Enumeration, Counting, and Random Generation of Ladder Lotteries

Katsuhisa Yamanaka[1,a)] Shin-ichi Nakano[2,b)]

Abstract: A *ladder lottery*, known as "Amidakuji" in Japan, is one of the most popular lotteries. In this paper, we consider the problems of enumeration, counting, and random generation of the ladder lotteries. For given two positive integers n and b, we give algorithms of enumeration, counting, and random generation of ladder lotteries with n lines and b bars.

1. Introduction

A *ladder lottery*, known as "Amidakuji" in Japan, is one of the most popular lotteries for kids. It is often used to assign roles to members in a group. Imagine that a group of four members A, B, C, and D wish to determine their group leader using a ladder lottery. First, four vertical lines are drawn, then each member chooses a vertical line. See Fig. 1(a). Next, a check mark (which represents an assignment of the leader) and some horizontal lines are drawn, as shown in Fig. 1(b). The derived one is called a ladder lottery, and it represents an assignment. In this example the leader is assigned to D since the top-to-bottom route from D ends at the check mark. (We will explain about the route soon.) In Fig. 1(b), the route is drawn as a dotted line.

Formally, a *ladder lottery* is a network with $n \geq 2$ vertical lines (*lines* for short) and b horizontal lines (*bars* for short) each of which connects two consecutive vertical lines. See Fig. 2 for an example. The top ends of the lines correspond to a permutation $\pi = (p_1, p_2, \ldots, p_n)$ of $[n] = \{1, 2, \ldots, n\}$, and the bottom ends of the lines correspond to the identity permutation $\iota = (1, 2, \ldots, n)$ and they satisfy the following rule. Each p_i in π starts the top end of the i-th line, then goes down along the line; whenever p_i meets an end of a bar, p_i goes horizontally along the bar to the other end, and then goes down again. Finally, p_i must reach the bottom end of the p_i-th line. Each bar corresponds to a modification of the current permutation by swapping the two neighboring elements.

A ladder lottery appears in a variety of areas. First, it is strongly related to primitive sorting networks, which are deeply investigated by Knuth [3]. Second, in algebraic combinatorics, a reduced decomposition of a permutation corresponds to a ladder lottery of a permutation with the minimum number of bars [4]. Third, a ladder lottery of the reverse permutation $(n, n-1, \ldots, 1)$

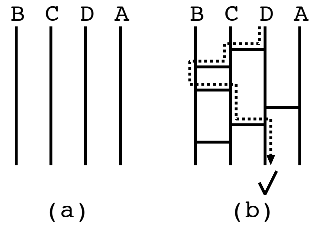

Fig. 1 An example of a ladder lottery.

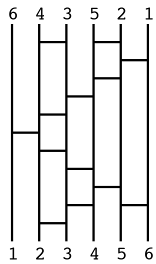

Fig. 2 A ladder lottery of (6,4,3,5,2,1).

corresponds to a pseudoline arrangement in discrete geometry [7].

In this paper we consider the problems of enumeration, counting, and random generation of ladder lotteries. We propose three algorithms for these three problems. All the three algorithms are based on the code [1] of ladder lotteries.

2. Preliminary

Code of ladder lotteries

We review a code of ladder lotteries in [1]. Using this code, we design three algorithms in this paper.

Let L be a ladder lottery with n lines and b bars. We first divide each bar of L into two horizontal line-segments, called *half-bars*. Fig. 3 illustrates the division of bars applied to the ladder lottery in Fig. 2. The left half of a bar is called an *l-bar* (left half-bar) and the right half of a bar is called an *r-bar* (right half-bar). We regard each original bar as a pair of an l-bar and an r-bar. Thus L has $2b$ half-bars. The division results in n connected components,

1 Department of Computer Science, Gunma University, Tenjin-cho 1-5-1, Kiryu, Gunma 376-8515, Japan.
2 Department of Electrical Engineering and Computer Science, Iwate University, Ueda 4-3-5, Morioka, Iwate 020-8551, Japan.
a) yamanaka@cis.iwate-u.ac.jp
b) nakano@cs.gunma-u.ac.jp

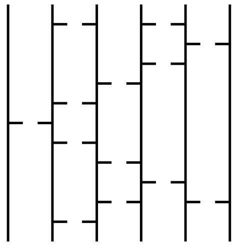

Fig. 3 Divisions of bars.

