Month 200X, Vol.21, No.X, pp.XX–XX *J. Comput. Sci. & Technol.*

Generating Combinations by Three Basic Operations

Yongxi Cheng[1]

[1] *Department of Computer Science, Tsinghua University, Beijing 100084, China*

E-mail: cyx@mails.tsinghua.edu.cn

Received January 1, 2007.

Abstract We investigate the problem of listing combinations using a special class of operations, *prefix shifts*. Combinations are represented as bitstrings of 0's and 1's, and prefix shifts are the operations of rotating some prefix of a bitstring by one position to left or right. We give a negative answer to an open problem asked by F. Ruskey and A. Williams (Generating Combinations by Prefix Shifts, Proc. 11th Annual International Computing and Combinatorics Conference 2005, LNCS 3595, Springer, (2005), 570-576), that is whether we can generate combinations by only using three very basic prefix shifts on bitstrings, which are transposition of the first two bits and the rotation of the entire bitstring by one position in either direction (i.e., applying the permutations σ_2, σ_n and $\sigma_n{}^{-1}$ to the indices of the bitstrings).

Keywords generating combinations, gray codes, prefix shifts

1 Introduction

An important class of problems in combinatorial algorithms is efficient listing of fundamental combinatorial objects such as permutations, combinations, subsets, integer partitions, and so on. Regarding this listing task, efficiency is usually a main concern, and a common approach is to generate the objects such that successive elements differ in a small way. A classic example is the binary reflected Gray code [1, 2] which lists all n-bit binary numbers so that successive numbers differ in exactly one bit.

The term *combinatorial Gray code* first appeared in [3], and now stands for any generation of combinatorial objects such that successive objects differ in a usually small, or other specified way. Gray codes have applications in diverse areas as data compression [4], statistical computation [5], graphics and image processing [6], processor allocation in the hypercube [7], information storage and retrieval [8], etc. For an excellent survey on combinatorial Gray codes, please see [9]. In particular, for combination generation the applications include, among others, cryptography, genetic algorithms, statistical computation, and exhaustive

This work is supported in part by National Natural Science Foundation of China Grant 60553001 and National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

combinatorial searches.

In [10], the authors present a new algorithm for generating combinations by prefix rotations. If we represent combinations as bitstrings of length $n = s + t$ containing s 0's and t 1's, and denote by $B(s,t) = \{b_1 b_2 \cdots b_n \in \{0,1\}^n : \sum_{i=1}^{n} b_i = t\}$ the combination set, the algorithm generates the combinations with a remarkably simple rule: identify the shortest prefix ending in 010 or 011 (or the entire bitstring if no such prefix exists) and then rotate it to the right by one position. The rotation, which is equivalent to a cyclic permutation $\sigma_k = (1, 2, \cdots, k)$ $(2 \leq k \leq n)$ acting on the indices of the bitstring, is called a *prefix shift*. Please see the following as an example of listing $B(3,3)$.

$$\begin{aligned}
&\hookrightarrow {}_{\sigma_6}111000 \rightarrow {}_{\sigma_4}011100 \rightarrow {}_{\sigma_2}101100 \rightarrow {}_{\sigma_3}110100\\
&\rightarrow {}_{\sigma_5}011010 \rightarrow {}_{\sigma_4}101010 \rightarrow {}_{\sigma_4}010110 \rightarrow {}_{\sigma_3}001110\\
&\rightarrow {}_{\sigma_3}100110 \rightarrow {}_{\sigma_4}110010 \rightarrow {}_{\sigma_6}011001 \rightarrow {}_{\sigma_2}101001\\
&\rightarrow {}_{\sigma_4}010101 \rightarrow {}_{\sigma_3}001101 \rightarrow {}_{\sigma_3}100101 \rightarrow {}_{\sigma_5}010011\\
&\rightarrow {}_{\sigma_3}001011 \rightarrow {}_{\sigma_4}000111 \rightarrow {}_{\sigma_4}100011 \rightarrow {}_{\sigma_5}110001
\end{aligned}$$

