PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE
Nouvelle série, tome 78(92) (2005), 87–92

NOTE ON GRAY CODES FOR PERMUTATION LISTS

Seymour Lipschutz, Jie Gao, and Dianjun Wang

Communicated by Slobodan Simić

ABSTRACT. Robert Sedgewick [**5**] lists various Gray codes for the permutations in S_n including the classical algorithm by Johnson and Trotter. Here we give an algorithm which constructs many families of Gray codes for S_n, which closely follows the construction of the Binary Reflexive Gray Code for the n-cube Q_n.

1. Introduction

There are $(n!)!$ ways to list the $n!$ permutations in S_n. Some such permutation lists have been generated by a computer, and some such permutation lists are Gray codes (where successive permutations differ by a transposition). One such famous Gray code for S_n is by Johnson [**4**] and Trotter [**6**]. In fact, each new permutation in the Johnson–Trotter (JT) list for S_n is obtained from the preceding permutation by an adjacent transposition.

Sedgewick [**5**] gave a survey of various Gray codes for S_n in 1977. Subsequently, Conway, Sloane and Wilks [**1**] gave a new Gray code CSW for S_n in 1989 while proving the existence of Gray codes for the reflection groups. Recently, Gao and Wang [**2**] gave simple algorithms, each with an efficient implementation, for two new permutation lists GW_1 and GW_2 for S_n, where the second such list is a Gray code for S_n. The four lists, JT, CSW, GW_1 and GW_2, for $n = 4$, are pictured in Figure 1.

This paper gives an algorithm for constructing many families of Gray codes for S_n which uses an idea from the construction of the Binary Reflected Gray Code ($BRGC$) of the n-cube Q_n.

Our paper is organized as follows. Section 2 describes the Binary Reflected Gray Code for Q_n, and also discusses the generation of the JT list and the CSW list for S_n. Section 3 gives our algorithm which constructs our new Gray code $L(n)$ for S_n, and also constructs many families of Gray codes for S_n.

2000 *Mathematics Subject Classification*: 94B60, 68R05, 20F65, 05C38.
Key words and phrases: Algorithms, Combinatorial problems, Permutations, Gray codes.
The third author partially supported by NSFC(60172005).

List $JT(4)$	List $CSW(4)$	List $GW_1(4)$	List $GW_2(4)$
1234 [**123**]	**123** 4		
1243		4**321**	1234
1423	124 3	4**312**	1243
4123	142 3	4**231**	1423
	412 3	4**132**	1324
4132 [**132**]	421 3	4**213**	1342
1432	241 3	4**123**	1432
1342	214 3		
1324		3421	4132
	123 4	3412	4123
3124 [**312**]	**231** 4	2431	2143
3142		1432	3142
3412	234 1	2413	3124
4312	243 1	1423	2134
	423 1		
4321 [**321**]	432 1	3241	2314
3421	342 1	3142	2413
3241	324 1	2341	4213
3214		1342	3214
	321 4	2143	3412
2314 [**231**]	**312** 4	1243	4312
2341			
2431	314 2	3214	4321
4231	341 2	3124	4231
	431 2	2314	2431
4213 [**213**]	413 2	1324	3421
2413	143 2	2134	3241
2143	134 2	1234	2341
2134			
	132 4		

FIGURE 1. JT, CSW, GW_1 and GW_2 Lists for $n = 4$

2. Gray Codes and the JT and CSW lists

The original idea of a Gray code was to list the codewords (n-bit strings) in the n-cube Q_n so that successive codewords differ in only one bit. Thus a Gray code for Q_n is a Hamiltonian circuit through the $2n$ vertices of Q_n. [The fact that this can be done for any Q_n follows from the construction of the Binary Reflected Gray Code ($BRGC$) for Q_n (see [**3**]) which we describe below.]

The above idea of a Gray code has been generalized as follows. A Gray code for any combinatorial family of objects is a listing of the objects such that only a "small change" takes place from one object to the next in the list. The definition of "small change" depends on the particular family, its context, and its applications.

Q_3	Q_3 reversed
0 1 1 0 0 1 1 0	0 0 1 1 1 1 0 0
0 0 0 0 1 1 1 1	0 1 1 0 0 1 1 0
0 0 1 1 1 1 0 0	1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1

FIGURE 2. Binary Reflected Gray Code for Q_4

For example, for a finite group G with a set of generators, a Gray code for G is usually a Hamiltonian circuit in its Cayley graph or, equivalently, a listing of the elements of G so that successive elements differ by the product of a generator.

This paper is mainly interested in the following type of Gray code for S_n. [Other types of Gray codes for S_n also exist.]

DEFINITION 1. A Gray code for S_n is a list L of the elements in S_n such that successive permutations differ by a transposition.

We begin by describing a recursive construction of the Gray code for the n-cube Q_n. Specifically, Figure 2 shows how the $BRGC$ for Q_4 can be obtained from the $BRGC$ for Q_3. That is, first we list the Gray code for Q_3 which appears as the upper left 3×8 matrix (where the codewords are the columns). Then next to it we list the Gray code for Q_3 but in reverse order. This yields a 3×8 matrix. Finally, we add a fourth row consisting of eight 0's followed by eight 1's. This gives the $BRGC$ for Q_4.