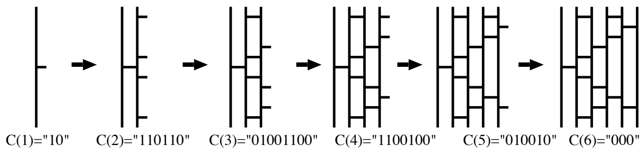

Fig. 4 An example of the reconstruction from the code

each of which consists of one line and some half-bars attached to the line.

We can encode how half-bars are attached to the i-th line, as follows. Let $\langle b_1, b_2, \ldots \rangle$ be the sequence of half-bars attached to the i-th line appearing from top to bottom. We replace b_i with `0` if b_i is an r-bar, and with `1` if b_i is an l-bar. Then appending a `0` to indicate the end-of-line. This results in the code of the i-th line, which is denoted by $C(i)$. Concatenating those codes $C(1), C(2), \ldots, C(n)$ results in the code $C(L)$ for L. For example, for the ladder lottery in Fig. 2 $C(1)$ = "`10`", $C(2)$ = "`110110`", $C(3)$ = "`01001100`", $C(4)$ = "`1100100`", $C(5)$ = "`010010`", $C(6)$ = "`000`", and

$$C(L) = \text{“}\texttt{1011011001001100110010001001000 0}\text{”}.$$

Since the code contains two bits for each bar and one bit for each end-of-line, its length is $n + 2b$ bits.

Reconstruction from the code

Now we explain how to reconstruct the original ladder lottery from the code.

In the code, a `0` represents either an r-bar or an end-of-line. Hence, we need to recognize the end-of-lines to partite $C(L)$ into $C(1), C(2), \ldots, C(n)$. After then, it is easy to reconstruct original bars by connecting corresponding l-bars and r-bars, since the k-th l-bar of the i-th line and the k-th r-bar of the $(i+1)$-th line correspond to an original bar. Fig. 4 shows an example of the reconstruction of the ladder lottery in Fig. 2 from its code.

We now explain how to recognize the end-of-lines. Since the first line has only l-bars, the first consecutive `1`s correspond to the l-bars of the first line, so the first `0` is the end-of-line of the first line. Now we assume that the end-of-line for the (i-1)-th line is recognized and we are now going to recognize the end-of-line for the i-th line. We know the number, say k, of l-bars attached to the (i-1)-th line, and it equals to the number of r-bars attached to the i-th line. Then the end-of-line for the i-th line is the $(k+1)$-th line `0` after the end-of-line for the (i-1)-th line.

Theorem 2.1 ([1]) Let L be a ladder lottery with n lines and b bars. One can encode L into a bitstring of length $n + 2b$. Both encoding and decoding can be done in $O(n + b)$ time.

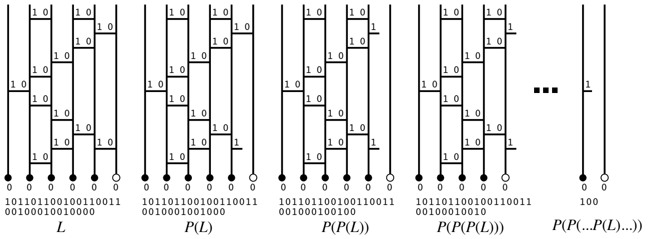

Fig. 5 Pre-ladders derived from L.

Pre-ladder and its code

Let L be a ladder lottery with n lines and b bars, and let $C(L)$ be the code of L. We define a substructure of L, as follows. Let $P(C(L))$ be the bitstring derived from $C(L)$ by removing the second last bit, and $P(L)$ be the substructure of L derived by "decoding" $P(C(L))$. Intuitively, $P(L)$ is the substructure of L only missing either a half-bar or an end-of-line, corresponding to the second last bit. Similarly $P(P(C(L)))$ is the bitstring derived from $P(C(L))$ by removing the second last bit, and $P(P(L))$ be the corresponding substructure of L. Similarly, we define $P(P(P(C(L)))), P(P(P(P(C(L))))), \ldots$. We assume that a pre-ladder has at least two lines. See Fig. 5 for an example. In the figure, end-of-lines are depicted as black circles except for the end-of-line of the rightmost line, which is depicted as a white circle. We say each of those substructure (including L itself) a *pre-ladder* of L, and the sequence $L, P(L), P(P(L)), \ldots$ the *removing sequence* of L. A pre-ladder possibly has unmatched l-bars only at the two rightmost lines.

3. Enumeration

Let $\mathcal{S}_{n,b}$ be the set of ladder lotteries with n lines and b bars. In this section, we design an algorithm that enumerates all ladder lotteries in $\mathcal{S}_{n,b}$.