The above generating algorithm has several remarkable properties, we mention some of them in the following (see [10] for more discussions). First, successive combinations differ by a prefix shift, which makes the algorithm very suitable for hardware implementation, and very fast in the situation where combinations are stored in a single computer word. In addition, the listing is *cyclic*, that is the generating rule also applies between the last and the first bitstrings. Second, successive combinations differ by one or two transpositions of a 0 and a 1. There are other algorithms with even more restricted operations between successive combinations. E.g., successive combinations differ by a single transposition [11], only zeros exist between the transposed bits [12], or the transposed bits have at most one bit between them [13]. Along with the one in [10], all these algorithms are discussed in Knuth [14]. Third, the algorithm has an efficient *loopless* implementation (see [15]). Finally, the new Gray code also has a simple ranking function whose running time is $O(n)$ arithmetic operations. However, in general this algorithm requires all the $n-1$ prefix shifts σ_k for $k = 2, 3, \cdots, n$.

One open problem regarding the power and limitation of prefix shifts in generating combinations is proposed in [10], that is, whether the number of different prefix shifts used can be reduced. In particular, can we generate combinations by only letting the three basic permutations σ_2, σ_n and ${\sigma_n}^{-1}$ act on the indices of the bitstrings? In this paper we make a step toward settling this problem by giving a negative answer to the latter question. Previous works on similar or related subjects are, among others, [16, 17].

In this paper we refer to operations of using the inverse of permutations (i.e., ${\sigma_k}^{-1}$ for $2 \leq k \leq n$) to act on the indices of bitstrings also as prefix shifts. We also use "prefix shift σ_k" $(2 \leq k \leq n)$ for short, to stand for the straightforward meaning when it is clear from the context.

2 Preliminaries

It is easy to see that, for the problem of listing each element once from a class of combinatorial objects such that successive objects differ in a specified way, there is a corresponding Hamiltonian path or cycle problem: for each object *obj* there is a vertex $v(obj)$ in the corresponding graph, and there is

an edge joining $v(obj_1)$ and $v(obj_2)$ if and only if obj_2 and obj_1 differ from each other in the specified way. The graph has a Hamiltonian path if and only if the required listing of the objects exists, and the graph has a Hamiltonian cycle corresponds to a cyclic listing, that is a listing in which the first and last objects also differ in the specified way.

In this paper we denote combinations as bit-strings of length $n = s + t$ containing s 0's and t 1's, and denote the set of combinations by $B(s,t) = \{b_1 b_2 \cdots b_n \in \{0,1\}^n : \sum_{i=1}{}^n b_i = t\}$. Also, we use $G(s,t)$ to denote the corresponding graph (sometimes we use G to denote the graph if it is clear from the context). First we list some facts that will be useful later.

Proposition 1 Suppose P is a Hamiltonian path of graph G, and a node $v \in G$ has degree 2 and v is not an endpoint of P, then the two nodes adjacent to v in P must be the two neighbors of v in the original graph G.

Proof: The conclusion follows immediately. □

Proposition 2 Suppose a node $v \in G$ has degree 3, with v_1, v_2 and v_3 as its three neighbors, and both v_1 and v_2 have degree 2, and suppose P is a Hamiltonian path of G. If neither of v_1 and v_2 is an endpoint of P, then the edge connecting v and its third neighbor v_3 cannot appear in P.

Proof: Since both v_1 and v_2 have degree 2 and they are not endpoints of P, by proposition 1, both v_1 and v_2 are adjacent to v in the Hamiltonian path P, it follows that v_3 can no longer be a neighbor of v in path P, thus the edge connecting v and v_3 cannot appear in P.□

Proposition 3 Suppose C is a simple cycle which is a subgraph of G, and suppose P is a Hamiltonian path of G. If for every two adjacent nodes in C there is at least one of them having degree 2 in G, then C must contain at least one endpoint of P.

Proof: For the sake of contradiction, assume that C contains no endpoint of P. Consider any node $v \in C$, since v is not an endpoint of P, it has two neighbors in P, we prove that they must be the two neighbors of v in cycle C.

There are two cases. If v has degree 2 in G, by proposition 1, the conclusion follows. If v has degree greater than 2, consider the two neighbors of v in C, v_1 and v_2, they must both have degree 2 in G. Since v_1 and v_2 are not endpoints of P, they must both be adjacent to v in path P, thus they are the two neighbors of v in P.

