Analysis of Algorithms for Listing Equivalence Classes of k-ary Strings

Andrzej Proskurowski* Frank Ruskey† Malcolm Smith‡

May 28, 1996

Abstract

We give efficient algorithms for listing equivalence classes of k-ary strings under reversal and permutation of alphabet symbols. As representative of each equivalence class we choose that string which is lexicographically smallest. These algorithms use space $O(n)$ and time $O(\sqrt{k}N)$, where N is the total number of strings generated and n is the length of each string. For $k = 2$, we obtain a recursive decomposition of the set of binary strings that allows the strings to be generated without rejecting any strings. For $k \geq 3$, some strings must be rejected. The algorithm is simple but its exact analysis is rather complicated. In the analysis we determine a quantity of independent interest: the average length of the common prefix of two randomly chosen infinite length "restricted-growth" strings.

Keywords: combinatorial generation, k-ary strings, Stirling numbers, restricted growth function, k-paths.

AMS Classification: 05A15, 05C30, 60J10, 68Q25, 68R05, 68R15.

Running Title: Listing k-ary strings.

---

*Department of Computer and Information Science, University of Oregon, Eugene, Oregon 97403, USA, andrzej@cs.uoregon.edu. Research supported in part by grant NSF-CCR-9213431.

†Department of Computer Science, University of Victoria, Victoria, B.C., Canada V8W 3P6, fruskey@csr.uvic.ca. Research supported in part by NSERC grant OGP0003379.

‡Department of Computer Science, University of Victoria, Victoria, B.C., Canada V8W 3P6.

List of Symbols

$(k)_j$: falling factorial $(k)_j = k(k-1)\cdots(k-j+1)$.
$\left\{ {n \atop k} \right\}$: Stirling number of the second kind.

$\mathbf{B}_{n,k}$: all k-ary strings of length n. Also, $\mathbf{B}_n = \mathbf{B}_{n,2}$.
$B_{n,k} = |\mathbf{B}_{n,k}| = k^n$. Also, $B_n = B_{n,2}$.

s: An element of $\mathbf{B}_{n,k}$.

$can(s)$: The canonical (lexicographically least) string equivalent to s, by consistently permuting symbols.

$mate(s)$: The "mate" of s, $mate(s) = can(s^R)$.

$$\mathbf{X}_{n,k} = \{s \in \mathbf{B}_{n,k} \mid s = can(s)\}$$
$$\mathbf{X}_{n,k} = \{s_1 s_2 \cdots s_n \mid s_1 = 0, 0 \le s_i \le \min(k-1, 1 + \max(s_1, s_2, \ldots, s_{i-1}))\}$$

$X_{n,k} = |\mathbf{X}_{n,k}| = \sum_{i=1}^{k} \left\{ {n \atop i} \right\}$.

$\mathbf{Y}_{n,k} = \{s \in \mathbf{X}_{n,k} \mid s \le mate(s)\}$. Also, $\mathbf{Y}_n = \mathbf{Y}_{n,2}$.
$Y_{n,k} = |\mathbf{Y}_{n,k}|$. Also, $Y_n = Y_{n,2}$.

π_s: The permutation of $0, 1, \ldots, k-1$ that takes s into $can(s)$.

$c(s)$: The length of the largest prefix common to both s and $mate(s)$.

$\mathbf{X}_{n,k,p} = \{s_1 s_2 \cdots s_n \mid s_1 = p, 0 \le s_i \le \min(k-1, 1 + \max(s_1, s_2, \ldots, s_{i-1}))\}$
$X_{n,k,p} = |\mathbf{X}_{n,k,p}|$.

$\mathbf{X}_{n,k}^{(c)} = \{s \in \mathbf{X}_{n,k} \mid c(s) = c\}$.
$X_{n,k}^{(c)} = |\mathbf{X}_{n,k}^{(c)}|$

$\mathbb{Z}_n$: Group of integers mod n.
$\mathbb{C}_n$: Cyclic group of order n.
$\mathbb{S}_k$: Symmetric group of order $k!$.

1 Introduction

What are the most natural group actions on strings over a fixed alphabet? Three actions immediately suggest themselves: (a) leaving the string unchanged, (b) reversing a string, (c) rotating a string, and (d) permuting the symbols of the string by a permutation of the alphabet. The three groups giving rise to these actions are (a) $\mathbb{Z}_1$, (b) $\mathbb{Z}_2$, (c) the cyclic group $\mathbb{C}_n$ and (d) the symmetric group $\mathbb{S}_k$, assuming the alphabet consists of k symbols.

Each group action, or composition of group actions, partitions the set of k-ary strings into equivalence classes, namely the orbits of the action. To generate these equivalence classes, it is natural to choose as representative the lexicographically smallest string. With this representation, efficient algorithms are known for generating the equivalence classes of each of the actions (a), (b), (c), and (d). By "efficient" we mean that the amount of computation used in generating the objects is proportional to the number of objects generated. For (a) we are simply counting in base k which is known to be efficient for $k \geq 2$. For (b) efficient algorithms were developed by Ruskey [15]. In case (c) the equivalence classes are usually called *necklaces*. Efficient algorithms for generating necklaces were developed by Fredericksen and Kessler [2] and Fredericksen and Maiorana [3]; these algorithms were proven to be efficient by Ruskey, Savage, and Wang [14]. In case (d) the representative strings are usually called *restricted growth functions* and efficient algorithms for generating them have been developed by Er [1], Kaye [8], and others.

In contrast to the case where our three non-trivial actions are considered in isolation, the composition of more than two of the actions gives rise to equivalence classes for which no efficient generation algorithms were previously known. For example, composing (b) and (c) results in the dihedral group, with the resulting equivalence classes known as *bracelets*. No more efficient algorithm is known than simply listing all necklaces and rejecting those that are larger than their reversals (i.e., are not representative of a bracelet). In this paper we compose (b) and (d) and develop efficient (for fixed k) algorithms for generating the resulting equivalence classes. It remains an interesting challenge to develop efficient algorithms for the other compositions.

Let us recast our problem. The problem is to list equivalence classes of k-ary strings under the transitive closure of the binary relation $\mathcal{R}$ defined on k-ary strings of length n by

$$x_1 \cdots x_n \;\mathcal{R}\; y_1 \cdots y_n \Leftrightarrow \begin{cases} x_1 \cdots x_n = y_n \cdots y_1 \text{ or} \\ x_1 \cdots x_n = \pi(y_1) \cdots \pi(y_n) \text{ for some } \pi \in \mathbb{S}_k \end{cases} \tag{1}$$

As representative of each equivalence class we choose the lexicographically smallest string.