Our enumeration algorithm is based on reverse search [2]. We first define a forest structure in which each leaf one-to-one corresponds to some ladder lottery in $\mathcal{S}_{n,b}$. Then, by traversing the forest, we can enumerate all leaves of the forest, and all corresponding ladder lotteries in $\mathcal{S}_{n,b}$. We designed several enumeration algorithms based on similar (but distinct) tree structures [5], [6], [7].

Family forest

Let L be a ladder lottery in $\mathcal{S}_{n,b}$. By merging the removing sequence for every $L \in \mathcal{S}_{n,b}$, we have the forest, called *family forest* $F_{n,b}$, in which each leaf one-to-one corresponds to some ladder lottery in $\mathcal{S}_{n,b}$. We regard each edge corresponds to some parent-child relation between the two pre-ladders. Each root is a pre-ladder with exactly two lines and no half-bar attached to the second line. See Fig. 6 for an example.

Child enumeration

We have the following lemma. See Appendix.

Lemma 3.1 Given any pre-ladder R in $F_{n,b}$, one can enumerate all child pre-ladders of R in O(1) time for each.

By recursively enumerating all child pre-ladders of a derived pre-ladder in $F_{n,b}$, we have the following theorem

Theorem 3.2 One can enumerate all ladder lotteries with n lines

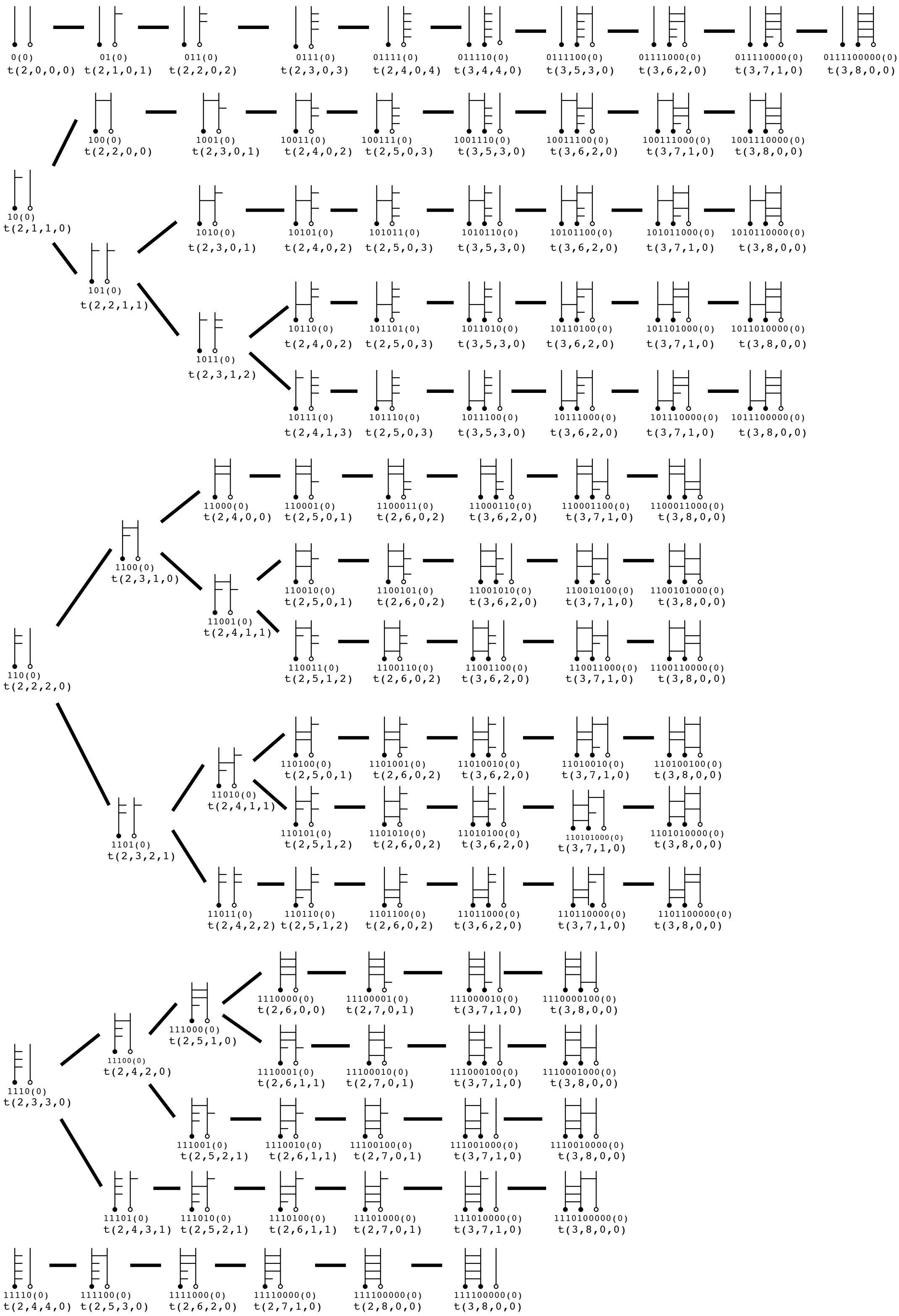

Fig. 6 The family forest $F_{3,4}$.