Now consider node u, the first node in P that belongs to C, then u is not an endpoint of P. By the above argument the node before u in P must also belong to C, which contradicts with the way we pick u. □

Proposition 4 Suppose P is a Hamiltonian path of $G(s,t)$, where $s,t \geq 2$, and $l \in B(s,t)$ is a string in which there is no single occurrence of 0 or 1, i.e., l can be partitioned into blocks of contiguous bits, each block is comprised of the same digits and is of size at least two. Let $C(l)$ denote the cycle in $G(s,t)$ corresponding to the sequence l, $\sigma_n(l)$, $\sigma_n{}^{(2)}(l)$, $\cdots$, then $C(l)$ contains at least one endpoint of P.

Proof: For any string l_0 in the above sequence, if the degree of $v(l_0)$ in G is 3, then l_0 must start

with 01 (or 10). Since there is no single occurrence of 0 or 1 in l, the two strings corresponding to the two nodes adjacent to $v(l_0)$ in $C(l)$ (i.e., ${\sigma_n}^{-1}(l_0)$ and $\sigma_n(l_0)$) must start with 11 and 00 (or 00 and 11), therefore the two nodes adjacent to $v(l_0)$ in $C(l)$ must both have degree two in G. By proposition 3, there must be a node in $C(l)$ which is an endpoint of P. □

3 The Main Results

In this section we will prove our main result, which is the following theorem. We prove the theorem by dividing it into three lemmas.

Theorem 1 One can generate combinations $B(s,t)$ by using prefix shifts σ_2, σ_n and ${\sigma_n}^{-1}$ if and only if $min\{s,t\} \leq 2$.

Lemma 5 If $min\{s,t\} \leq 2$, then $B(s,t)$ can be generated by using prefix shifts σ_2, σ_n and ${\sigma_n}^{-1}$.

Proof of Lemma 5: Without loss of generality, assume that $t \leq 2$. For $t = 1$, we can simply generate $B(s,t)$ by a sequence of $n-1$ σ_n's:

$$10^s \rightarrow_{\sigma_n} 010^{s-1} \rightarrow_{\sigma_n} \cdots \rightarrow_{\sigma_n} 0^s 1.$$

For $t = 2$, there are two cases according to the parity of n. If n is odd, we can generate $B(s,t)$ by starting with 110^s, then applying the sequence of $n-1$ σ_n's and one σ_2 $\frac{n-3}{2}$ times, finally followed by a sequence of $n-1$ σ_n's. If n is even, we also start with 110^s, and then apply the sequence of $n-1$ σ_n's and one σ_2 $\frac{n-2}{2}$ times, finally followed by a sequence of $\frac{n-2}{2}$ σ_n's. It is easy to see that in this way we can list all elements of $B(s,t)$ without repetition.

$$\begin{aligned}
110^s &\rightarrow_{\sigma_n} 0110^{s-1} \rightarrow_{\sigma_n} \cdots \rightarrow_{\sigma_n} 10^s 1 \rightarrow_{\sigma_2} \\
010^{s-1}1 &\rightarrow_{\sigma_n} 1010^{s-1} \rightarrow_{\sigma_n} \cdots \rightarrow_{\sigma_n} 10^{s-1}10 \rightarrow_{\sigma_2} \\
010^{s-2}10 &\rightarrow_{\sigma_n} 0010^{s-2}1 \rightarrow_{\sigma_n} \cdots \rightarrow_{\sigma_n} 10^{s-2}100 \rightarrow_{\sigma_2} \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
010^{\lceil \frac{s}{2} \rceil}10^{\lfloor \frac{s}{2} \rfloor - 1} &\rightarrow_{\sigma_n} 0010^{\lceil \frac{s}{2} \rceil}10^{\lfloor \frac{s}{2} \rfloor - 2} \rightarrow_{\sigma_n} \cdots .
\end{aligned}$$

□

Lemma 6 If $min\{s,t\} = 3$, then the underlying graph G of $B(s,t)$ with allowed operations σ_2, σ_n and ${\sigma_n}^{-1}$ has no Hamiltonian path.

