,520. The number is unknown for

$n \geq 6$, although Silverman et al. estimate the number to be approximately $7 \cdot 10^{22}$ for $n = 6$ [9].

Two gray codes are said to be *equivalent* if there exists an automorphism of Q_n that maps the edge set of one onto the other. We denote the automorphism group of Q_n by $\Gamma(Q_n)$.

Lemma 1.1 *The group $\Gamma(Q_n)$ has order $n! \cdot 2^n$.*

Remark 1.1 *Using a different definition of equivalence, Sloane's sequence A091302 lists the number of equivalence classes of n-bit gray codes. The definition of equivalence used by Sloane and its relationship to that used here is discussed in Section 2.*

By definition, an automorphism of Q_n is a permutation of the vertices that preserves the incidence relation. The action of an automorphism of Q_n on a transition sequence can be described by a permutation of the set $\{1, \ldots, n\}$ of bit positions, a cyclic rotation of the sequence, and whether or not to reverse the sequence. Note that this gives $2n! \cdot 2^n$ possiblilties - exactly twice $|\Gamma(Q_n)|$. This is what is expected, since each hamilton cycle on Q_n gives two gray codes – one in each direction. If two n-bit gray codes, say A and B are equivalent, then either there exists an automorphism that transforms the transition sequence of A into B, or there exists an automorphism that transforms the transition sequence of A into the reverse of the transition sequence of B.

It is easy to see that there is only one equivalence class for for $n = 1$ and $n = 2$. For $n = 1$, $Q_1 \cong K_2$, and there is only gray code; for $n = 2$, $Q_2 \cong C_4$, and the two gray codes arise from traversing the 4-cycle in both directions. For $n = 3$, each gray code is equivalent to the binary reflected gray code. The case $n = 4$ was settled in 1958 by Gilbert [3], who also determined the number of equivalence classes of shorter cycles and paths on the 4-cube. Gilbert showed that the 2688 4-bit gray codes fall into 9 equivalence classes. His results are summarized in Table 1. In [6] Knuth notes that for $n = 5$, the number of equivalence classes is 237,675; however, no details are given as to how this number was determined. In this paper, we describe a method for efficiently counting equivalence classes that also sorts them according to the number of times each bit-position occurs in their transition sequences. We then use this method to corroborate Knuth's result and classify the equivalence classes of 5-bit gray codes.

2 Generating all n-bit Gray Codes

Before we can sort the n-bit gray codes into equivalence classes, we generate all of them. The method used to generate all the n-bit gray codes uses an n-ary tree in which each node contains an n-bit binary word, with the zero word at the root. Each node has n children, the i^{th} child node containing the word that differs from the word in the parent node in the i^{th} bit position. The arc from a parent to a child is labeled i. (Note that each of the binary words will appear in multiple nodes.) Only the first

Equivalence Class	Transition Sequence	Equivalence Class Size
1	1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 4	96
2	1, 2, 1, 3, 1, 2, 1, 4, 1, 3, 2, 3, 1, 3, 2, 4	768
3	1, 2, 1, 3, 1, 2, 1, 4, 2, 1, 2, 3, 2, 1, 2, 4	192
4	1, 2, 1, 3, 1, 2, 4, 2, 1, 3, 1, 2, 1, 3, 4, 3	192
5	1, 2, 1, 3, 1, 2, 4, 3, 1, 2, 1, 3, 1, 2, 4, 3	192
6	1, 2, 1, 3, 1, 4, 1, 2, 4, 1, 4, 3, 2, 1, 2, 4	384
7	1, 2, 1, 3, 1, 4, 3, 2, 3, 4, 1, 4, 2, 3, 2, 4	384
8	1, 2, 1, 3, 2, 1, 2, 4, 1, 2, 1, 3, 2, 1, 2, 4	96
9	1, 2, 1, 3, 2, 1, 2, 4, 1, 3, 1, 2, 3, 1, 3, 4	384

Table 1: 4-bit gray codes equivalence classes

$2^n + 1$ levels of the tree are needed, so the nodes on level $2^n + 1$ are the leaves of the tree. A directed path gives a gray code if it

1) begins at the root,

2) ends at a leaf containing the zero word, and

3) contains no repeated words, other than the zero word at the root and leaf.

