Rotation Distance between Two Binary Trees

Yen-Ju Chen[1], Jou–Ming Chang[2], Yue-Li Wang[1,*]

[1] Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan, ROC

[2] Department of Information Management, National Taipei College of Business, Taipei, Taiwan, ROC

Abstract

There are a number of ways to measure the difference in shape between two rooted binary trees with the same number of leaves. Pallo introduced a left weight sequence, which is a sequence of positive integers, to characterize the structure of a binary tree. By applying the AVL tree transformation on binary trees, we develop an algorithm to transform the left weight sequences between two binary trees efficiently.

Keywords: Binary trees; Rotation distance; AVL trees; Algorithms

1. Introduction

Binary trees are a fundamental data structure in computer science and has been extensively studied over the past forty years [1, 2, 3, 28]. There are a number of ways to measure the difference in shape between two binary trees with the same number of leaves. In particular, the rooted binary trees with a specific ordering on nodes is interesting in tree balancing, such as binary search trees. One of the most common operations to carry out the distance measures on trees is the use of rotations. A *rotation* in a binary tree is a local restructuring that changes the tree into another tree and preserves the inorder (or symmetric order) of the tree [7, 8].

The combinatorial properties of binary trees related to rotations have been the focus of extensive researches. For example, enumerating binary trees with a Gray code with respect to rotations was studied in [13], [14], and [32]. There exists a well-known one-to-one correspondence between a binary tree with n internal nodes and a triangulation of a convex polygon with $n+2$ vertices [30]. Under this bijection, flipping an edge in a triangulation corresponds precisely to a rotation in the corresponding binary tree [30] (see also [10] and [13]). In [19], Pallo introduced the left weight sequences for binary trees and showed that every binary tree can be characterized by a left weight sequence (the formal definition of left weight sequence will be given in Section 2). In that paper, Pallo also proved that rotations on binary trees with n internal nodes induce a combinatorial structure which is the so-called nth Tamari lattice, i.e., an n-element partially ordered set with a unique maximum and minimum. As a result, coding binary trees by left weight sequences is a suitable tool for the study of Tamari lattice [20, 21, 23].

The *rotation distance* $d(T, T')$ between two binary trees T and T' with the same number of leaves is the minimum number of rotations needed to transform T into T'. However, it still remains an open problem whether the rotation distance between any two binary trees can be computed in polynomial time [29]. Pallo [22] and Rogers [27] gave approximation algorithms to estimate rotation distance of two trees. On the other hand, Culik and Wood [7] showed an upper bound to $2n-2$ on rotation distance between any pair of binary trees with n internal nodes. This bound was improved to $2n-6$ for $n \geq 11$ [17, 18, 30]. Some properties of rotations in binary trees were studied by Hanke, Ottmann and Schuierer [9], Hurtado and Noy [10], Hurtado, Noy and Urrutia [11], and Rogers and Dutton [25, 26]. These authors obtained their results using the equivalence between rotations on binary trees and the diagonal-flipping in triangulations of a convex polygon.

Many researchers have focused on the study of various restrictions on rotations. In [4], Bonnin and Pallo introduced a special case of rotations on binary trees where rotations are restricted at nodes whose parent has a leaf as left subtree. They showed that the corresponding distance between two binary trees can be computed in quadratic time. Sundar [31] studied transformations of binary trees when only a single direction of rotation called right rotation is permitted. Recently, Cleary [5] introduced another restricted rotation on binary trees where rotations are allowed at one of the two nodes, namely the root or the right child of the root. Using the metric properties on Thompson's group, Cleary obtained a linear upper bound and lower bound for this restricted rotation distance in terms of n, the

*All correspondence should be addressed to Professor Yue-Li Wang, Department of Information Management, National Taiwan University of Science and Technology, No. 43, Section 4, Kee-Lung Road, Taipei, Taiwan, Republic of China. (Phone: 886–2–27376768, Fax: 886–2–27376777, Email: ylwang@cs.ntust.edu.tw).

number of internal nodes in the trees. Furthermore, he also provided an efficient algorithm for computing the restricted rotation distance between two binary trees. Lately, a significantly improved upper bound and lower bound for this type of restricted rotation distance can be found in [6, 15]. Pallo [24] generalized this case to the situation where rotations are permitted only at nodes along the right arm of the tree (i.e., the path from the root to its rightmost leaf). In this case, an efficient algorithm to compute this right-arm rotation distance was established. Lucas [16] recently presented a polynomial time algorithm for finding the exact rotation distance between any two binary trees that are of a restricted form (each node has at most one child).