and b bars in $\mathrm{O}(n+b)$ time for each. Our algorithm uses $\mathrm{O}(n+b)$ space.

4. Counting

In this section we consider a counting problem. Given two positive integers $n \geq 2$ and $b \geq 0$, we wish to count the number of ladder lotteries with n lines and b bars. Using the enumeration algorithm in the previous section, we can count such ladder lotteries one by one, but very slowly. This method takes $\Omega(|\mathcal{S}_{n,b}|)$ time, which may be exponential on n and b. In this section, we propose an efficient counting algorithm. Our algorithm does not count ladder lotteries one by one, but counts each "type" of pre-ladders all together, and runs in polynomial time.*1

We now define the type for each pre-ladder. A pre-ladder R is *type* $t(\ell, h, p, q)$ if R satisfies the following conditions:

(a) R contains $\ell \geq 2$ lines;

(b) R contains $h \geq p+q$ half-bars (Each bar is counted as two half-bars);

(c) p unmatched l-bars are attached to the (ℓ-1)-th line; and

(d) q unmatched l-bars are attached to the ℓ-th line.

For example, the pre-ladder $P(L)$ in Fig. 5 is type $t(6, 25, 1, 0)$. Note that any ladder lottery with n lines and b bars is type $t(n, 2b, 0, 0)$. We denote by $T(\ell, h, p, q)$ the set of pre-ladders of type $t(\ell, h, p, q)$. We give a useful recurrence for $|T(\ell, h, p, q)|$. We have the following four cases.

Case 1: $h < p+q$ or $\ell < 2$.

$|T(\ell, h, p, q)| = 0$ holds, since $h \geq p+q$ and $\ell \geq 2$ hold for any pre-ladder.

Case 2: $\ell = 2$, $q = 0$, and $h = p$

Clearly such pre-ladder is unique, so $|T(\ell, h, p, q)| = 1$ holds.

Case 3: $h \geq p+q$ and $q = 0$.

Let R be a pre-ladder of type $t(\ell, h, p, q)$. The second last bit of $C(R)$ is always $\mathtt{0}$. (Otherwise, ℓ-th line has an l-bar, a contradiction.) The second last bit $\mathtt{0}$ in $C(R)$ represents either an r-bar of ℓ-th line or the end-of-line of (ℓ-1)-th line. For the former case $P(R)$ is type $t(\ell, h-1, p+1, 0)$. For the latter case $P(R)$ is type $t(\ell-1, h, 0, p)$. For any distinct R_1 and R_2 of $t(\ell, h, p, q)$ with $h \geq p+q$ and $q = 0$, $P(R_1)$ and $P(R_2)$ are distinct. Thus $|T(\ell, h, p, 0)| = |T(\ell, h-1, p+1, 0)| + |T(\ell-1, h, 0, p)|$ holds.

Case 4: $h \geq p+q$ and $q > 0$.

Let R be a pre-ladder of type $t(\ell, h, p, q)$. The second last bit in $C(R)$ is either $\mathtt{0}$ or $\mathtt{1}$. If the second last bit of $C(R)$ is $\mathtt{0}$, then it represents an r-bar attached to ℓ-th line. Thus, $P(R)$ is type $t(\ell, h-1, p+1, q)$. Otherwise, the second last bit of $C(R)$ is 1, then it represents an l-bar attached to ℓ-th line. Hence, $P(R)$ is type $t(\ell, h-1, p, q-1)$. Thus $|T(\ell, h, p, q)| = |T(\ell, h-1, p+1, q)| + |T(\ell, h-1, p, q-1)|$ holds.

For example, Fig. 7 shows the recurrence for $|T(3, 8, 0, 0)|$. By the recurrence, we have the following lemma.

