Simple Combinatorial Gray Codes
Constructed by Reversing Sublists

Frank Ruskey

Department of Computer Science, University of Victoria,
Victoria, B.C., V8W 3P6, Canada, e-mail: fruskey@csr.uvic.ca

Abstract. We present three related results about simple combinatorial
Gray codes constructed recursively by reversing certain sublists. First,
we show a bijection between the list of compositions of Knuth and the
list of combinations of Eades and McKay. Secondly, we provide a short
description of a list of combinations satisfying a more restrictive closeness
criteria of Chase. Finally, we develop a new, simply described, Gray code
list of the partitions of a set into a fixed number of blocks, as represented
by restricted growth sequences. In each case the recursive definition of
the list is easily translatable into an algorithm for generating the list in
time proportional to the number of elements in the list; i.e., each object
is produced in $O(1)$ amortized time by the algorithm.

1 Introduction

Frank Gray patented the Binary Reflected Gray Code (BRGC) in 1953 for use
in “pulse code communication”, but the underlying construction of the code
existed for centuries as the solution of a puzzle that is known in the West as
the “Chinese Rings”. The BRGC has since found use in the minimization of
logic circuits, in hypercube architectures, in the construction of multikey hashing
functions, and has even been proposed for use in the organization of books on
library shelves! For the purposes of this paper, we regard the BRGC as a way
of listing all $2^n$ bitstrings of length $n$ so that successive strings differ by a single
bit or, equivalently, as a way of listing all subsets of an $n$-set so that successive
subsets differ by a single element.

It is natural to take a more abstract view and look for listings of other
combinatorial objects so that successive objects differ by a very small amount;
such lists have come to be known as combinatorial Gray codes. In this paper, we
shed new light on some previously known lists of combinations and compositions,
and develop new lists for combinations and set partitions. In each case the list has
a very simple recursive description. The simplicity of the recursion is mirrored
in simple proofs and simple efficient algorithms for generating the lists. These
algorithms generate the objects so that only a constant amount of computation
is done between successive objects, in an amortized sense.

* Research supported in part by the Natural Sciences and Engineering Research Coun-
cil of Canada under Grant A3379.
There is a well-known simple recursive construction for the BRGC. This construction involves the concatenation of two recursively defined smaller sublists, one of which is reversed, as shown below.

\[
B(n) = \begin{cases} 
\emptyset & \text{if } n = 0 \\
B(n-1) \cdot 0 \circ B(n-1) \cdot 1 & \text{if } n > 0
\end{cases}
\]

This simple recursive construction is easily proven to be correct and can be implemented in an efficient manner to actually generate the list. In the construction of other combinatorial Gray codes, it is natural to try to invent similar recursive definitions. In general, the recursion defining the combinatorial Gray code should mirror some simple recursion for counting the combinatorial objects, in this case, \( a_n = 2a_{n-1} \) with \( a_0 = 1 \).

Very simple constructions have been obtained in the cases of combinations (e.g., Reingold, Nievergelt, and Deo [ReNiDe] or Eades and McKay [EaMc]), compositions (e.g., Wilf [Wi]), and well-formed parentheses (e.g., Ruskey and Proskurowski [RuPr]). Less simple recursive constructions, that are nevertheless based on reversing sublists, have been obtained for numerical partitions (Savage [Sa]), and set partitions (Fill and Reingold [FiRe], Ehrlich [Eh]).

We regard the objects being listed as strings over some alphabet. If \( L \) is a list of strings and \( x \) is a symbol, then \( L \cdot x \) denotes the list of strings obtained by appending an \( x \) to each string of \( L \). For example if \( L = 01, 10 \) then \( L \cdot 1 = 011, 101 \).

If \( L \) and \( L' \) are lists then \( L \circ L' \) denotes the concatenation of the two lists. For example, if \( L = 01, 10 \) and \( L' = 11, 00 \), then \( L \circ L' = 01, 10, 11, 00 \).

For a list \( L \), let \( \text{first}(L) \) denote the first element on the list and let \( \text{last}(L) \) denote the last element on the list \( L \). If \( L \) is a list \( l_1, l_2, \ldots, l_n \), then \( \overline{L} \) denotes the list obtained by listing the elements of \( L \) in reverse order; i.e., \( \overline{L} = L^{-1} = l_n, \ldots, l_2, l_1 \). Note the obvious equations \( \text{first}(\overline{L}) = \text{last}(L) \) and \( \text{last}(\overline{L}) = \text{first}(L) \).

By a \textit{k-combination} of \( n \) we mean a bitstring of length \( n \) with exactly \( k \) ones. The set of all \( k \)-combinations of \( n \) is denoted \( B(n, k) \); i.e.,

\[
B(n, k) = \{ b_1 b_2 \cdots b_n \mid b_i \in \{0, 1\}, \Sigma b_i = k \}
\]

By a \textit{k-composition} of \( n \) we mean a solution \( (x_1, x_2, \ldots, x_k) \), in non-negative integers, to the equation \( x_1 + x_2 + \cdots + x_k = n \).

A \textit{partition} of \([n] = \{1, 2, \ldots, n\}\) into \( k \) \textit{blocks} is a collection of \( k \) non-empty disjoint sets \( B_0, B_1, \ldots, B_{k-1} \) whose union is \([n]\). Assuming that the \( B_i \) are ordered by their smallest elements, a convenient representation of the partition is as a sequence \( x_1 x_2 \cdots x_n \) where \( x_i = j \) if \( x_i \in B_j \). For example, the sequence corresponding to the partition \{1, 3, 4\}, \{2, 6\}, \{5, 7\} is 0100212. Such sequences are known as \textit{restricted growth sequences} (or RG sequences) in the combinatorial literature (e.g., Milne [Mi]) and are characterized by the following property, which explains the name. If \( i = 1 \) then \( x_i = 0 \), and if \( i > 1 \), then

\[
0 \leq x_i \leq 1 + \max\{x_1, x_2, \ldots, x_{i-1}, k - 2\}.
\]