To find all such directed paths, we perform a depth-first search on the tree. At each node we traverse the arcs leaving the node in order of their labels. Thus the gray codes are found in a lexical order. If a path is found that meets the three conditions above, the words contained by the nodes of the path form a gray code, and the edge labels give its transition sequence. Otherwise, during the search, if a node N contains a word that is contained by a previous node in the path, then the path does not give a gray code, and furthermore cannot be a subpath of a path that gives a gray code. The branch of the tree beginning at (and containing) N can be pruned and the search continued from the parent of N.

Because this process runs in exponential time, we need to exploit the symmetry of the tree to reduce the runtime. If we add the restriction that the first occurrance of the bit positions $\{1, \ldots, n\}$ in the transition sequence occur in numerical order, then this process generates an incomplete set $\mathcal{T}_n$ of transition sequences; however if we apply each permutation of the symmetric group S_n to each $T \in \mathcal{T}_n$, we get the entire set of n-bit gray codes. We call the gray codes contained in $\mathcal{T}_n$ the n-bit *canonical* gray codes. (The values of $|\mathcal{T}_n|$ are given by Sloane's sequence A091302 [10].) In the $n = 5$ case, $|\mathcal{T}_5|$ =15,109,096, which implies that there are 1,813,091,520 5-bit gray codes. Note that when sorting into equivalence classes, we need only to sort the codes listed in $\mathcal{T}_n$.

3 Sorting into Equivalence Classes.

Given all n-bit gray codes, a representative from each equivalence class is found by Algorithm 3.1.

Algorithm 3.1
Initialization:

Let C be the sequence of all n-bit gray codes, and let $i = 1$.

Iteration:

1. For $k \in \{i+1, \ldots, |C|\}$, if C_k is equivalent to C_i, then flag C_k.
2. Remove the flagged codes from C.
3. Increment i.
4. If $i \geq |C|$ then output C and STOP. Else goto step 1.

Algorithm 3.1 leaves C as a list of n-bit gray codes, no two of which are pairwise equivalent.

Although Algorithm 3.1 runs in $O(|C|^2)$ time, it is still somewhat slow because the comparison operation used in Step 1 computationally expensive. This can be mitigated if we can partition the set C into multiple sets $C_1, \ldots, C_k$ such that no two equivalent gray codes are placed into different parts of the partition, and then running Algorithm 3.1 on each of the parts of C_i for $1 \leq i \leq k$. We can find such a partition by looking at the number of times the bit positions are used in each transition sequence. Given a transition sequence T, define its *dimension vector* $D(T) = [a_1, \ldots, a_n]$ where a_i is the number of occurrances of i in T. Note that $\sum_{j=1}^{n} a_j = 2^n$, and that if T is canonical, then the entries of $D(T)$ form a non-increasing sequence.

Lemma 3.1 *Let T_1 and T_2 be the transition sequences of two canonical gray codes. If $D(T_1) \neq D(T_2)$, then T_1 and T_2 are not equivalent.*

Proof: Cyclic shifting and reversing the gray code do not affect the number of occurrences of each bit position in the transition sequence. Applying a permutation to the bit positions gives a non-canonical code, and thus doesn't need to be considered. □

There are 30 possible dimension vectors for the canonical 5-bit gray codes which are listed in Table 2. Note that the reflected gray codes have dimension vector #30 and the balanced gray codes have dimension vector #1. Let D_i be the set of canonical gray codes with the dimension vector listed in Table 2.

Running Algorithm 3.1 on the D_1 through D_{30} produces a listing of one transition sequence for each equivalence class of 5-bit gray codes. The numbers of equivalence classes of gray codes for each dimension vector are given in Table 3.