In this paper, we propose an algorithm that takes the left weight sequences as input, but uses the full operations provided in the AVL trees to construct a sequence of rotations for estimating the rotation distance. Consequently, our algorithm can be run in O(Δn) time, where Δ denotes the sum of weight differences between two binary trees T and T' with n internal nodes. The proposed algorithm is more efficient than Pallo's since Δ is at most n^2 and Pallo's algorithm requires O(n^4) time [20].

2. Left weight sequences

A binary tree T considered here is a rooted, ordered tree with n internal nodes $v_1, v_2, \ldots, v_n$ where the indices are numbered by the inorder traversal of T and such that each internal node is restricted to having two children, a left child and a right child. The tree is sometimes called an *extended binary tree* [12], and we will refer it as a binary tree for convenience. The *weight* of T, denoted by $w(T)$, is the number of leaves (external nodes) in the tree. For each internal node $v_i \in T$, the subtree rooted at v_i is denoted by T_i. The *left subtree* (respectively, *right subtree*) of v_i is the subtree rooted at the left child (respectively, right child) of v_i and is denoted by L_i (respectively, R_i). The weight $w_T(i) = w(L_i)$ is called the *left weight* of v_i in T and the integer sequence $w_T = (w_T(i))_{i=1}^n = (w_T(1), w_T(2), \ldots, w_T(n))$ is called the *left weight sequence* (LW-sequence for short) of T [19]. Note that the subtree L_i contains $w_T(i) - 1$ internal nodes. For example, Figure 1 shows a tree T containing six internal nodes whose left weights are 1,2,1,2,1, and 6 respectively. Thus the LW-sequence of T is $(1, 2, 1, 2, 1, 6)$.

In [19], Pallo characterized an integer sequence to be an LW-sequence of a binary tree and gave the following theorem.

Theorem 1 *An integer sequence* $(w_1, w_2, \ldots, w_n)$ *is the LW-sequence of a binary tree* T *with* n *internal nodes* $v_1, v_2, \ldots, v_n$ *if and only if for all* $i \in \{1, \ldots, n\}$ *the following conditions are satisfied:*
(1) $1 \le w_i = w_T(i) \le i$, *and*
(2) $i - w_i \le j - w_j$ *for all* $j \in [i - w_i + 1, i]$.

In the following, we will show that every binary tree T has an interval representation corresponding

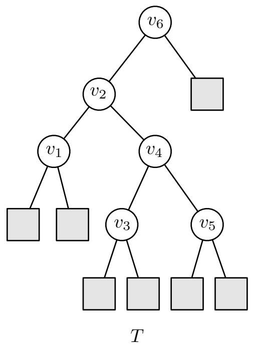
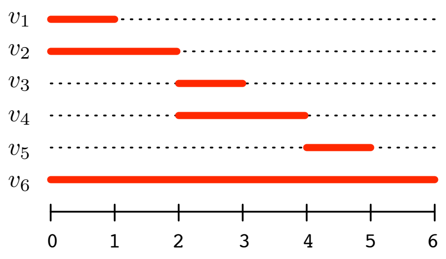


Figure 1: The LW-sequence of T is $(1, 2, 1, 2, 1, 6)$.

to the LW-sequence of T. This representation contains a set of intervals such that for each internal node $v_i \in T$, there is a corresponding interval with the starting point $S_T(i)$ and the end point $D_T(i)$ on a line which are defined as follows: $S_T(i) = i - w_T(i)$ and $D_T(i) = i$. For example, Figure 2(a) shows the starting points and the end points of intervals corresponding to the LW-sequence $(1, 2, 1, 2, 1, 6)$. Also we draw the interval representation in Figure 2(b).

v_i	$w_T(i)$	$S_T(i)$	$D_T(i)$
v_1	1	0	1
v_2	2	0	2
v_3	1	2	3
v_4	2	2	4
v_5	1	4	5
v_6	6	0	6

(a)



(b)

Figure 2: (a) The starting points and the end points of intervals; (b) The interval representation.

Throughout the rest, we write $pre_T(k) = i$ (or $pre_T^{-1}(i) = k$) to mean that the position of node v_i in the preorder traversal of T is k. For instance, using the binary tree T in Figure 1 as an example, we can easily set up $(pre_T(i))_{i=1}^6 = (6, 2, 1, 4, 3, 5)$ and $(pre_T^{-1}(i))_{i=1}^6 = (3, 2, 5, 4, 6, 1)$.