Lemma 4.1 For four non-negative integers ℓ, h, p, and q,

$$|T(\ell, h, p, q)| = \begin{cases} 0 & \text{if } h < p+q \text{ or } \ell < 2 \\ 1 & \text{if } \ell = 2,\ h = p \text{ and } q = 0 \\ |T(\ell, h-1, p+1, 0)| + |T(\ell-1, h, 0, p)| & \\ \quad \text{if } h \geq p+q \text{ and } q = 0 & \\ |T(\ell, h-1, p+1, q)| + |T(\ell, h-1, p, q-1)| & \\ \quad \text{if } h \geq p+q \text{ and } q > 0 & \end{cases}$$

Algorithm 1: DP-COUNT(n, b)

for $\ell = 2$ **to** n **do**
  for $h = 0$ **to** $2b$ **do**
    for $p = 0$ **to** h **do**
      for $q = 0$ **to** h **do**
        if $h < p+q$ **then**
          $|T(\ell, h, p, q)| = 0$
        else if $\ell = 2$, $q = 0$, *and* $h = p$ **then**
          $|T(\ell, h, p, q)| = 1$
        else if $q = 0$ **then**
          $|T(\ell, h, p, q)| = |T(\ell, h-1, p+1, 0)| + |T(\ell-1, h, 0, p)|$
        else if $q > 0$ **then**
          $|T(\ell, h, p, q)| = |T(\ell, h-1, p+1, q)| + |T(\ell, h-1, p, q-1)|$

Based on the recurrence above, **Algorithm 1** computes the number of ladder lotteries with n lines and b bars. **Algorithm 1** is a dynamic programming algorithm on the table of types. The number of entries is nb^3, and each entry is calculated in constant time, so the total running time is $\mathrm{O}(nb^3)$. As a byproduct the number of ladder lotteries with every $n' \leq n$ lines and every $b' \leq b$ bars are also computed.

Theorem 4.2 The number of ladder lotteries with every $n' \leq n$ lines and every $b' \leq b$ bars can be calculated in $\mathrm{O}(nb^3)$ time in total.

5. Random generation

In this section we consider random generation of ladder lotteries. The recurrence in Lemma 4.1 generates a tree structure among the types (see an example in Fig. 7), in which each path from the root to a leaf one-to-one corresponds to some ladder lottery of type $t(n, 2b, 0, 0)$. The choice of i-th generation type decides the meaning of the i-th second last bit of the code. (Here the root belongs the first generation.)

The table generated by **Algorithm 1** tells us the number of leaves in the subtree rooted at each type. We can choose a random path from the root to some leaf, by repeatedly choosing some child of the current type so that each leaf has an equal chance to be reached. Thus we can generate ladder lotteries, uniformly at random.

Our algorithm is shown in **Algorithm 2**. Suppose that we are now at a type $T(\ell, h, p, q)$ in the tree structure, and $T(\ell_1, h_1, p_1, q_1)$ and $T(\ell_2, h_2, p_2, q_2)$ are the two child types of $T(\ell, h, p, q)$. **Algorithm 2** computes a random value, say x, in $[1, |T(\ell, h, p, q)|]$ uniformly at random, chooses $T(\ell_1, h_1, p_1, q_1)$ if $x \leq |T(\ell_1, h_1, p_1, q_1)|$ and $T(\ell_2, h_2, p_2, q_2)$ otherwise, then recursively call with the chosen type. Since **Algorithm 1** computes the table as the preprocessing, these numbers can be looked up

*1 We assume that n and b are coded in unary codes.