Proof of Lemma 6: Without loss of generality, assume that $t = 3$ and $s \geq 3$. Assume that there is a Hamiltonian path P in G. Define $a_i = 10^i 1^2 0^{s-i}$, $i = 1, 2, \cdots, s$. It is easy to see that for each $1 \leq i \leq s$, the following sequence a_i, $\sigma_n(a_i)$, ${\sigma_n}^{(2)}(a_i)$, $\cdots$, ${\sigma_n}^{(n-1)}(a_i)$ contains n distinct strings, thus it corresponds a cycle of size n in G, we denote the cycle by $C(a_i)$. Moreover, these s cycles $C(a_1)$, $C(a_2)$, $\cdots$, $C(a_s)$ are vertex-disjoint (notice that we cannot simply apply proposition 4 here to argument that we get the contradiction that P has at least s endpoints, since adjacent vertices in $C(a_i)$ can have degree three), and they connect each other in G by the s edges $v(a_i)v(\sigma_2(a_i))$, $i = 1, 2, \cdots, s$, along with an extra edge $v(1010^{s-1}1)v(01^20^{s-1}1)$ (see Figure 1). Notice that the extra edge $v(1010^{s-1}1)v(01^20^{s-1}1)$ will not exist when $t > 3$, see Figure 2.

Let $b_i = 01^20^{s-i}10^{i-1}$, $c_i = 10^{s-i}10^i1$, $i = 1, 2, \cdots, s-1$. Let A be the set of nodes covered by the above s cycles $C(a_i)$, $i = 1, 2, \cdots, s$, and let O be the set of the other nodes in G.

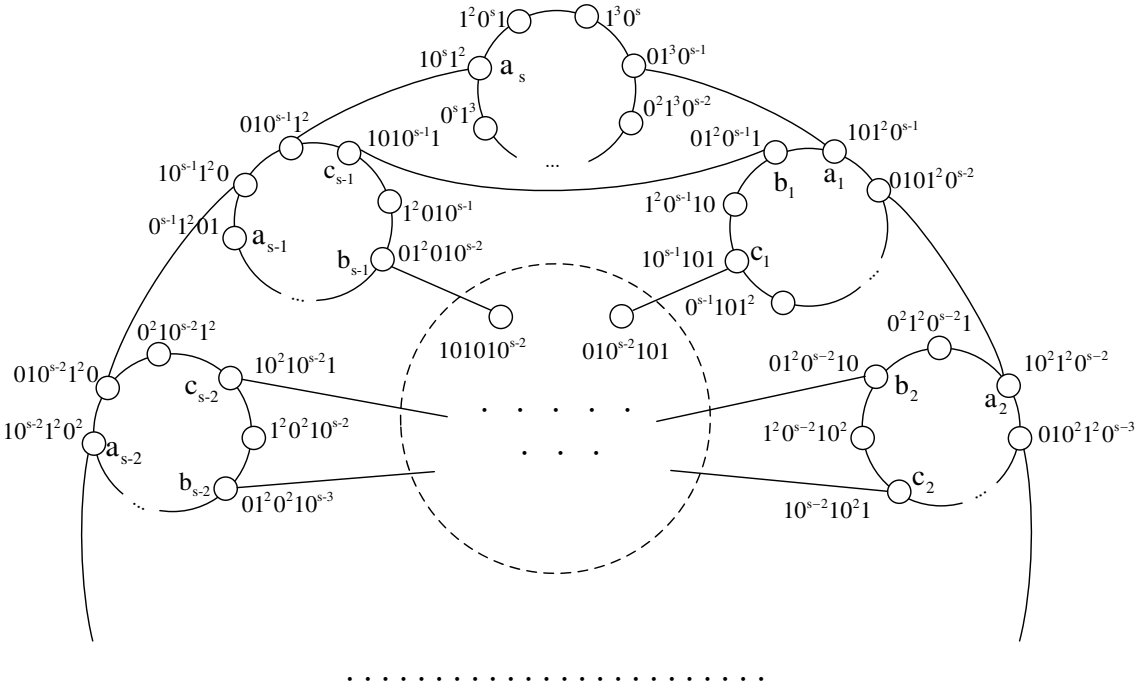

Fig. 1. Underlying graph of $B(s,t)$ with operations σ_2, σ_n and ${\sigma_n}^{-1}$ $(s \geq 3, t = 3)$