i	Dimension Vector	$\|D_i\|$	# Gray Codes $5! \cdot \|D_i\|$	i	Dimension Vector	$\|D_i\|$	# Gray Codes $5! \cdot \|D_i\|$
1	8,6,6,6,6	1,658,264	198,991,680	16	12,6,6,6,2	90,176	10,821,120
2	8,8,6,6,4	4,954,208	594,504,960	17	12,8,4,4,4	114,624	13,754,880
3	8,8,8,4,4	808,384	97,006,080	18	12,8,6,4,2	248,704	29,844,480
4	8,8,8,6,2	607,424	72,890,880	19	12,8,8,2,2	8,416	1,009,920
5	10,10,4,4,4	110,144	13,217,280	20	14,10,4,2,2	224	26880
6	10,10,6,4,2	239,040	28,684,800	21	14,12,2,2,2	0	0
7	10,10,8,2,2	16,120	1,934,400	22	14,6,4,4,4	13,952	1,674,240
8	10,6,6,6,4	1,776,960	213,235,200	23	14,6,6,4,2	14,784	1,774,080
9	10,8,6,4,4	2,621,152	314,538,240	24	14,8,4,4,2	6,336	760,320
10	10,8,6,6,2	963,712	115,645,440	25	14,8,6,2,2	1,888	226,560
11	10,8,8,4,2	460,160	55,219,200	26	16,10,2,2,2	0	0
12	12,10,4,4,2	27,808	3,336,960	27	16,4,4,4,4	32	3,840
13	12,10,6,2,2	8,048	965,760	28	16,6,4,4,2	160	19,200
14	12,12,4,2,2	280	33,600	29	16,6,6,2,2	24	2,880
15	12,6,6,4,4	358,064	42,967,680	30	16,8,4,2,2	8	960
					Total	15,109,096	1,813,091,520

Table 2: Dimension vectors for $n = 5$

i	Dimension Vector	Equivalence Classes	i	Dimension Vector	$\|D_i\|$
1	8,6,6,6,6	26,155	16	12,6,6,6,2	1,457
2	8,8,6,6,4	77,706	17	12,8,4,4,4	1,802
3	8,8,8,4,4	12,764	18	12,8,6,4,2	3,906
4	8,8,8,6,2	9,625	19	12,8,8,2,2	154
5	10,10,4,4,4	1,773	20	14,10,4,2,2	6
6	10,10,6,4,2	3,802	21	14,12,2,2,2	0
7	10,10,8,2,2	286	22	14,6,4,4,4	227
8	10,6,6,6,4	27,891	23	14,6,6,4,2	248
9	10,8,6,4,4	41,029	24	14,8,4,4,2	106
10	10,8,6,6,2	15,163	25	14,8,6,2,2	34
11	10,8,8,4,2	7,271	26	16,10,2,2,2	0
12	12,10,4,4,2	451	27	16,4,4,4,4	1
13	12,10,6,2,2	133	28	16,6,4,4,2	5
14	12,12,4,2,2	12	29	16,6,6,2,2	2
15	12,6,6,4,4	5,665	30	16,8,4,2,2	1
				Total	237,675

Table 3: Equivalence classes for $n = 5$, sorted by dimension vector.

References

[1] C.C. Chang, H.Y. Chen and C.Y. Chen, Symbolic Gray Codes as a Data Allocation Scheme for Two Disc Systems, *Comput. J.* 35 (1992), 299–305.

[2] M. Chen and K.G. Shen, Subcube Allocation and Task Migration in Hypercube Machines, *IEEE Trans. Comput.* 39 (1990), 1146–1155.

[3] E.N. Gilbert, Gray Codes and Paths on the n-cube, *Bell Systems Tech. J.* 37 (1958), 815–826.

[4] M. Gardner, Curious Properties of the Gray Code and How it Can Be Used to Solve Puzzles, *Sci. Amer.* 227 (1972), 106–109.

[5] F. Gray, Pulse Code Communications, U.S. Patent 2632058, March 1953.

[6] D. Knuth, The Art of Computer Programming, Vol 4, Fascicle 2: Generating all Tuples and Permutations, Addison-Wesley Professional, 2005.

[7] D. Richards, Data Compression and Gray-code Sorting, *Inform. Process. Lett.* 22 (1986), 201–205.

[8] C. Savage, A Survey of Combinatorial Gray Codes, *SIAM Rev.* 39, No. 4 (1997), 605–629.

[9] J. Silverman, V.E. Vickers and J.L. Sampson, Statistical Estimates of th n-bit Gray Codes by restricted random generation of permutations of 1 to 2^n, *IEEE Trans. Infor. Theory* 29 (1983), 894–901.

[10] N.J.A. Sloane, On-Line Encyclopedia of Integer Sequences, `http://www.research.att.com/~njas/sequences`.

(Received 22 Aug 2006)