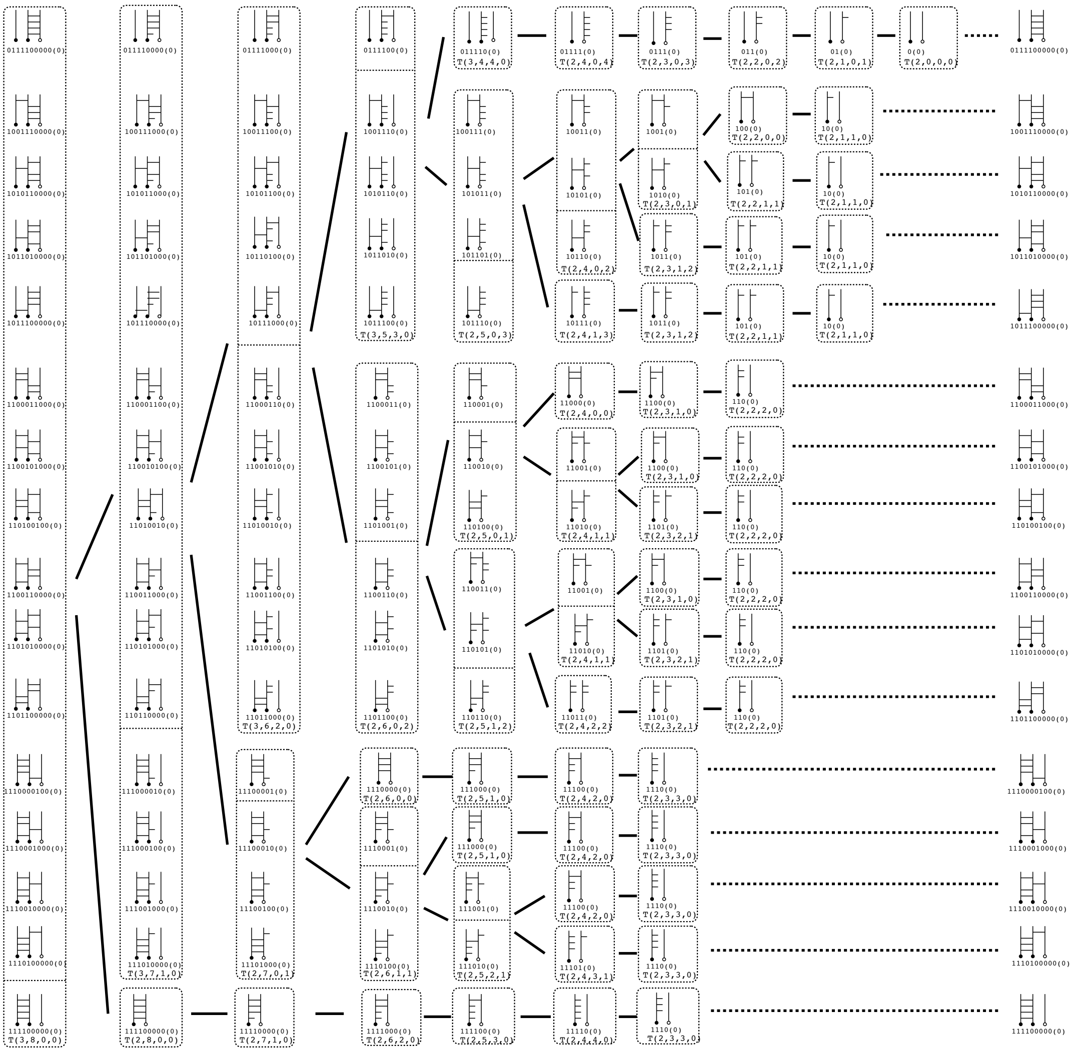

Fig. 7 The recurrence for $|T(3,8,0,0)|$.

Algorithm 2: Random-Generation(ℓ, h, p, q)

begin
  if $\ell = 2$, $h = p$, *and* $q = 0$ **then**
    return *the ladder lottery corresponding to the path from the root to the current leaf.*
  else
    if $T(\ell, h, p, q)$ *has only one child, say* $T(\ell_1, h_1, p_1, q_1)$ **then**
      Random-Generation(ℓ_1, h_1, p_1, q_1)
    /* Let $T(\ell_1, h_1, p_1, q_1)$ and $T(\ell_2, h_2, p_2, q_2)$ be the two child types of $T(\ell, h, p, q)$. */
    Generate an integer x in $[1, |T(\ell, h, p, q)|]$ uniformly at random.
    if $x \leq |T(\ell_1, h_1, p_1, q_1)|$ **then** /* Choose $T(\ell_1, h_1, p_1, q_1)$ */
      Random-Generation(ℓ_1, h_1, p_1, q_1)
    else /* Choose $T(\ell_2, h_2, p_2, q_2)$. */
      Random-Generation(ℓ_2, h_2, p_2, q_2)

in O(1) time. Thus we can generate ladder lotteries uniformly at random, as in the following theorem.

Theorem 5.1 Given two integers $n \geq 2$ and $b \geq 0$, after computing the table of $|T(\ell, h, p, q)|$ by **Algorithm 1**, we can generate ladder lotteries with n lines and b bars in $O(n + b)$ time for each, uniformly at random.

6. Conclusions

We have designed three algorithms for enumeration, counting, and random generation of ladder lotteries with n lines and b bars. All the three algorithms are based on the code [1] of ladder lotteries.

Our enumeration algorithm enumerates all the ladder lotteries with n lines and b bars in $O(n + b)$ time for each. Our counting algorithm counts the number of ladder lotteries with n lines and b bars in $O(nb^3)$ time. Our random generation algorithm takes $O(nb^3)$ time as a preprocessing, then generates ladder lotteries with n lines and b bars in $O(n + b)$ time for each, uniformly at random.