Since $s \geq 3$, O is non-empty. It is easy to see that there are $2s-4$ edges $v(b_2)v(\sigma_2(b_2))$, $\cdots$, $v(b_{s-1})v(\sigma_2(b_{s-1}))$, $v(c_1)v(\sigma_2(c_1))$, $\cdots$, $v(c_{s-2})v(\sigma_2(c_{s-2}))$ connecting one node from A and one node from O, and the nodes $v(b_2)$, $v(b_3)$, $\cdots$, $v(b_{s-1})$ and $v(c_1)$, $v(c_2)$, $\cdots$, $v(c_{s-2})$ are the endpoints of these edges in A. For any node $v \in C(a_i)$ for some $1 \leq i \leq s$, let $N_C(v)$ be the set of the two nodes adjacent to v in the corresponding cycle $C(a_i)$ containing v, and denote $N_C = N_C(v(b_2)) \cup N_C(v(b_3)) \cup \cdots \cup N_C(v(b_{s-1})) \cup N_C(v(c_1)) \cup N_C(v(c_2)) \cup \cdots \cup N_C(v(c_{s-2}))$.

Since P is connected, among the above $2s-4$ edges connecting A and O there must be at least one appearing in P. Assume that the edge $v(l)v(\sigma_2(l)) \in P$, where $l \in \{b_2, b_3, \cdots, b_{s-1}, c_1, c_2, \cdots, c_{s-2}\}$. Notice that all nodes in N_C have degree 2 in G. By proposition 2, since $v(l)v(\sigma_2(l)) \in P$, there must be one node from $N_C(v(l))$ which is an endpoint of P. Since P has two endpoints and by proposition 4 at least one of them is contained in cycle $C(a_s)$, it follows that there is exactly one node from N_C which is the other endpoint of P.

Next we will show that among the above $2s-4$ edges connecting A and O, there can be at most one appearing in P. Let v_0 denote the unique endpoint of P such that $v_0 \in N_C$, there are two cases:

Case 1. v_0 is not a common neighbor of any two nodes $v(b_i)$ and $v(c_i)$, for $2 \leq i \leq s-2$. Then by proposition 2, among the above $2s-4$ edges only the one with an endpoint having v_0 as its neighbor could appear in P.

Case 2. v_0 is a common neighbor of $v(b_i)$ and $v(c_i)$ in cycle $C(a_i)$, for some $2 \leq i \leq s-2$. Then only the two edges $v(b_i)v(\sigma_2(b_i))$ and $v(c_i)v(\sigma_2(c_i))$ could possibly appear in P.