Given the LW-sequence of a binary tree T, we now show that the preorder of T can be determined by using the above interval representation.

To achieve the preorder, we first sort $v_i \in T$ according to their $S_T(i)$ in increasing order. If more than one interval have the same starting point, a node with a larger index in inorder traversal is smaller than a node with a smaller index. Consequently, the sorted sequence of nodes, called the *α-sequence* of T, is the resulting preorder of T. We summarize the rules as follows:

(R1) if $S_T(j) < S_T(i)$ then $pre_T^{-1}(j) < pre_T^{-1}(i)$

(R2) if $S_T(j) = S_T(i)$ and $i < j$ then $pre_T^{-1}(j) < pre_T^{-1}(i)$

For example, consider the binary tree T in Figure 1 again. Scanning the starting points of intervals from left to right and then from bottom to up in the representation of Figure 2(b), gives the preorder $v_6, v_2, v_1, v_4, v_3, v_5$ of T.

Theorem 2 *Given the LW-sequence of a binary tree T with n internal nodes, the preorder of T can be determined in $O(n)$ time.*

Proof. Obviously, the starting points of intervals can be computed in $O(n)$ time and the α-sequence of T can also be obtained in $O(n)$ time by using the counting sort. To show the α-sequence of T is the preorder of T, we need to show that, from rules (R1) and (R2), if $pre_T^{-1}(i) < pre_T^{-1}(j)$ then either $S_T(i) < S_T(j)$ or $S_T(i) = S_T(j)$ and $j < i$. Consider two internal nodes $v_i, v_j \in T$ and without loss of generality assume that $pre_T^{-1}(i) < pre_T^{-1}(j)$. There are three possibilities for the positions of v_i and v_j as follows.

Case 1: Node v_j is located in the left subtree of v_i (i.e., $v_j \in L_i$). In this case, $j \in [i - w_T(i) + 1, i - 1]$. By Theorem 1, $i - w_T(i) \leq j - w_T(j)$. Thus $S_T(i) \leq S_T(j)$. In particular, $j < i$ holds whenever $S_T(i) = S_T(j)$.

Case 2: Node v_j is located in the right subtree of v_i (i.e., $v_j \in R_i$). Let $v_{j'} \in T_j$ be the node that has the smallest index in inorder traversal of T. Then $j' = j - w_T(j) + 1$. Clearly, $v_{j'} \in R_i$ and thus $i < j'$. This implies that $i - w_T(i) < i \leq j' - 1 = j - w_T(j)$. Thus $S_T(i) < S_T(j)$.

Case 3: There is a common ancestor of v_i and v_j, say v_k, in T such that $v_i \in L_k$ and $v_j \in R_k$. Choose $v_{i'} \in T_i$ and $v_{j'} \in T_j$ with the smallest index in inorder traversal of T, respectively. Then $i' = i - w_T(i) + 1$ and $j' = j - w_T(j) + 1$. Since $v_{i'} \in L_k$ and $v_{j'} \in R_k$, $i' < k < j'$. This implies that $i - w_T(i) < j - w_T(j)$. Thus $S_T(i) < S_T(j)$. □

3. Types of rotations

Rotations are commonly used as primitive steps in tree balancing such as AVL-trees and splay trees [12]. A rotation can be performed at any internal node of a binary tree, and it is classified by two kinds of operations: the left rotation and the right rotation. A *left rotation*, applied to a node y which is the right child of another node x, is an operation that raises y to the place of x such that x becomes the new left child of y and the left subtree of y becomes the new right subtree of x, while the rest trees remain unchanged. A *right rotation* is symmetric to the left rotation, i.e., if we apply a right rotation to the former node x after the left rotation, it reconstructs the original tree.

In this paper, we use more general rotations: LL-, RR-, LR- and RL-rotations, which are required for balancing AVL trees, to transform the LW-sequences between two binary trees. An *AVL tree*, introduced by Adelson-Velskii and Landis [1], is a height-balanced binary search tree in which the height difference between the two subtrees of each node is no more than 1. To carry out the tree being balanced, four types of rotations at a node v_i are shown in Figure 4.