We show that this is in fact a Gray code for Q_4. Observe that only one bit changes within the first eight columns since the fourth bit is always 0, and only one bit changes within the last eight columns since the fourth bit is always 1. Furthermore, the two middle columns (and also the first and last columns) are identical except for the added last bits, and so again only one bit changes. The $BRGC$ for Q_n is the one obtained recursively in this way beginning with the matrix $[0, 1]$ for $Q = Q_1$. [Note this construction gives a Hamiltonian circuit, not just a Hamiltonian path, for the n-cube Q_n.]

Discussing the $BRGC$, Herbert Wilf [**7**] comments: "This method of copying a list and its reversal seems to be a good thing to think of when trying to construct some new kind of Gray code." We do this below in our main algorithm.

The Johnson–Trotter permutation list is also defined recursively. The list for $n = 4$ is shown in Figure 1. Note that the list is partitioned into six "blocks", each with four successive permutations. Each block corresponds and is labelled by a permutation in S_3 which is in boldface and is to the right of the block. We note that the boldface labels form the JT list for S_3.

Now in the first block, the largest item 4 sweeps from right to left, in the second block from left to right, in the third block for right to left, and so on. Also, the relative positions of the remaining items 1, 2, 3 do not change in each block and correspond to the label of the block. Moreover, the recursive changes of the relative positions of 1, 2, 3 occur only from block to block when the largest item 4 is in an end position. Thus the item 4 does not interfere with any transposition involving the items 1, 2, 3. Thus the Johnson–Trotter list $JT(n)$ is a Gray code for S_n. [In fact, successive permutations in $JT(n)$ differ by only an adjacent transposition.]

The Conway–Sloane–Wilks (CSW) list is also defined recursively. The second column in Figure 1 indicates how $CSW(4)$ is obtained from $CSW(3)$. First the number $n = 4$ is appended to each permutation in $CSW(3)$ which appears in boldface in Column 2. Then a sequence of $3! = 6$ permutations is inserted between each pair of permutations in $CSW(3)$ which end in the same number, that is, between 123 and 213, between 231 and 321, and between 312 and 132. We refer the reader to their paper for a full description of their algorithm.

The last two columns in Figure 1 contain, respectively, the Gao–Wang permutation lists GW_1 and GW_2 for $n = 4$. Observe that each list is partitioned into four "blocks", where each block contains 6 successive permutations (which actually correspond to the 6 permutations in S_3). We describe the construction of the list $GW_1(n)$ here by showing how $GW_1(4)$ is obtained from $GW_1(3)$. The first block in $GW_1(4)$ contains, in boldface, the permutation list for $GW_1(3)$ with each such permutation preceded by the number 4. The second block contains the same permutation list $GW_1(3)$ but now the 4 is in the second position. And so on. The construction of the permutation list GW_2 lies beyond the scope of this paper.

3. Generating Algorithm and a New Gray Code $L(n)$ for S_n

Here we give an algorithm which is used to generate a new Gray code $L(n)$ for S_n and which can also be used to generate many other families of gray codes for S_n. First, however, we need some new notation and terminology. Let L be a list. We let L^R denote the list L written backwards, and we let $r * L$ denote the list consisting of r alternating copies of L and L^R; that is:

$$r * L = [L; L^R; L; L^R; ...; L \text{ or } L^R] \quad (r \text{ components}).$$

Each copy of L or L^R will be called a block in $r * L$. Note that $r * L$ ends in L or L^R according as r is odd or even.

Now let x be a letter. Note first that there are $r + 1$ places to add x to an r-letter word w. For example, if $w = abcd$, then there are 5 places to add x to the word w as follows:

$$xabcd, axbcd, abxcd, abcxd, abcdx.$$

Next we define a new type of list. Let $L = [B_1; B_2; \ldots; B_{r+1}]$ be a list of r-item words which is partitioned into $r + 1$ nonempty sublists B_i (called blocks), and let x be a letter. Then we define

$$I[x, L]$$

to be the list obtained by "**interweaving x block by block in** L", that is, where x is added before the first item in every word in B_1, before the second item in every word in B_2, and so on, until x is added after the last item in every word in B_{r+1}.

For example, suppose $L = [ab, a'b'; cd; ef, e'f']$, a list of five $r = 2$ letter words partitioned into $r + 1 = 3$ blocks (separated by semicolons). Then

$$I[x, L] = [xab, xa'b'; cxd; efx, e'f'x].$$

The following algorithm applies.

Algorithm 3.1. The input is a permutation list L for S_{n-1}, where $n > 2$, and the output is a permutation list L' for S_n.

Step 1. Form the list $n * L$ which consists of n alternating copies of L and L^R.

Step 2. Interweave the number n block by block in $n*L$ yielding $L' = I[n, n*L]$.

The fact that L' is a list of permutations of S_n follows from the fact that L' contains $n!$ permutations and no two of them are the same.