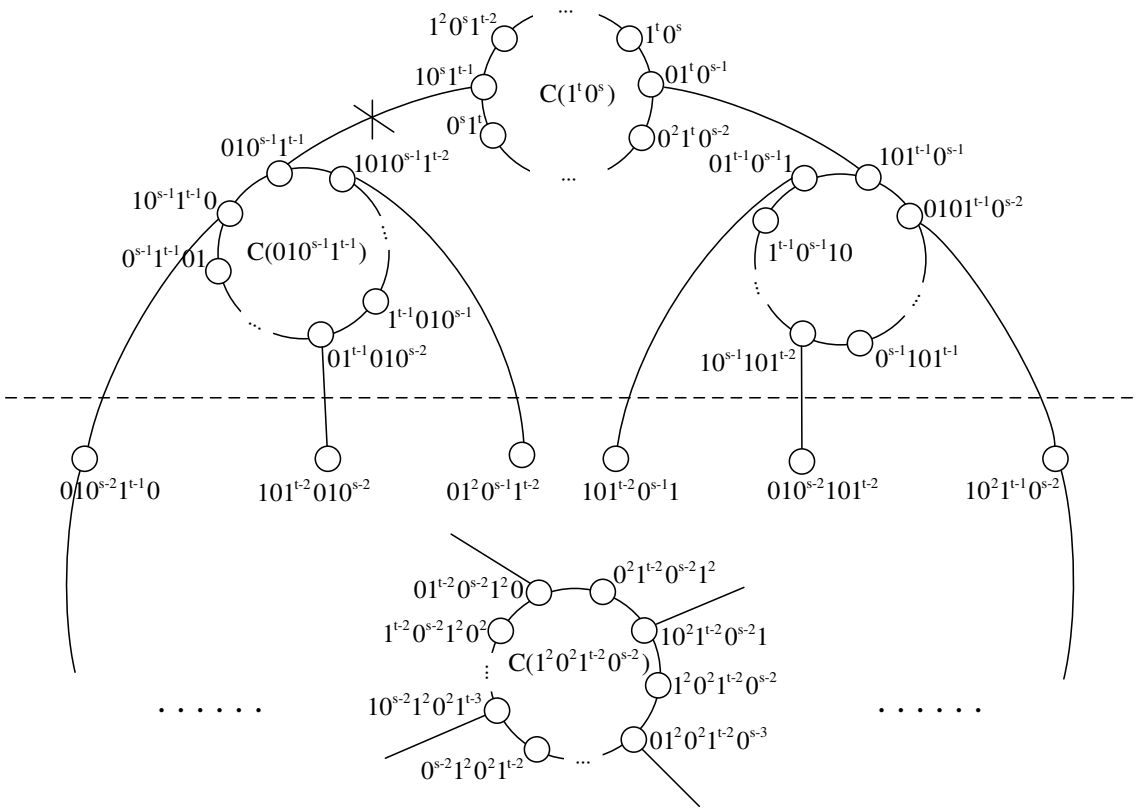

Fig. 2. Underlying graph of $B(s,t)$ with operations σ_2, σ_n and ${\sigma_n}^{-1}$ $(s,t \geq 4)$

Without loss of generality, assume that v_0 is adjacent to $v(b_i)$ in P, since the node in $N_C(v(b_i))$ other than v_0 (which has degree 2 in G) can no longer be an endpoint of P, it must be adjacent to $v(b_i)$ in P, thus $v(b_i)$ already has two neighbors in P and so the edge $v(b_i)v(\sigma_2(b_i))$ cannot appear in P, thus $v(c_i)v(\sigma_2(c_i))$ is the only possible edge that appears in P.

Therefore, in either case, among the above $2s-4$ edges connecting A and O there is exactly one of them appearing in P. However, this implies that the node set O must contain an endpoint of P, which is a contradiction since P cannot have three endpoints.□

Lemma 7 If $min\{s,t\} \geq 4$, then the underlying graph G of $B(s,t)$ with allowed operations σ_2, σ_n and ${\sigma_n}^{-1}$ has no Hamiltonian path.

Proof of Lemma 7: Assume that there is a Hamiltonian path P in G. For any string $l \in B(s,t)$, let $C(l)$ denote the cycle in G corresponding to the sequence l, $\sigma_n(l)$, ${\sigma_n}^{(2)}(l)$, $\cdots$. Since $s,t \geq 4$, by proposition 4, both cycles $C(1^t0^s)$ and $C(1^20^21^{t-2}0^{s-2})$ contain at least one endpoint of P. It is easy to see that $C(1^t0^s)$ and $C(1^20^21^{t-2}0^{s-2})$ are vertex-disjoint, therefore each of them contains exactly one endpoint of P.

Consider the two nodes of $C(1^t0^s)$ having degree three, $v(10^s1^{t-1})$ and $v(01^t0^{s-1})$, since $C(1^t0^s)$ contains only one endpoint of P, among the above two nodes there must exist one such that its two neighbors in $C(1^t0^s)$ are not endpoints of P, without loss of generality, assume it is $v(10^s1^{t-1})$. By proposition 2, the edge connecting $v(10^s1^{t-1})$ and its third neighbor in G, $v(010^{s-1}1^{t-1})$, will not