An important tool in listing sets of combinatorial objects consists of recurrence relations describing the sets. The typical operation on lists is *concatenation* ($\oplus$) corresponding to *addition* ($+$) of cardinalities and to *union* of disjoint sets, ($\uplus$). Strings are denoted using lower case bold letters. Sets and lists of strings are indicated by upper case bold letters, with their corresponding cardinalities indicated by upper case (non-bold) roman letters. Thus, as a trivial example, the set of all binary strings of length n will be denoted $\mathbf{B}_n$ fulfilling the recurrence relation

$$\mathbf{B}_n = 0\cdot\mathbf{B}_{n-1} \uplus 1\cdot\mathbf{B}_{n-1} \qquad \text{for } n > 0, \tag{2}$$

where $\mathbf{B}_0 = \{\epsilon\}$; the corresponding recurrence relation for its cardinality is $B_n = 2B_{n-1}$ with $B_0 = 1$, which has the closed form solution $B_n = 2^n$. Additionally, strings can be *reversed.* The reversal of a string $\mathbf{s}$ is denoted $\mathbf{s}^R$; this notation is extended to sets of strings in the natural manner. For example, $\{0011, 0101\}^R = \{1100, 1010\}$. If $\mathbf{s}$ and $\mathbf{t}$ are strings then by $\mathbf{s} \le \mathbf{t}$ we mean that $\mathbf{s}$ is lexicographically less than or equal to $\mathbf{t}$; comparisons among strings are always with respect to lexicographic order.

An algorithm for generating combinatorial objects is said to be CAT, standing for *C*onstant *A*mortized *T*ime, if it has running time $O(p(n) + N)$, where N is the total number of objects generated, and $p(n)$ is a polynomial in n, where n is the "size" of the object being generated. The term $p(n)$ is meant to represent any time spent in preprocessing. In this paper N is exponential in n and so the $p(n)$ term can be ignored. The algorithms that we consider are recursive and the underlying computation tree provides a great conceptual aid in analyzing the behavior of the algorithm. In many algorithms the total amount of computation is proportional to the number of nodes in the computation tree; the algorithms we present have this feature. A desirable property of a generation algorithm is that an object is produced at every leaf (terminal vertex) of the computation tree. We call this the BEST[1] property (for Backtracking Ensuring Success at Terminals). The *CAT principle*

[1]The CAT and BEST acronyms are due to the second author.

states sufficient conditions for a BEST algorithm to run in constant amortized time. These conditions are: (a) in the computation tree, there are not "too many" nodes in the computation tree with a single child and (b) the total computation can be partitioned so that each node is assigned constant time computation. In particular, if the degree of each non-leaf is at least two, then there are more leaves than internal nodes and so condition (a) is met. The CAT principle is the most common way of showing that a recursive algorithm runs in constant amortized time.

In Section 2 we present a BEST CAT algorithm for the binary case $k = 2$. In Section 3 we present a non-BEST algorithm that is CAT for fixed $k \geq 2$; this algorithm is analyzed in Sections 4 and 5. The analysis of Section 4 provides an exact count of the number of operations used by the algorithm for $k = 3$ and $k = 4$. In principle this type of analysis could be extended to larger values of k. The asymptotic analysis of Section 5 is based on the expected length of the longest common prefix of two infinite canonical k-ary strings. This quantity, which turns out to be $O(\sqrt{k})$, is of independent mathematical interest. In Section 6 we indicate how our algorithms solve the problem of listing non-isomorphic k-paths, which was our initial motivation for studying the problem. The final section suggest some interesting open problems.

2 Generating Binary Strings

In order to analyze equivalence classes under the group actions of symbol permutations (*i.e.,* complementation in the binary case) and under string reversal, we shall look first at the the problem of listing all equivalence classes under reversal only. That is to say, two bitstrings $a_1a_2\cdots a_n \neq b_1b_2\cdots b_n$ are considered equivalent if $a_1a_2\cdots a_n = b_n\cdots b_2b_1$. Given two equivalent bitstrings we will list the one that is lexicographically (i.e., numerically) smaller. Thus, for example, 1010011 is listed but 1100101 is not. Let the set of listed bitstrings be denoted $\mathbf{M}_n$.

$$\mathbf{M}_n \stackrel{\text{def}}{=} \{\mathbf{s} \in \mathbf{B}_n \mid \mathbf{s} \leq \mathbf{s}^R\} \tag{3}$$

The recurrence relation for this set follows the observation that only identical first and last bits allow for symmetry under reversal. Thus, $\mathbf{M}_0 = \{\epsilon\}$, $\mathbf{M}_1 = \{0, 1\}$, and

$$\mathbf{M}_n \;=\; 0\mathbf{M}_{n-2}0 \;\uplus\; 0\mathbf{B}_{n-2}1 \;\uplus\; 1\mathbf{M}_{n-2}1. \tag{4}$$

We use $\mathbf{L}_n$ to denote those elements of $\mathbf{M}_n$ that are non-palindromic.

$$\mathbf{L}_n \stackrel{\text{def}}{=} \{s \in \mathbf{B}_n \mid s < s^R\} = \{s \in \mathbf{M}_n \mid s \neq s^R\}$$

These sets of strings satisfy a recurrence relation of the same form as (4), but with different initial conditions. Here, $\mathbf{L}_0 = \mathbf{L}_1 = \emptyset$. The recurrence relation is

$$\mathbf{L}_n = 0\mathbf{L}_{n-2}0 \uplus 0\mathbf{B}_{n-2}1 \uplus 1\mathbf{L}_{n-2}1. \tag{5}$$

If s is a bitstring then by $\overline{s}$ we denote the complement of s. For example, $\overline{1010011} = 0101100$.

Observation 1: The operations of complement and reversal commute. That is, for all binary strings s, we have $\overline{s^R} = \overline{s}^R$

Observation 2: For all binary strings s and t, we have $s \leq t$ if and only if $\overline{s} \geq \overline{t}$.

Observation 3: For all binary strings, if $s \leq s^R$ and $s \leq \overline{s}^R$, then $s \leq \overline{s}$. [2]

Observations 1 and 2 are trivial. To prove Observation 3, note that by Observation 2, if $s \leq s^R$, then $\overline{s} \geq \overline{s}^R$. Thus, $\overline{s} \geq \overline{s}^R \geq s$.

Let $\mathbf{Y}_n$ denote the set of all bitstrings of length n that are lexicographically smallest in the equivalence classes induced by relation $\mathcal{R}$ defined by (1) in the introduction. Let $\mathbf{N}_n$ be the set of all non-palindromic elements of $\mathbf{Y}_n$. For example, $\mathbf{N}_4 = \{0010, 0001, 0011, 0101\}$ and $\mathbf{Y}_4 = \mathbf{N}_4 \cup \{0000, 0110\}$. More formally,

$$\mathbf{Y}_n \stackrel{\text{def}}{=} \{s \in \mathbf{B}_n \mid s \leq \overline{s}, s \leq s^R, \text{ and } s \leq \overline{s}^R\} \quad \text{and} \tag{6}$$

$$\mathbf{N}_n \stackrel{\text{def}}{=} \{s \in \mathbf{Y}_n \mid s \neq s^R\}. \tag{7}$$

The sets $\mathbf{Y}_n$ and $\mathbf{N}_n$ admit the following recursive decompositions.

