Available online at www.sciencedirect.com
ScienceDirect
Journal of Discrete Algorithms 4 (2006) 633–648
JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

A loop-free two-close Gray-code algorithm for listing k-ary Dyck words

Vincent Vajnovszki [a], Timothy Walsh [b,*]

[a] *LE21 FRE-CNRS 2309, Université de Bourgogne, B.P. 47 870, 21078 Dijon-Cedex, France*
[b] *Department of Computer Science, University of Quebec At Montreal, P.O. 8888, Station A, Montreal, Quebec, Canada, H3C 3P8*

Available online 30 August 2005

Abstract

P. Chase and F. Ruskey each published a Gray code for length n binary strings with m occurrences of 1, coding m-combinations of n objects, which is two-close—that is, in passing from one binary string to its successor a single 1 exchanges positions with a 0 which is either adjacent to the 1 or separated from it by a single 0. If we impose the restriction that any suffix of a string contains at least $k-1$ times as many 0's as 1's, we obtain k-suffixes: suffixes of k-ary Dyck words. Combinations are retrieved as special case by setting $k=1$ and k-ary Dyck words are retrieved as a special case by imposing the additional condition that the entire string has exactly $k-1$ times as many 0's as 1's. We generalize Ruskey's Gray code to the first two-close Gray code for k-suffixes and we provide a loop-free implementation for $k \geqslant 2$. For $k=1$ we use a simplified version of Chase's loop-free algorithm for generating his two-close Gray code for combinations. These results are optimal in the sense that there does not always exist a Gray code, either for combinations or Dyck words, in which the 1 and the 0 that exchange positions are adjacent.

Keywords: Gray code; k-ary Dyck words; Two-close; Loop-free algorithm

* Corresponding author. Tel.: +1 (514) 987-3000, extension 6139; fax: +1 (514) 987-8477.
E-mail addresses: vvajnov@u-bourgogne.fr (V. Vajnovszki), walsh.timothy@uqam.ca (T. Walsh).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2005.07.003

1. Introduction

Combinatorial objects such as subsets of the n-set $\{1, 2, \ldots, n\}$, m-combinations (subsets of cardinality m) of the n-set, permutations of the n-set, binary trees and the more general k-ary trees (rooted trees of which every node has either k children or none) can be coded by words on a finite alphabet. To study various properties of these combinatorial objects one can generate a list of code-words of a fixed length, representing the objects of a fixed size, and test them all for these properties. The smaller the difference between consecutive code-words in the list, the less time it takes to generate each new word and to update the properties being studied in passing from one object to the next. It is therefore useful to put the set of code-words of fixed length into an order that minimizes the greatest difference between two consecutive code-words.

We call a family of word lists in which all the words in each list are of the same length a *Gray code* if the family contains arbitrarily long words but the number of positions in which two consecutive words in any list differ is bounded independently of the word length. An algorithm for generating a Gray code is called *loop-free* [5] if the number of operations necessary to transform each word into its successor in its list, or to determine that the current word is the last one in its list, is bounded independently of the word length.

A *binary string* is a word on the alphabet $\{0, 1\}$. The binary reflected Gray code, published by F. Gray [6], is a family of lists, one for each n, of length n binary strings in which consecutive strings in any list differ by only one letter. J.R. Bitner, G. Ehrlich and E.M. Reingold [1] used an auxiliary array to obtain a loop-free algorithm for implementing this Gray code.

An m-combination of the n-set can be coded by a binary string w with m 1's and $n - m$ 0's. Alternatively, we can use the length m array whose ith component is the position in w of the ith occurrence of 1 in w. We call this representation the 1-*vector* of w and we define the 0-*vector* analogously; we extend these definitions to lists of binary strings. A Gray code for combinations is called *minimal* if each binary string can be transformed into its successor by exchanging a single 1 with a 0. There are several minimal Gray codes in the literature for combinations. The simplest of these Gray codes [10] is called the Liu–Tang Gray code after its authors C.N. Liu and D.T. Tang; in the 1-vector (list) of this Gray code, at most two letters change from one 1-vector to the next. A loop-free implementation of this Gray code was obtained by T.R. Walsh [22] and possibly other researchers as well. Another Gray code for combinations was discovered and given a loop-free implementation by Ehrlich [5].

A minimal Gray code for combinations is called *homogeneous* if the 1 and the 0 that exchange positions are separated only by 0's, implying that in the 1-vector only a single letter changes value from one 1-vector to the next. Such a Gray code was discovered by P. Eades and B. McKay [4]; a non-recursive description and a loop-free implementation of this Gray code appear in [21].

A homogeneous Gray code for combinations is called *two-close* if the 1 and the 0 that exchange positions are either adjacent or separated by a single 0, implying that in the 1-vector the letter that changes value does so by at most 2. Such a Gray code is optimal in the sense that for some values of m and n there does not exist a Gray code in which the 1 and 0 that exchange positions are always adjacent [14]. A two-close Gray code for com-