appear in P. Thus we can remove this edge from G, and the resulting graph will still contain the Hamiltonian path P (see Figure 2). After removing the edge $v(10^s1^{t-1})v(010^{s-1}1^{t-1})$, consider the cycle $C(010^{s-1}1^{t-1})$, now it contains only three nodes with degree three, $v(10^{s-1}1^{t-1}0)$, $v(1010^{s-1}1^{t-2})$ and $v(01^{t-1}010^{s-2})$, and they are not adjacent to each other. Therefore, by proposition 3, $C(010^{s-1}1^{t-1})$ also contains at least one endpoint of P. However, $C(1^t0^s)$, $C(1^20^21^{t-2}0^{s-2})$ and $C(010^{s-1}1^{t-1})$ cover disjoint node sets in G, thus P must have at least three endpoints, which is a contradiction. □

Proof of Theorem 1: It follows immediately from lemmas 5, 6 and 7. □

4 Conclusion and Future Studies

In this paper we showed that in general prefix shifts σ_2, σ_n and ${\sigma_n}^{-1}$ are not sufficient for generating combinations. Combination generation is of wide applications, and prefix shifts are basic operations which are very suitable for hardware implementation. It is interesting to further investigate whether we can generate combinations by using some more restricted class of prefix shifts.

Acknowledgments

We would like to thank the reviewers for their helpful comments and suggestions.

References

[1] Gilbert E N. Gray Codes and paths on the n-cube. *Bell Systems Technical Journal*, 37 (1958) 815–826.

[2] Gray F. Pulse code communication. *U.S. Patent Number* 2632058, 1953.

[3] Joichi J T, White D E, Willamson S G. Combinatorial Gray Codes. *SIAM Journal on Computing*, 9 (1980) 130-141.

[4] Richards D. Data compression and Gray-code sorting. *Information Processing Letters*, 22 (1986) 201-205.

[5] Diaconis P, Holmes S. Gray codes for randomization procedures. *Statistics and Computing*, 4 (1994) 287-302.

[6] Amalraj D J, Sundararajan N, Dhar G. A data structure based on Gray code encoding for graphics and image processing. *Proceedings of the SPIE: International Society for Optical Engineering*, (1990) 65-76.

[7] Chen M, Shin K. Subcube Allocation and Task Migration in Hypercube Multiprocessors. *IEEE Transactions on Computers*, C-39, (1990) 1146-1155.

[8] Chang C C, Chen H Y, Chen C Y. Symbolic Gray Code As A Data Allocation Scheme For Two-Disc Systems. *Computer Journal*, 35(3): (1992) 299-305.

[9] Savage C. A survey of combinatorial gray codes. *SIAM Review*, 39(4): 605–629, 1997.

[10] Ruskey F, Williams A. Generating Combinations by Prefix Shifts. *Proc. 11th Annual International Computing and Combinatorics Conference (COCOON'2005), LNCS 3595*, Springer, (2005) 570-576.

[11] Tang D T, Liu C N. Distance-2 Cycle Chaining of Constant Weight Codes. *IEEE Transactions*, C-22 (1973) 176C180.

[12] Eades P, McKay B. An Algorithm for Generating Subsets of Fixed Size with a Strong Minimal Change Property. *Information Processing Letters*, 19 (1984) 131-133.

[13] Chase P J. Combination generation and Graylex ordering. *Congressus Numerantium*, 69 (1989) 215-242.

[14] Knuth D E. The Art of Computer Programming. Pre-fascicle 4A (a draft of Section 7.2.1.3: Generating all Combinations), Addison-Wesley, 2004, 61 pages (http://www-cs-faculty.stanford.edu/ knuth/fasc3a.ps.gz).

[15] Ehrlich G. Loopless Algorithms for Generating Permutations, Combinations, and Other Combinatorial Configurations. *Journal of the ACM*, 20 (1973) 500-513.

[16] Chinburg T, Savage C D, Wilf H S. Combinatorial families that are exponentially far from being listable in Gray code sequence, *Transactions of the American Mathematical Society*, 351 (1999) 379-402.

[17] Compton R C, Williamson S G. Doubly adjacent Gray codes for the Symmetric group: how to braid n strands. *Linear and Multilinear Algebra*, 34 (1993) 237-293.