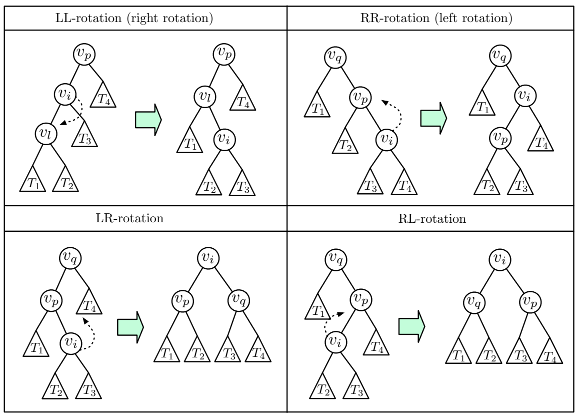


Figure 3: The LL-rotation, RR-rotation, LR-rotation, and RL-rotation at a node v_i.

The LL-rotation is similar to the right rotation and the RR-rotation is similar to the left rotation. The LR-rotation is a combination of two rotations: we perform an RR-rotation at the node v_i followed by an LL-rotation of the new tree at v_q. The RL-rotation is the mirror image of the LR-rotation, i.e., we perform an LL-rotation at the node v_p followed by an RR-rotation of the new tree at v_i. The following lemma describes the change of left weights in each of the four types of rotations. We omit the proof since its validity can easily be checked from Figure 4.

Lemma 3 *Let T be a binary tree and $v_i \in T$ be an internal node. If we perform a rotation $\Re$ at v_i to transform T into T', then the left weight $w_{T'}(i)$ can be determined by the following cases:*

(1) *If $\Re$ is an LL-rotation, then $w_{T'}(i) = w_T(i) - w_T(l)$ where v_l is the left child of v_i in T.*

(2) *If $\Re$ is an RR-rotation, then $w_{T'}(i) = w_T(i) + w_T(p)$ where v_i is the right child of v_p in T.*

(3) *If $\Re$ is an LR-rotation, then $w_{T'}(i) = w_T(i) + w_T(p)$ and $w_{T'}(q) = w_T(q) - w_{T'}(i)$, where v_i is the right child of v_p and v_p is the left child of v_q in T.*

(4) *If $\Re$ is an RL-rotation, then $w_{T'}(i) = w_T(i) + w_T(q)$ and $w_{T'}(p) = w_T(p) - w_T(i)$, where v_i is the left child of v_p and v_p is the right child of v_q in T.*

From Lemma 3, we can see that performing a rotation $\Re$ at a node $v_i \in T$ will increase $w_T(i)$ if $\Re$ is an RR-, LR-, or RL-rotation. Contrastively, $w_T(i)$ will be decreased if $\Re$ is an LL-rotation. For convenience to refer v_l, v_p, and v_q in Lemma 3, we use the following auxiliary lemmas.

Lemma 4 *Let T be a binary tree and $v_i, v_l \in T$ be internal nodes. If v_l is the left child of v_i, then $l = pre_T(pre_T^{-1}(i) + 1)$.*

Lemma 5 *Let T be a binary tree and $v_i, v_p \in T$ be internal nodes. If v_i is the right child of v_p, then $p = i - w_T(i)$.*

Lemma 6 *Let T be a binary tree and $v_i, v_p, v_q \in T$ be internal nodes. If v_i is the right child of v_p and v_p is the left child of v_q, then $p = i - w_T(i)$ and $q = pre_T(pre_T^{-1}(p) - 1)$.*

Lemma 7 *Let T be a binary tree and $v_i, v_p, v_q \in T$ be internal nodes. If v_i is the left child of v_p and v_p is the right child of v_q, then $p = pre_T(pre_T^{-1}(i) - 1)$ and $q = i - w_T(i)$.*