Lemma 1 *For all $n > 1$,*

$$\mathbf{Y}_n = 0\mathbf{Y}_{n-2}1 \uplus 0\mathbf{N}^R_{n-2}1 \uplus 0\mathbf{M}_{n-2}0, \quad \textit{and} \tag{8}$$

$$\mathbf{N}_n = 0\mathbf{Y}_{n-2}1 \uplus 0\mathbf{N}^R_{n-2}1 \uplus 0\mathbf{L}_{n-2}0 \tag{9}$$

The initial values are $\mathbf{Y}_0 = \{\epsilon\}$, $\mathbf{Y}_1 = \{0\}$, and $\mathbf{N}_0 = \mathbf{N}_1 = \emptyset$.

[2] Also, $s \leq \overline{s}$ and $s \leq s^R$ implies that $s \leq \overline{s}^R$. However, the example of $s = 0100$ shows that $s \leq \overline{s}$ and $s \leq \overline{s}^R$ does not imply $s \leq s^R$.

Proof: To prove (8), first note that a string s in $\mathbf{Y}_n$ cannot start with a 1. If $s = 0t0$, then $s \le \bar{s}$ and $s \le \bar{s}^R$, so the only remaining condition that has to be satisfied is $s \le s^R$. This is true if and only if $t \le t^R$. Thus, a string of the form $0t0$ is in $\mathbf{Y}_n$ if and only if $t \in \mathbf{M}_{n-2}$. If $s = 0t1$, then $s \le s^R$ and $s \le \bar{s}$, so the only remaining condition that has to be satisfied is $s \le \bar{s}^R$. By Observation 3, $\mathbf{Y}_n = \{s \in \mathbf{B}_n \mid s \le s^R, s \le \bar{s}^R\}$. By Observation 2, the condition $s \le \bar{s}^R$ is equivalent to the condition $s^R \le \bar{s}$. Thus,

$$\begin{aligned} \{s \in \mathbf{B}_n \mid s \le \bar{s}^R\} &= \{s \in \mathbf{B}_n \mid (s \le s^R \text{ and } s \le \bar{s}^R) \text{ or } (s^R \le s \text{ and } s^R \le \bar{s})\} \\ &= \mathbf{Y}_n \cup \mathbf{Y}_n^R. \end{aligned}$$

Hence, a string of the form $0t1$ is in $\mathbf{Y}_n$ if and only if $t \in \mathbf{Y}_{n-2} \cup \mathbf{Y}_{n-2}^R$. Recurrence relation (8) follows by partitioning $\mathbf{Y}_{n-2} \cup \mathbf{Y}_{n-2}^R$ into the two disjoint sets $\mathbf{Y}_{n-2}$ and $\mathbf{N}_{n-2}^R$.

The proof of (9) is similar and is omitted. □

These equations allow us to write a program for listing the elements of $\mathbf{Y}_n$ that runs in constant amortized time. One could write separate procedures for listing the elements of $\mathbf{Y}_n$, $\mathbf{N}_n$, $\mathbf{L}_n$, $\mathbf{M}_n$, and $\mathbf{B}_n$. Pascal code for $\mathbf{Y}_n$ is given in Figure 1. The array `s[1..n]` of bits is global. Each procedure has two parameters `lo` and `hi` indicating the subarray of `s` whose bits still remain to be set. The initial call is `Y(1,n)`, with no initialization necessary. Note that, aside from the recursive calls, only a constant amount of computation is done within `Y`. Further, any non-leaf call to `Y` has at least two (in fact three) children in the computation tree. Since each of the recurrences (2), (4), (5), (8), and (9) has at least two terms these same observations apply to other procedures called by `Y`. Thus we can invoke the CAT principle to conclude that the algorithm runs in constant amortized time.

3 Generating k-ary strings for $k > 2$

In this section we develop an algorithm for generating the lexicographically smallest k-ary strings from each equivalence class induced by the actions of reversing a string and permuting the symbols of its alphabet. We first generate the lexicographically smallest k-ary strings from from each equivalence class induced only by the action of permuting alphabet symbols, and then reject those strings that are not also lexicographically smallest when both group actions are used.

```
procedure Y ( lo, hi : integer );
begin
  if lo = hi then begin
     s[lo] := 0;  {s[hi] := 0;}  PrintIt;
  end else
  if lo+1 = hi then begin
     s[lo] := 0;   s[hi] := 0;   PrintIt;
     s[lo] := 0;   s[hi] := 1;   PrintIt;
  end else begin
     s[lo] := 0;   s[hi] := 0;   Mr( lo+1, hi-1 );
    {s[lo] := 0;}  s[hi] := 1;   Nr( lo+1, hi-1 );
    {s[lo] := 0;} {s[hi] := 1;}   Y( lo+1, hi-1 );
  end
end {of Y};
```

Figure 1: Pascal code for generating $\mathbf{Y}_n$.

Definition 1 *If* s *is a* k*-ary string, then by* $\mathrm{can}(s)$ *we denote the lexicographically smallest string* $\pi(s)$*, taken over all permutations* π *of* $\{0, 1, \ldots, k-1\}$*. If* $s = \mathrm{can}(s)$*, then we say that* s *is* canonical*. The set of canonical* k*-ary strings of length* n *is denoted* $\mathbf{X}_{n,k}$*.*

For example, $\mathrm{can}(660240032644) = 001231142033$. There may be several permutations π for which $\pi(s) = \mathrm{can}(s)$ (we think of the permutation π as an element of a group acting on the string s). In the example above, $\pi = (0\ 1\ 6)(2)(3\ 4)(5)$ and $\pi = (0\ 1\ 5\ 6)(2)(3\ 4)$ are such permutations. In order to define a unique permutation π_s such that $\pi_s(s) = \mathrm{can}(s)$, we assume that all elements of $0, 1, \ldots, k-1$ not used in s or in $\mathrm{can}(s)$ form 1-cycles in π_s. Thus $\pi_s = (0\ 1\ 6)(2)(3\ 4)(5)$ in our example. We will write permutations in cycle notation, because it is the cycle structure of these permutations that will be of most importance to us in the ensuing discussion.

The following lemma, whose proof is immediate, characterizes canonical strings.