References

[1] T. Aiuchi, K. Yamanaka, T. Hirayama, and Y. Nishitani. Coding ladder lotteries. In *Proc. of European Workshop on Computational Geometry 2013, Braunschweig, Germany*, pages 151–154, March 2013.
[2] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.
[3] D.E. Knuth. *Axioms and hulls.* LNCS 606, Springer-Verlag, 1992.
[4] L. Manivel. *Symmetric Functions, Schubert Polynomials and Degeneracy Loci.* American Mathematical Soc., 2001.
[5] S. Nakano. Efficient generation of triconnected plane triangulations. *Computational Geometry: Theory and Applications*, 27(2):109–122, 2004.
[6] K. Yamanaka and S. Nakano. Listing all plane graphs. *Journal of Graph Algorithms and Applications*, 13(1):5–18, 2009.
[7] K. Yamanaka, S. Nakano, Y. Matsui, R. Uehara, and K. Nakada. Efficient enumeration of all ladder lotteries and its application. *Theoretical Computer Science*, 411:1714–1722, 2010.

Appendix

A.1 Child generation

If we can traverse the family forest $F_{n,b}$, then we can enumerate all ladders lotteries in $\mathcal{S}_{n,b}$ which correspond to leaves of $F_{n,b}$. To traverse the family forest, we consider to enumerate all root pre-ladders of $F_{n,b}$, and to enumerate all child pre-ladders of a pre-ladder in $F_{n,b}$. First, by enumerating pre-ladders with two lines and p unmatched l-bars attached to the first line for each $p = 1, 2, \ldots, b$, we obtain all root pre-ladders of $F_{n,b}$. We next consider to enumerate all child pre-ladders of any pre-ladder in $F_{n,b}$. Now imagine to generate the code corresponding to a child pre-ladder from the code corresponding to the parent pre-ladder. The code of a child pre-ladder is obtained by appending `0` or `1` to the code corresponding to the parent pre-ladder as the second last bit. That is, a child pre-ladder is obtained by appending a half-bar or a line to the parent pre-ladder. Now we explain the details.

Let R be a pre-ladder in $F_{n,b}$, and let $C(R)$ be the code of R. Recall that R is a pre-ladder derived from some ladder lottery with n lines and b bars. We introduce a notation, as follows. $R(0)$ is the pre-ladder corresponding to the code obtained from $C(R)$ by appending `0` as the second last bit. Similarly, $R(1)$ is the pre-ladder corresponding to the code obtained from $C(R)$ by appending `1` as the second last bit. Note that $R(0)$ is obtained from R by attaching a r-bar or appending a line, and $R(1)$ is obtained from R by attaching an l-bar. $R(0)$ and $R(1)$ are candidates of child pre-ladders of R. $R(i)$ for each $i = 1, 2$ is a child pre-ladder of R if and only if $R(i)$ is a vertex of $F_{n,b}$.

We assume that R has ℓ lines, h half-bars, p unmatched l-bars attached to the (ℓ-1)-th line, and q unmatched l-bars attached to the ℓ-th line. Recall that a bar is regarded as a pair of an l-bar and an r-bar. If R is a ladder lottery with n lines and b bars, then R has no child. We hence assume that R is not a ladder lottery with n lines and b bars. Note that R is ladder lottery with n lines and b bars if and only if R satisfies $\ell = n$, $h = 2b$, $p = 0$, $q = 0$.

Case 1: $p = 0$

We first assume that $\ell = n$ holds. Since there is no unmatched l-bar attached to the (n-1)-th line in R, $R(0)$ contains $n + 1$ lines. (Note that, in this case, $R(0)$ is obtained from R by appending a new line.) Hence, $R(0)$ is not a child pre-ladder of R. $R(1)$ includes an unmatched l-bar attached to the n-th line. Note that $R(1)$ is obtained from R by appending an unmatched l-bar attached to the n-th line. Hence, $R(1)$ is not a child pre-ladder of R.