4. The algorithm

In this section, we present an algorithm to transform the LW-sequences between two binary trees T and T'. For convenience, we call T the source tree and T' the destination tree. Note that T and T' have the same inorder. Define $\delta(i) = w_{T'}(i) - w_T(i)$ as the *weight difference* of node v_i in the trees. Our strategy adjusts $\delta(i)$ to become zero by using the mentioned rotations. Thus, if the left weight of a node v_i is changed by a rotation, then we need to revise $\delta(i)$. According to Lemma 3 and the auxiliary lemmas (from Lemma 4 to Lemma 7), each of the four types of rotations can be implemented as follows (where the argument i of each function indicates that the rotation is performed at node v_i in the current tree):

```
Function LL-rotation($i$)
Step 1. By Lemma 4, $v_l$ is the left child of $v_i$ and
          compute $l = pre_T(pre_T^{-1}(i) + 1)$.
Step 2. According to Statement (1) of Lemma 3,
          update left weight $w_T(i) = w_T(i) - w_T(l)$.
Step 3. Update weight difference
          $\delta(i) = w_{T'}(i) - w_T(i)$.
end LL-rotation
```

```
Function RR-rotation($i$)
Step 1. By Lemma 5, $v_i$ is the right child of $v_p$
          and compute $p = i - w_T(i)$.
Step 2. According to Statement (2) of Lemma 3,
          update left weight $w_T(i) = w_T(i) + w_T(p)$.
Step 3. Update weight difference
          $\delta(i) = w_{T'}(i) - w_T(i)$.
end RR-rotation
```

```
Function LR-rotation($i$)
Step 1. By Lemma 6, $v_i$ is the right child of $v_p$ and
          $v_p$ is the left child of $v_q$ and compute
          $p = i - w_T(i)$ and $q = pre_T(pre_T^{-1}(p) - 1)$.
Step 2. According to Statement (3) of Lemma 3,
          update left weights $w_T(i) = w_T(i) + w_T(p)$
          and $w_T(q) = w_T(q) - w_T(i)$.
Step 3. Update weight differences
          $\delta(i) = w_{T'}(i) - w_T(i)$ and
          $\delta(q) = w_{T'}(q) - w_T(q)$.
end LR-rotation
```

```
Function RL-rotation($i$)
Step 1. By Lemma 7, $v_i$ is the left child of $v_p$ and
          $v_p$ is the right child of $v_q$ and compute
          $p = pre_T(pre_T^{-1}(i) - 1)$ and $q = i - w_T(i)$.
Step 2. According to Statement (4) of Lemma 3,
          update left weights $w_T(p) = w_T(p) - w_T(i)$
          and $w_T(i) = w_T(i) + w_T(q)$.
Step 3. Update weight differences
          $\delta(i) = w_{T'}(i) - w_T(i)$ and
          $\delta(p) = w_{T'}(p) - w_T(p)$.
end RL-rotation
```

We now list our algorithm to transform two binary trees as follows.

```
Algorithm A
Input: LW-sequences $w_T$ and $w_{T'}$ of source tree $T$
          and destination tree $T'$, respectively.
Output: $dist(T, T')$, an estimated rotation
          distance between $T$ and $T'$.
Step 1.   Let $dist(T, T') = 0$, and compute $(\delta(i))_{i=1}^n$,
          $(pre_T(i))_{i=1}^n$, and $(pre_T^{-1}(i))_{i=1}^n$.
Step 2.   while $\delta(i) \neq 0$ for some $i \in \{1, \ldots, n\}$ do
Step 2.1.   if there exist $v_i, v_p$ and $v_q$ in the current
                tree such that $v_i$ is the right child of
                $v_p$ and $v_p$ is the left child of $v_q$ for
                some $i \in \{1, \ldots, n\}$ and $\delta(i) > 0 > \delta(q)$
            then do LR-rotation($i$);
Step 2.2.   else if there exist $v_i, v_p$ and $v_q$ in the
                current tree such that $v_i$ is the left
                child of $v_p$ and $v_p$ is the right child
                of $v_q$ for some $i \in \{1, \ldots, n\}$ and
                $\delta(i) > 0 > \delta(p)$
            then do RL-rotation($i$);
Step 2.3.   else if there exist $v_i$ and $v_p$ in the current
                tree such that $v_i$ is the right child of $v_p$
                for some $i \in \{1, \ldots, n\}$ and $\delta(i) > 0$
            then do RR-rotation($i$);
Step 2.4.   else if there exist $v_i$ and $v_l$ in the current
```

```
          tree such that $v_l$ is the left child of
          $v_i$ for some $i \in \{1,\dots,n\}$ and $\delta(i) < 0$
        then do LL-rotation($i$);
        end if
Step 2.5. $dist(T,T') = dist(T,T') + 1$ and
        recompute $(pre_T(i))_{i=1}^n$ and
        $(pre_T^{-1}(i))_{i=1}^n$.
      end while
Step 3. Output $dist(T,T')$.
```

Theorem 8 *Given the LW-sequences of two binary trees T and T' with the same number of internal nodes, say n, Algorithm A produces a sequence of rotations to convert T into T' correctly in $O(\Delta n)$ time, where $\Delta = \sum_{i=1}^{n} |w_{T'}(i) - w_T(i)|$.*

Proof. The correctness of Algorithm A follows immediately from Lemma 3 and its auxiliary lemmas (from Lemma 4 to Lemma 7).