Lemma 2 *A string* $s = s_1 s_2 \ldots s_n$ *is canonical if and only if* $s_1 = 0$ *and for all* $i = 1, \ldots, n-1$,

$$s_{i+1} \ \leq \ \min(k-1, 1 + \max\{s_1, \ldots, s_i\}). \tag{10}$$

Sequences satisfying the conditions of the Lemma 2 are sometimes called "restricted-growth" functions (e.g., Stanton and White [17]) and are studied in connection with set partitions. The Stirling number of the second kind, denoted $\left\{{n \atop k}\right\}$ count the number of restricted growth functions of length n with maximal value k. Let $X_{n,k} = |\mathbf{X}_{n,k}|$. It is well-known that

$$X_{n,k} = \sum_{i=1}^{k} \left\{{n \atop i}\right\}. \tag{11}$$

What is the asymptotic value of $X_{n,k}$ for fixed k and large n? Note that $X_{n,k}$ is the number of ways of placing n labeled balls into k unlabeled boxes and thus that $k^n/k! \leq X_{n,k}$ since there are at most $k!$ distinct ways of labeling the boxes once the balls have been placed. On the other hand, $\left\{{n \atop i}\right\} \leq i^n/i!$ since $i!\left\{{n \atop i}\right\}$ is the number of surjective functions from an n-set onto an i-set. Thus

$$\frac{k^n}{k!} \;\leq\; X_{n,k} \;\leq\; \sum_{i=1}^{k} \frac{i^n}{i!}.$$

From these upper and lower bounds we obtain the following asymptotic expression for $X_{n,k}$ (with k fixed and n large).

$$X_{n,k} \;\sim\; \frac{k^n}{k!} \tag{12}$$

Definition 2 *If* s *is a canonical* k*-ary string, then by* $\mathrm{mate}(s)$ *we denote the string* $\mathrm{can}(s^R)$*. If* $s = \mathrm{mate}(s)$*, then we say that* s *is* symmetric*. Let* $c(s)$ *denote the length of the longest prefix common to both of* s *and* $\mathrm{mate}(s)$*.*

Since the first symbol of both s and $\mathrm{mate}(s)$ is a zero, $1 \leq c(s) \leq n$, where n is the length of s. Continuing our example, if $s = 001231142033$, then $\mathrm{mate}(s) = \mathrm{can}(330241132100) = 001234402411$ and $c(s) = 5$ since the largest prefix on which they agree is 00123. An example of a symmetric string is 0010231011.

We define $\mathbf{Y}_{n,k} \stackrel{\text{def}}{=} \{s \in \mathbf{X}_{n,k} \mid s \leq \mathrm{mate}(s)\}$.

Lemma 2 gives rise to a simple-minded algorithm generating all canonical strings of length n. The algorithm, which we call gen, is given in Figure 2. A similar algorithm for generating all set partitions (i.e., when the maximum block size k is n) was given by Er [1], and other iterative algorithms are given by Hutchinson [7], Kaye [8], Semba [16], and Stanton and White [17].

The call `gen(l,m)` generates all sequences $s[l], \ldots, s[n]$ such that $0 \leq s[i] \leq \min(k-1, 1+\max\{m, s[l], \ldots, s[i-1]\})$ for $i = l, l+1, \ldots, n$. Here, parameter `m` is

```
procedure gen( l, m : integer );
var i : integer;
begin
   if l > n (* and Check *) then PrintIt
   else begin
       for i := 0 to m do begin
          s[l] := i;
          gen( l+1, m );
       end;
       if m < k-1 then begin
          s[l] := m+1;
          gen( l+1, m+1 );
       end;
   end;
end {of gen};
```

Figure 2: Pascal procedure to generate $\mathbf{X}_{n,k}$ (and $\mathbf{Y}_{n,k}$).

```
function Check : boolean;
var
  i, last: integer;
  pi : array[0..100] of integer;
begin
  for i := 0 to k - 1 do pi[i] := maxint; {#1}
  last := 0;
  i := n;
  pi[s[n]] := 0;
  while (i >= 1) and (s[n-i+1] = pi[s[i]]) do begin
     i := i - 1;
     if pi[s[i]] > last then begin
        last := last + 1;
        pi[s[i]] := last;
     end;
  end;
  Check := (i = 0) or (s[n-i+1] < pi[s[i]]);
end {of Check};
```

Figure 3: The Pascal function Check.

the maximum value of `s[1..l-1]`. Given n, the initial call is `gen(1,0)`. Assuming $k > 1$, note that every call to `gen(l,m)` for which $l \leq n$ generates at least two calls to `gen(l+1,i)` for some value of i. Thus there are more leaves than internal nodes in the computation tree and so the CAT principle implies that the algorithm runs in constant amortized time. Observe that the algorithm is BEST.

In order to generate the elements of $\mathbf{Y}_{n,k}$ we use `gen` and check whether each generated string s satisfies $s \leq mate(s)$. This checking is done by the procedure `Check` of Figure 3. With the addition of `Check`, algorithm `gen` is no longer BEST. Clearly, the amount of computation done by $\mathtt{Check}(s)$ is $O(k + c(s))$. The term k comes from the initialization (marked `#1` in Figure 3) but that term can be eliminated by using a constant-time array initialization as explained, for example, in Lewis and Denenberg [10]. With this modification the running time of `Check` is $O(c(s))$.

To find the amortized cost per string generated we need to determine the average value of $c(s)$, taken over all strings s in $\mathbf{X}_{n,k}$. Denoting by $X_{n,k}^{(c)}$ the number of strings $s \in \mathbf{X}_{n,k}$ for which $c(s) = c$, we define the weighted sum

$$S_{n,k} \stackrel{\text{def}}{=} \sum_{c=1}^{n} c \cdot X_{n,k}^{(c)}. \tag{13}$$

Define

$$A_{n,k} \stackrel{\text{def}}{=} \frac{S_{n,k}}{X_{n,k}}. \tag{14}$$

The quantity of interest is $S_{n,k}/Y_{n,k}$, which is at most $2A_{n,k}$.

4 Exact analysis for small values of k

In this section we give a general classification scheme useful in deriving exact expressions for $S_{n,k}$. Given a canonical string s, this classification is based upon the value of $c(s)$ relative to the length of s and the cycle structure of π_s. This scheme is then used to derive exact expressions for $S_{n,3}$ and $S_{n,4}$. The details of these derivations may be found in [12].

We wish to count those strings $s \in \mathbf{Y}_{n,k}$ with a fixed value of $c(s)$. Let us first make a couple of observations. Consider a fixed canonical string s and let $c = c(s)$ and $\pi = \pi_{s^R}$. If $c < n$, then let $x = s_{c+1}$ and $y = s_{n-c}$. Clearly, $\pi(y) \neq x$. There are five basic cases to consider: [I] $1 \le c < (n-1)/2$, [II] n is odd and $c = (n-1)/2$, [III] n is even and $c = n/2$, [IV] $n/2 < c < n$, and [V] $c = n$.

α	x	β	y	γ

[I]: In this case x is to the left of y in s as illustrated above. We denote by α the string $s_1 s_2 \cdots s_c$, $\beta = s_{c+2} \cdots s_{n-c-1}$, and $\gamma = s_{n-c+1} \cdots s_n$.

α	x	γ

[II]: In this case $x = y$ as illustrated above. Clearly, we must have $\pi(x) \neq x$. Here $\alpha = s_1 s_2 \cdots s_m$ and $\gamma = s_{m+2} \cdots s_n$, where $n = 2m + 1$.

[III]: In cases III and IV, x is to the right of y in s. We must have $\pi(x) = y$ (and $\pi(y) \neq x$. Thus there is a j-cycle, where $j \ge 3$, in π of the form $(\ldots\ x\ \pi(x)\ \pi(\pi(x))\ \ldots) = (\ldots\ x\ y\ z\ \ldots)$ for some z distinct from x and y. Hence, we must have $k \ge 3$ for case III or IV to occur.