We next assume that $\ell = n - 1$ holds. If $h + q < 2b$ holds, then we show that $R(0)$ is not a child pre-ladder of R and $R(1)$ is a child pre-ladder, as follows. $R(0)$ contains n lines and q unmatched l-bar such that $h + q < 2b$ holds. Hence any pre-ladder derived from $R(0)$ cannot contains n lines and $2b$ half-bars, namely, b bars. Hence, $R(0)$ is not a child pre-ladder. On the other hand, $R(1)$ is a child pre-ladder of R. If $h + q = 2b$ holds, then we show that $R(0)$ is a child and $R(1)$ is not a child. $R(0)$ is obtained from R by appending a new line. Hence, $R(0)$ is a child pre-ladder of R. Since $h + q = 2b$ holds, any unmatched l-bar cannot be appended

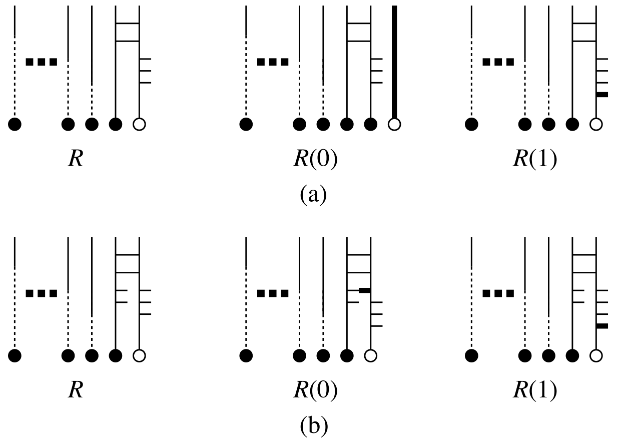

Fig. A·1 Illustration for child generations.

Algorithm 3: GENERATE(n, b)

```
for p = 0 to b do
    Create a root pre-ladder R with two lines and p unmatched
    l-bars attached to the first line.
    FIND-ALL-CHILDREN(R)
```

to derive a ladder lottery with n lines and b bars. Hence, $R(1)$ is not a child pre-ladder.

We finally assume that $\ell < n-1$ holds. If $h+q < 2b$ holds, then both $R(0)$ and $R(1)$ are child pre-ladders, as illustrated in Fig. A·1(a). If $h+q = 2b$ holds, then only $R(0)$ is a child pre-ladder.

Case 2: $p > 0$

In this case, we can assume that $h < 2b$ holds. (If $h = 2b$ holds, we cannot append any r-bar such that it matches to an l-bar attached to the (ℓ-1)-th line.) If $h+p+q < 2b$ holds, then both $R(0)$ and $R(1)$, as illustrated in Fig. A·1(b), are child pre-ladder.

Next, we assume that $h+p+q = 2b$ holds. Then $R(0)$ is a child pre-ladder. On the other hand, $R(1)$ is not a child pre-ladder, since any ladder lottery in $\mathcal{S}_{n,b}$ cannot be derived from $R(1)$.

By the case analysis above, we have the algorithms shown in **Algorithm 3** and **Algorithm 4**. **Algorithm 3** is the main routine. It enumerates all root pre-ladders of $F_{n,b}$, then calls **Algorithm 4** (FIND-ALL-CHILDREN) for each root pre-ladder. **Algorithm 4** recursively enumerates all child pre-ladders of a given pre-ladder, so it traverses the family forest and output a ladder lottery at each leaf. **Algorithm 4** always stores the current pre-ladder in global memory and updates it with some difference information which is needed to reconstruct the previous ladder lotteries. Hence, memory space required in our algorithm is $O(n+b)$. By **Algorithms 3** and **4**, we have the following theorem.

Theorem A.1.1 Our algorithm uses $O(n+b)$ space and enumerates all ladder lotteries with n lines and b bars in $O(n+b)$ time for each.

Algorithm 4: FIND-ALL-CHILDREN(L)

```
Let L be a pre-ladder in F_{n,b}. Assume that L has ℓ lines, h
half-bars, p unmatched l-bars attached to the (ℓ-1)-th line, and q
unmatched l-bars attached to the ℓ-th line.
if ℓ = n and h = 2b then                          /* No child */
    Output R
    return
if p = 0 then
    if ℓ = n − 1 then
        if h + q = 2b then
            FIND-ALL-CHILDREN(R(0))
        else                                      /* h + q < 2b */
            FIND-ALL-CHILDREN(R(1))
    else if ℓ < n − 1 then
        if h + q = 2b then
            FIND-ALL-CHILDREN(R(0))
        else                                      /* h + q < 2b */
            FIND-ALL-CHILDREN(R(0))
            FIND-ALL-CHILDREN(R(1))
else                                              /* p > 0 */
    if h + p + q = 2b then
        FIND-ALL-CHILDREN(R(0))
    else                                          /* h + p + q < 2b */
        FIND-ALL-CHILDREN(R(0))
        FIND-ALL-CHILDREN(R(1))
```