We are now ready to define our permutation list $L(n)$. Firstly, we let $L(1) = [1]$ and $L(2) = [21, 12]$). Then, for $n > 2$, we let $L(n)$ be obtained from $L(n-1)$ using the above Algorithm 3.1. We illustrate how $L(3)$ and $L(4)$ are obtained below.

Example 1. We apply Algorithm 3.1 to $L(2) = [21, 12]$ to obtain $L(3)$:

Step 1. Form the list $3 * L(2) = [L(2); L^R(2); L(2)]$, that is,

$$3 * L(2) = [21, 12; 12, 21; 21, 12]$$

Step 2. Interweave 3 block by block in $3 * L(2)$ yielding

$$L(3) = [321, 312, 132, 231, 213, 123]$$

Example 2. We apply Algorithm 3.1 to $L(3)$ to obtain $L(4)$:

Step 1. Form the list $4 * L(3) = [L(3); L^R(3); L(3); L^R(3)]$, that is,

$$\begin{aligned} 4 * L(3) = [&321, 312, 132, 231, 213, 123; 123, 213, 132, 231, 312, 321; \\ &321, 312, 132, 231, 213, 123; 123, 213, 132, 231, 312, 321] \end{aligned}$$

Step 2. Interweave 4 block by block in $4 * L(3)$ yielding $L(4) = I[4, 4 * L(3)]$. Namely:

$$\begin{aligned} L(4) = [&4321, 4312, 4132, 4231, 4213, 4123; 1423, 2413, 1432, 2431, 3412, 3421; \\ &3241, 3142, 1342, 2341, 2143, 1243; 1234, 2134, 1324, 2314, 3124, 3214] \end{aligned}$$

Observe that that $L(4)$ has $4(6) = 24$ permutations, as expected.

Remark. Suppose in Algorithm 3.1 we change Step 1 so that we form the list L^n, n copies of L, not alternating L and L^R. Then, beginning with $L(1) = 1$ and successively applying the modified Algorithm 3.1, we would obtain the Gao-Wang List GW_1 which is not a Gray code for S_n.

The importance of our Algorithm 3.1 comes from the following theorem.

Main Theorem. *Suppose L is a Gray code for S_{n-1}, and suppose L' is obtained from L using Algorithm* 3.1. *Then L' is a Gray code for S_n.*

Proof. Since L is a Gray code for S_{n-1}, we have that L^R is also a Gray code for S_{n-1}. Thus successive permutations within each block of $L' = I[n, n * L]$ differ only by a transposition since n is always in the same position and the remaining members of the list are in the same relative position as in L or L^R.

Now consider successive permutations from one block to the next in L'. Note that the last permutation in L is the same as the first permutation in L^R, and the last permutation in L^R is the same as the first permutation in L. Thus the only change from the last permutation in a block in L' to the first permutation in the next block is that n interchanges its position with the next integer. Thus again successive permutations in L' differ only by a transposition.

Accordingly, L' is a Gray code for S_n, and the theorem is proved. □

We now give applications of our Main Theorem.

EXAMPLE 3. (a) $L(2) = [21, 12]$ is a Gray code for S_2. Therefore our lists L(n) are Gray codes for S_n.

(b) Consider an integer $r > 1$, the Johnson–Trotter Gray code $JT(r)$ for S_r, and the Gao–Wang list $GW_2(r)$ which is also a Gray code for S_r. Accordingly, applying Algorithm 3.1 successively to $JT(r)$ or to $GW(r)$ yields Gray codes for S_n for $n > r$.

The above example, which gives two families of Gray codes for almost all S_n, may indicate that the number of Gray codes for S_n which can be easily generated by a computer may be more numerous than some may have previously assumed.

Acknowlegement. The work reported here was done while the first author Prof. Seymour Lipschutz was visiting Tsinghua University. He is thankful to Tsinghua University for its hospitality.

References

[1] J. H. Conway, N. J. A. Sloane and A.R. Wilks, *Gray codes for reflection groups*, Graphs and Combin. **5** (1989), 315–325.
[2] J. Gao and D. Wang, *Permutation generation: two new algorithms*, (2004), preprint.
[3] E. N. Gilbert, *Gray codes and paths on the n-cube*, Bell Syst. Tech. J. **37** (1958), 815–826.
[4] S. M. Johnson, *Generation of permutations by adjacent transpositions*, Math. Comput. **17** (1963), 282–285.
[5] R. Sedgewick, *Permutation generation methods*, Comp. Surveys **9** (1977), 137–164.
[6] H. F. Trotter, *Algorithm 115, Permutations*, Comm. ACM **5** (1962), 434–435.
[7] H. Wilf, *Combinatorial algorithms—an update*, SIAM, Philadelphia, 1989

Department of Mathematics
Temple University
Philadelphia
PA 1912, USA
seymour@temple.edu

(Received 29 03 2005)
(Revised 05 12 2005)

School of Software
Tsinghua University
Beijing 100084
China

Department of Mathematical Sciences
Tsinghua University
Beijing 100084
China
djwang@math.tsinghua.edu.cn