α	y	β	x	γ

[IV]: This case is illustrated above. Let β denote the string of symbols between x and y, $\beta = s_{n-c} \cdots s_c$. If $a = s_j$ and $b = s_{n-j+1}$ for some $n - c < j < c$, then we must have $\pi(a) = b$ and $\pi(b) = a$. Thus a and b are either both in 1-cycles $(a)(b)$ or a 2-cycle $(a\ b)$ of π. Hence, in order for case IV to occur (i.e., β is non-empty), we must have $k \ge 4$.

[V]: In case V the string is symmetric and π must consist of 1-cycles and 2-cycles.

4.1 Ternary strings

Here case [IV] cannot occur. For $k = 3$ and even n,

$$S_{n,3} = \frac{5}{16}3^n + \frac{1}{6}(2n-3)3^{n/2} + \frac{n}{4} + \frac{3}{16}.$$

For odd n,

$$S_{n,3} = \frac{5}{16}3^n + \frac{1}{2}(n-2)3^{(n-1)/2} + \frac{n-1}{4} + \frac{9}{16}.$$

The above expressions, together with (12) for $k = 3$ show that the average value of $c(s)$ tends to $3! \cdot 5/16 = 15/8 = 1.875$ as n gets large.

4.2 Quaternary Strings

We now show the results for $k = 4$. If $n = 2m$, then

$$S_{n,4} = \frac{26}{315}4^n + \frac{5}{12}m2^n + \frac{5}{18}2^n - \frac{15}{14}2^m + \frac{4}{3}m + \frac{32}{45}. \tag{15}$$

If $n = 2m + 1$ is odd then we have

$$S_{n,4} = \frac{26}{315}4^n + \frac{1}{3}m2^n + \frac{7}{18}2^n - \frac{9}{7}2^m + \frac{4}{3}m + \frac{53}{45}. \tag{16}$$

Thus, when $k = 4$, the average value of $c(s)$ tends to $24{\cdot}26/315 = 1.980952381...$ as n gets large.

In the next section we will show that asymptotically $A_{n,k}$ is $O(\sqrt{k})$ as $n \to \infty$.

5 Asymptotic Analysis

For fixed k, we now show how to determine the asymptotic value of $A_{n,k}$. We first show that, asymptotically, the average value of $c(s)$ is the same as the expected position of the first mismatch between two canonic infinite length k-ary strings.

Lemma 3 *Let* s *be chosen uniformly at random from* $\mathbf{X}_{n,k}$*. With probability tending to 1 as* n *increases,* $c(s) < n/2$*.*

Proof: If the mismatch occurs at a position greater than $n/2$, then the reversal of the last $n/2$ symbols of s is equivalent, under some permutation of the k alphabet symbols, to the first $n/2$ symbols of s. The number of such strings is thus at most $k!X_{n/2,k}$. By (12), the fraction of these strings in the set $\mathbf{X}_{n,k}$ is $k! \cdot k^{-n/2}$, which tends to 0 when n grows. $\square$

Denote by $c(s, t)$ be the length of the longest prefix common to both s and t. The previously used $c(s)$ is the same as $c(s, mate(s))$.

Lemma 4 *Let* s *be chosen uniformly at random from* $\mathbf{X}_{n,k}$ *and* $\mathbf{t}'$ *be chosen uniformly at random from* $\mathbf{B}_{n,k}$*. Let* $\mathbf{t} = can(\mathbf{t}')$*. Then*

$$\lim_{n\to\infty} A_{n,k} = \sum_{i\geq 1} i \cdot \textit{Prob}(c(s, \mathbf{t})) = i)$$

Proof: The fraction of strings in $\mathbf{X}_{n/2,k}$ that do not use the symbol k is $X_{n/2,k-1}/X_{n/2,k}$. By (12), this is asymptotically $k((k-1)/k)^{n/2}$, which tends to 0 as n grows. Thus, with probability tending to 1, every symbol $0, 1, \ldots, k-1$ occurs among the first $n/2$ symbols of a string s chosen uniformly at random from $\mathbf{X}_{n,k}$. Thus, asymptotically, the substring $s_{n/2} \cdots s_n$ of s is a random string in $\mathbf{B}_{n/2,k}$. By the previous lemma, $c(s) < n/2$ with probability tending to 1. Therefore, comparing s with its mate is asymptotically equivalent to comparing it with any random string in $\mathbf{B}_{n,k}$. $\square$

It proves useful to have a notation $X_{n,k,p}$ for the number of strings $s = s_1 s_2 \cdots s_n$ satisfying $s_1 = p$ and the restricted growth condition (10). Thus, $X_{n,k,0} = X_{n,k}$. These numbers have also been studied before in connection with ranking algorithms for restricted growth functions in lexicographic order (see Williamson [19], [20] where they are called "restricted tail coefficients"). These numbers satisfy the following recurrence relation for $n > 0$.

$$X_{n,k,p} \;=\; (p+1)X_{n-1,k,p} + X_{n-1,k,p+1} \tag{17}$$

Lemma 5 *The following two limits hold.*

$$\lim_{n\to\infty} \frac{X_{n,k,p}}{X_{n+1,k,p}} = \frac{1}{k} \tag{18}$$

$$\lim_{n\to\infty} \frac{X_{n,k,p+1}}{X_{n+1,k,p}} = 1 - \frac{p+1}{k} \tag{19}$$

Proof: Recall that by (12), $X_{n,k} \sim k^n/k!$. The lemma follows from the following asymptotic, which may be proven by induction using (17).

$$X_{n,k,p} \;\sim\; \frac{(k)_{p+1} k^{n-1}}{k!}$$

Note: by $(k)_j$ we denote the falling factorial power $(k)_j = k(k-1)\cdots(k-j+1)$. $\square$

Lemma 6 *Let s be an element of $\mathbf{X}_{n,k}$ chosen uniformly at random. Then*

$$\lim_{n\to\infty} \mathrm{Prob}_n(s_i = j \mid \max(s_1, \ldots, s_{i-1}) = p) \;=\; \begin{cases} 1/k & \textit{if } 0 \le j \le p \\ 1-(p+1)/k & \textit{if } j = p+1. \end{cases}$$

Proof: The number of strings in $\mathbf{X}_{n,k}$ whose largest value in the first $i-1$ positions is p is $\left\{ {i-1 \atop p} \right\} X_{n-i+2,k,p}$. The number of strings that, in addition, have a j in position i is $\left\{ {i-1 \atop p} \right\} X_{n-i+1,k,p}$ if $0 \le j \le p$, and is $\left\{ {i-1 \atop p} \right\} X_{n-i+1,k,p+1}$ if $j = p+1$. To complete the proof, divide each of the latter numbers by the first and apply Lemma 5 to the respective quotients. $\square$

Let s' be an infinite length k-ary string chosen uniformly at random and $s = can(s')$. Note that

$$\mathrm{Prob}(s_i = j \mid \max(s_1, \ldots, s_{i-1}) = p) \;=\; \begin{cases} 1/k & \text{if } 0 \le j \le p \\ 1-(p+1)/k & \text{if } j = p+1. \end{cases},$$

exactly the same as the probabilities given in Lemma 6 above.

Let s' and $\mathbf{t}'$ be two infinite k-ary strings chosen uniformly at random and let $s = can(s') = s_1 s_2 s_3 \cdots$ and $\mathbf{t} = can(\mathbf{t}') = t_1 t_2 t_3 \cdots$. We have proven that

$$A_k = \lim_{n\to\infty} A_{n,k} \;=\; \sum_{i\ge 1} i \cdot \mathrm{Prob}(c(s,\mathbf{t}) = i). \tag{20}$$

Computing and analyzing A_k using (20) is the subject of the next two subsections.

5.1 The value of A_k

The purpose of this section is to prove the following rather attractive expression for A_k (below, the notation $(k)_j^2$ means $[(k)_j]^2$, the square of the falling factorial).

Theorem 1 $$A_k \;=\; \sum_{j=1}^{k} \frac{(k)_j^2}{(k^2-1)_j}.$$

Select s' and $\mathbf{t}'$ uniformly at random from the set of infinite length k-ary strings and let $s = can(s')$ and $\mathbf{t} = can(\mathbf{t}')$. Denote by β_p the probability that the first p symbols of s and $\mathbf{t}$ agree.

$$\beta_i \stackrel{\text{def}}{=} \mathrm{Prob}(s_1 = t_1, s_2 = t_2, \ldots, s_i = t_i) \tag{21}$$

By classifying the prefix of the first i symbols of s' and $\mathbf{t}'$ according to j, the number of distinct symbols in that prefix, we obtain the following expression for β_i.

$$\beta_i \;=\; \frac{1}{k^{2i}} \sum_{j=1}^{k} (k)_j^2 \left\{ {i \atop j} \right\}$$

By (20),

$$
\begin{aligned}
A_k &= \sum_{i\geq 0} i \cdot \mathrm{Prob}(s_1 = t_1, s_2 = t_2, \ldots, s_i = t_i, s_{i+1} \neq t_{i+1}) \\
&= \sum_{i\geq 0} i \cdot (\beta_i - \beta_{i+1}) \\
&= \sum_{i\geq 0} i \sum_{j=1}^{k} (k)_j^2 \left(\frac{\left\{ {i \atop j} \right\}}{k^{2i}} - \frac{\left\{ {i+1 \atop j} \right\}}{k^{2i+2}} \right) \\
&= \sum_{j=1}^{k} (k)_j^2 \sum_{i\geq 0} \frac{1}{k^{2i}} \left\{ {i \atop j} \right\} \qquad (22)
\end{aligned}
$$

The inner sum of (22) can be simplified by using the following identity from Wilf [18] (equation (1.6.5) on page 19).

$$
\sum_n \left\{ {n \atop k} \right\} x^n = \frac{x^k}{(1-x)(1-2x)\cdots(1-kx)} \qquad (23)
$$

Setting x to k^{-2} and k to j in (23) and substituting into (22) finishes the proof of Theorem 1. It is remarkable that A_k has such a simple expression; perhaps there is a direct combinatorial proof.

5.2 The Asymptotic Value of A_k

Our goal in this section is to prove the following theorem.

Theorem 2 $\quad A_n = \frac{1}{2}\sqrt{\pi \cdot n} - \frac{1}{6} + o(1).$

In this section we will use A_n instead of A_k, as is customary in the asymptotics literature. Our main technique is called "trading tails" in Graham, Knuth, and Patashnik [4] (starting at page 452). In principle, our techniques could be applied to obtain additional terms in the asymptotic expansion of A_n. With this in mind, some of the expressions below are given with greater generality than is strictly necessary to prove Theorem 2.

We start by presenting Taylor's expansion of $\ln(1-z)$ and an identity involving the Bernoulli numbers. The expansion below is valid for $z \to 0$ (e.g., page 438 of [4]). This is just Taylor's expansion with remainder.

$$
\ln(1-z) = -\sum_{k=1}^{p} \frac{z^k}{k} + O(z^{p+1}). \qquad (24)
$$

From *Concrete Mathematics* [4] (exercise 9.30, page 477 with solution on page 570),

$$\sum_{r\geq 0} r^{l-1}e^{-r^2/n} = \frac{1}{2}n^{l/2}\Gamma(\frac{l}{2}) - \sum_{k=0}^{p}\frac{(-1)^k B_{l+2k}}{n^k(l+2k)!k!} + O(\frac{1}{n^{p+1}}), \tag{25}$$

where $B_0, B_1, B_2, B_3, B_4, \ldots = 1, -\frac{1}{2}, \frac{1}{6}, 0, -\frac{1}{30}, 0, \ldots$ are the Bernoulli numbers (see, for instance, Section 6.5 of [4]) and Γ is the gamma function with $\Gamma(1/2) = \sqrt{\pi}$, $\Gamma(3/2) = \frac{1}{2}\sqrt{\pi}$, $\Gamma(5/2) = \frac{3}{4}\sqrt{\pi}$, and $\Gamma(l) = (l-1)!$ if l is an integer.

In particular, setting $l = 1, 2, 4, 5$ in (25) yields

$$\sum_{r\geq 1} e^{-r^2/n} = \frac{1}{2}\sqrt{\pi n} - \frac{1}{2} + o(1), \tag{26}$$

$$\frac{1}{n}\sum_{r\geq 1} re^{-r^2/n} = \frac{1}{2} + o(1), \tag{27}$$

$$\frac{1}{n^2}\sum_{r\geq 1} r^3e^{-r^2/n} = \frac{1}{2} + o(1), \tag{28}$$

$$\frac{1}{n^3}\sum_{r\geq 1} r^4e^{-r^2/n} = o(1). \tag{29}$$

Recall that

$$A_n = \sum_{r=1}^{n}\frac{(n)_r^2}{(n^2-1)_r} = \sum_{r=1}^{n}\prod_{x=0}^{r-1}\frac{(n-x)^2}{n^2-x-1},$$

and define

$$T(n,r) \stackrel{\text{def}}{=} \frac{n^2}{n^2-r}\prod_{x=1}^{r-1}\frac{(n-x)^2}{n^2-x} = \frac{1}{1-\frac{r}{n^2}}\prod_{x=1}^{r-1}\frac{(1-\frac{x}{n})^2}{1-\frac{x}{n^2}}. \tag{30}$$

For any small fixed $\epsilon > 0$ we may write A_n as

$$A_n = \sum_{r=1}^{n} T(n,r) = \sum_{r=1}^{n^{1/2+\epsilon}} T(n,r) \ + \sum_{r=1+n^{1/2+\epsilon}}^{n} T(n,r) \tag{31}$$

Take logarithms of $T(n,r)$ as expressed in the right hand side of (30) and then use (24) to obtain the following equality, valid for any r, $r/n \to 0$ and any natural numbers p and q,

$$\begin{aligned}
\ln(T(n,r)) &= 2\sum_{x=1}^{r-1} \ln(1-\frac{x}{n}) - \sum_{x=1}^{r} \ln(1-\frac{x}{n^2}) \\
&= \left(-2\sum_{k=1}^{p} \frac{1}{kn^k} \sum_{x=1}^{r-1} x^k + O(\frac{r^{p+2}}{n^{p+1}})\right) + \left(\sum_{k=1}^{q} \frac{1}{kn^{2k}} \sum_{x=1}^{r} x^k + O(\frac{r^{q+2}}{n^{2q+2}})\right)
\end{aligned}$$

Setting $p = 2$ and $q = 0$ in the above expression, we obtain

$$\begin{aligned}
\ln(T(n,r)) &= \left(-\frac{r(r-1)}{n} - \frac{r(r-1)(2r-1)}{6n^2} + O(\frac{r^4}{n^3})\right) + \left(O(\frac{r^2}{n^2})\right) \\
&= -\frac{r^2}{n} + \frac{r}{n} - \frac{r^3}{3n^2} + O(\frac{r^4}{n^3}).
\end{aligned}$$

Thus, exponentiating the above and using the power series expansion of e^x,

$$\begin{aligned}
T(n,r) &= \exp(-\frac{r^2}{n}) \exp(\frac{r}{n}) \exp(-\frac{r^3}{3n^2}) \exp(O(\frac{r^4}{n^3})) \\
&= \exp(-\frac{r^2}{n})(1 + \frac{r}{n} + O(\frac{r^2}{n^2}))(1 - \frac{r^3}{3n^2} + O(\frac{r^6}{n^4}))(1 + O(\frac{r^4}{n^3})) \\
&= \exp(-\frac{r^2}{n})(1 + \frac{r}{n} - \frac{r^3}{3n^2} + O(\frac{r^4}{n^3})).
\end{aligned}$$

Substituting the four expressions (26–29) into the sum $\sum_r T(n,r)$ and using the last derived expression for $T(n,r)$, we obtain

$$\begin{aligned}
\sum_{r\geq 1} T(n,r) &= \sum_{r\geq 1} \exp(-\frac{r^2}{n})(1 + \frac{r}{n} - \frac{r^3}{3n^2} + O(\frac{r^4}{n^3})) \\
&= \frac{1}{2}\sqrt{n\pi} - \frac{1}{6} + o(1).
\end{aligned} \tag{32}$$

For any fixed integer j, small $\epsilon > 0$, $n^{1/2+\epsilon} \leq r \leq n$, and large enough n, note that

$$r^j e^{-r^2/n} < e^{-r^2/(2n)}.$$

Thus (following [4], page 474),

$$\begin{aligned}\sum_{r>n^{1/2+\epsilon}} r^j e^{-r^2/n} &< \sum_{r>n^{1/2+\epsilon}} e^{-r^2/(2n)} \\ &< \exp(-\lfloor n^{1/2+\epsilon}\rfloor^2/(2n))(1+e^{-1/(2n)}+e^{-2/(2n)}+\cdots) \\ &= O(e^{-\frac{1}{2}n^{2\epsilon}})\cdot O(n) \\ &= O(n^{-M}) \text{ for any fixed } M.\end{aligned}$$

Substituting $j = 0, 1, 3, 4$ into (32), we get

$$\sum_{r=1}^{n^{1/2+\epsilon}} T(n,r) = \frac{1}{2}\sqrt{n\pi} - \frac{1}{6} + o(1). \tag{33}$$

It remains only to show that the second sum on the right hand side of (31) is $o(1)$.

Lemma 7 *For all* $j \geq 0$ *and* $\epsilon > 0$,

$$\lim_{n\to\infty} n^j \left(1 - \frac{1}{n}\right)^{n^{1+\epsilon}} = 0.$$

Proof: Rewriting and exploiting continuity, we obtain

$$\begin{aligned}&\exp(\lim_{n\to\infty} \ln \left(n^j \left(1-\frac{1}{n}\right)^{n^{1+\epsilon}}\right)) \\ &= \exp(\lim_{n\to\infty} \left(j\ln n + n^{1+\epsilon}(\ln(n-1) - \ln n\right)) \\ &= \exp(\lim_{n\to\infty} \frac{jn^{-(1+\epsilon)}\ln n + \ln(n-1) - \ln n}{n^{-(1+\epsilon)}})\end{aligned}$$

The limit may now be evaluated by using L'Hopital's rule.

$$\begin{aligned}&= \exp(\lim_{n\to\infty} \frac{j(n^{-(2+\epsilon)} - (1+\epsilon)n^{-(2+\epsilon)}\ln n) + \frac{1}{n-1} - \frac{1}{n}}{-(1+\epsilon)n^{-(2+\epsilon)}}) \\ &= \exp(\lim_{n\to\infty} \left(-\frac{j}{1+\epsilon} + j\ln n - \frac{n^{2+\epsilon}}{n(n-1)(1+\epsilon)}\right)) \\ &= \exp(\lim_{n\to\infty} \left(O(1) + o(n^\epsilon) + \frac{n^\epsilon}{(n-1)(1+\epsilon)} - \frac{n^\epsilon}{1+\epsilon}\right)) \\ &= \exp(\lim_{n\to\infty} \left(o(n^\epsilon) - \frac{n^\epsilon}{1+\epsilon}\right)) \;=\; 0\end{aligned}$$

□

Lemma 8 *If* $r \geq n^{1/2+\epsilon}$ *where* $\epsilon > 0$*, then*

$$T(n,r) = O(n^{-M}), \quad \textit{for any fixed } M.$$

Proof: Note that the quantities $T(n,r)$ are monotonically decreasing in r, and thus we need only consider $T(n, n^{1/2+\epsilon})$. First, split the definitional product (30) into two factors and then bound each factor.

$$T(n, n^{1/2+\epsilon}) = \frac{n^2}{n^2 - n^{1/2+\epsilon}} \prod_{x=1}^{\lfloor \frac{1}{2} n^{1/2+\epsilon} \rfloor} \frac{(n-x)^2}{n^2 - x} \prod_{x=\lfloor \frac{1}{2} n^{1/2+\epsilon} \rfloor + 1}^{n^{1/2+\epsilon}} \frac{(n-x)^2}{n^2 - x}.$$

$$\begin{aligned} T(n, n^{1/2+\epsilon}) &\leq 2 \left(\frac{(n - \frac{1}{2} n^{1/2+\epsilon})^2}{n^2 - \frac{1}{2} n^{1/2+\epsilon}} \right)^{\frac{1}{2} n^{1/2+\epsilon}} \\ &= 2 \left(1 - \frac{n^{3/2+\epsilon} - \frac{1}{4} n^{1+2\epsilon} - \frac{1}{2} n^{1/2+\epsilon}}{n^2 - \frac{1}{2} n^{1/2+\epsilon}} \right)^{\frac{1}{2} n^{1/2+\epsilon}} \end{aligned}$$

This last expression grows like

$$2 \left(1 - \frac{1}{n^{1/2-\epsilon}} \right)^{\frac{1}{2} n^{1/2+\epsilon}}$$

which by Lemma 7 is $O(n^{-M})$ for any fixed M. □

From Lemma 8 it follows that

$$\sum_{r=1+n^{1/2+\epsilon}}^{n} T(n,r) \ = \ O(n^{-M}), \text{ for any fixed } M.$$

This last equation, together with (33) proves Theorem 2.

6 Application: generating k-paths

A generalization of trees yields the notion of k-*trees* as the skeletons of acyclic simplicial complexes for dimensions higher than 2: For a fixed k, the complete graph with $k+1$ vertices, K_{k+1}, is a k-tree, and every k-tree with more vertices can be constructed from a smaller k-tree by adding a new node and making it adjacent to all vertices of a K_k subgraph. Thus, every k-tree has at least two vertices of degree

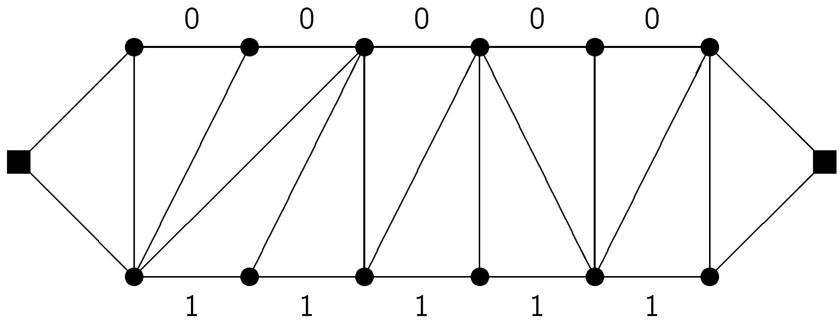

Figure 4: The 2-path corresponding to the binary string 0011011001.

k, k-*leaves*, and every *minimal separator* (a set of vertices disconnecting the graph, minimal with respect to set inclusion) consists of k mutually adjacent vertices [13]. (We will say that this set *induces* K_k.) In the simplest case of a k-tree with exactly two k-leaves, we have a generalization of a path, namely a k-*path*. The two k-leaves of a k-path (say, u and v) are connected by k unique vertex-disjoint paths and every minimal separator contains exactly one vertex of each path [11]. One can view the construction process of such a graph as starting with one k-leaf, say u, of K_{k+1}, and then adding vertices until the other desired k-leaf, say v, is added, completing the construction. Recording which of the (eventual) vertex disjoint paths are augmented during the process, gives the unique string over an alphabet of k symbols corresponding to the particular construction process (see Figure 4). A given k-path can be constructed in this manner from either of its ends (from u or from v) and vertex disjoint paths can be identified by any of the $k!$ permutations of symbols $0, 1, \ldots, k-1$. Thus the problem of listing all non-isomorphic k-paths can be dealt with as the problem of listing all equivalence classes of strings of k symbols, under the group actions of string reversal and/or symbol permutation.

7 Final Remarks

We list below some open problems that are inspired by or related to the problems considered in this paper.

1. Is there a direct combinatorial proof of the simple expression in Theorem 1 for A_k?

2. Is there a CAT BEST algorithm for generating the elements of $\mathbf{Y}_{n,k}$ when $k > 2$? The results of Section 2 show that $\mathbf{Y}$ is a context-free language for $k = 2$ and it is an unambiguous grammar for this language that leads us to the CAT algorithm. Can such a grammar be found for larger values of k?

3. Can these results be extended to generate all non-isomorphic k-trees? Here, even the counting problem is unsolved. Related enumeration (counting) results are reported in Harary and Palmer [5] and Hering, Read, and Shephard [6].

4. Does there exist a CAT algorithm to generate all k-ary strings where the group acting on the string contains rotations as well as permutations of the symbols and/or reversal of the string? The lexicographically smallest strings in classes of strings equivalent under rotations are referred to as necklaces. These are useful, for example, in generating de Bruijn sequences.

8 Acknowledgement

We wish to thank Andrew Odzlyko, and also Philippe Flajolet and Bruno Salvy for their help in proving Theorem 2. They responded to a query posted to the internet newsgroup `sci.math.research`. We also wish to thank Peter Winkler for helpful discussions and the referees for carefully reading the paper and making many useful suggestions.

References

[1] M.C. Er, "A Fast algorithm for generating set partitions", *The Computer Journal* 31 (1988) 283-284.

[2] H. Fredricksen and I.J. Kessler, "An algorithm for generating necklaces of beads in two colors", *Discrete Mathematics*, 61 (1986) 181-188.

[3] H. Fredricksen and I.J. Maiorana, "Necklaces of beads in k colors and k-ary de Bruijn sequences", *Discrete Mathematics*, 23 (1978) 207-210.

[4] R.L. Graham and D.E. Knuth and O. Patashnik, *Concrete Mathematics*, Addison-Wesley, 1989.

[5] F. Harary and E.M. Palmer, "On acyclic simplicial complexes", *Mathematika*, 15 (1968) 115-122.

[6] F. Hering, R.C. Read, and G.C. Shephard, "The enumeration of stack polytopes and simplicial clusters", *Discrete Mathematics*, 40 (1982) 203-217.

[7] G. Hutchinson, "Partitioning algorithms for finite sets", CACM, 6 (1963) 613-614.

[8] R.A. Kaye, "A Gray code for set partitions", *Information Processing Letters*, 5 (1976) 171-173.

[9] D.E. Knuth, *The Art of Computer Programming, vol. I, Fundamental Algorithms*, 2nd Ed., Addison-Wesley, 1973.

[10] H. R. Lewis and L. Denenberg, *Data Structures & Their Algorithms*, Harper-Collins, New York, 1991.

[11] A. Proskurowski, "Separating subgraphs in k-trees: Cables and Caterpillars", *Discrete Math. 49(1984), 275-285.*

[12] A. Proskurowski, F. Ruskey, and M. Smith, "Analysis of algorithms for generating k-paths", DCS-178, Tech. Report, Univ. of Victoria, 1992.

[13] D.J. Rose, "Triangulated graphs and the elimination process", *J. Math. Anal. Appl. 32 (1970) 597-609.*

[14] F. Ruskey, C.D. Savage, and T. Wang, "Generating necklaces", *J. Algorithms*, 13 (1992) 414-430.

[15] F. Ruskey, "Listing all bitstrings inequivalent under reversal", unpublished manuscript, 1990.

[16] I. Semba, "An efficient algorithm for generating all partitions of the set $\{1, 2, \ldots, n\}$", *Journal of Information Processing*, 7 (1984) 41-42.

[17] D. Stanton and D. White, *Constructive Combinatorics*, Springer-Verlag, 1986.

[18] H.S. Wilf, *Generatingfunctionology*, Academic Press, 1990.

[19] S. G. Williamson, *Combinatorics for Computer Science*, Computer Science Press, 1985.

[20] S. G. Williamson, "Ranking algorithms for lists of partitions", *SIAM J. Computing*, 5 (1976) 602-617.