In what follows, we analyze the time complexity of Algorithm A. Initially, $(\delta(i))_{i=1}^n$ can be set up in O(n) time. By Theorem 2, $(pre_T(i))_{i=1}^n$ and $(pre_T^{-1}(i))_{i=1}^n$ are attainable by using a counting sort. Thus each of Step 1 and Step 2.5 takes O(n) time. We now show that testing the condition of Step 2 in the main loop only needs a constant time. We use two variables namely *upper* and *lower* which are defined as $upper = \max_{i=1}^n \delta(i)$ and $lower = \min_{i=1}^n \delta(i)$ for testing. Since the contents of two variables are updated in each round of the loop, we only need to compare the difference $upper - lower$ for checking the conditions of Step 2. If the comparing result is equal to zero, it means that $\delta(i) = 0$ for all $i = 1,\dots,n$, and we output the estimated distance between T and T'. Otherwise, it occurs that $\delta(i) \neq 0$ for some $i \in \{1,\dots,n\}$. Let $\Delta = \sum_{i=1}^{n} |\delta(i)|$. Obviously, $\Delta \leq$ O(n^2). Since there is at least one of the weight difference $\delta(i)$ that is revised to approach zero in each round of the main loop in Step 2, the loop is repeated at most Δ times. We now look into the detail of the main loop. Obviously, one of LR-, RL-, RR-, or LL-rotation must be performed in the current tree for each round of the loop. Note that, the LR-rotation and RL-rotation have the higher priority for testing and accomplishing in our implementation. Each of the testing from Step 2.1 to 2.4 requires O(n) time, and each rotation can be accomplished in a constant time. Also, we can see easily that Step 2.1 to 2.4 are exclusive with each other in each round of the loop. Therefore, the total complexity of Algorithm A is bounded in O(Δn) time. □

5. Conclusion

In this paper, we propose an efficient algorithm that takes O(Δn) time to construct a sequence of rotations for estimating rotation distance, where Δ denotes the sum of weight differences between two binary trees T and T' with n internal nodes. Since Δ can be bounded in O(n^2), the time complexity of our algorithm is therefore O(n^3). By using an amortized analysis to evaluate the time complexity of Algorithm A, then we believe that Δ can be diminished and will be less than O(n^2).

In [20], Pallo introduced the lattice of binary trees to study the rotation distance between binary trees. In this paper we provide two extra rotations LR-rotation and RL-rotation used on tree transformations. An interesting question is what the effect on the lattice of binary trees will be if the two additional rotations are included. For instance, Figure 6 shows the lattice of binary trees with n=3 internal nodes, where an LL-rotation or RR-rotation is represented by a directed solid arcs, and an LR-rotation or RL-rotation is represented by a directed dashed arcs.

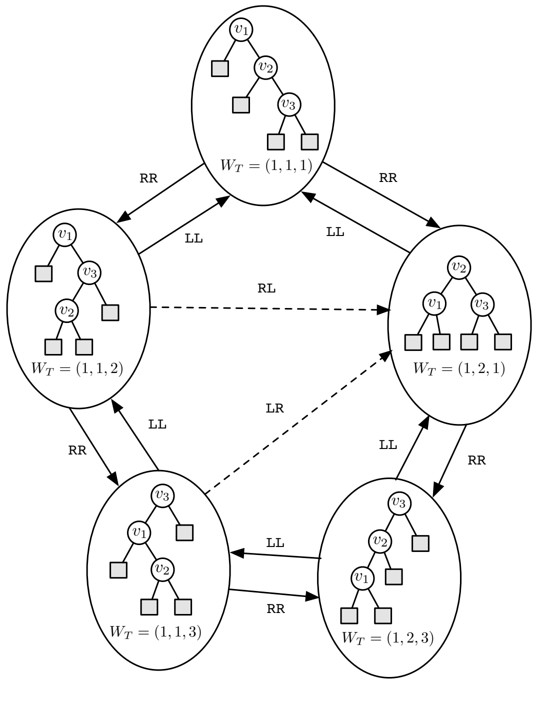


Figure 4: The lattice of binary trees with n=3 internal nodes.

Recall that the number of binary trees with n nodes is the well-known Catalan numbers, denoted by $Bn = \frac{1}{n+1}\binom{2n}{n}$. Let $R(n)$ denote the directed graph induced by the lattice of binary trees with n internal nodes. Now we can compute the *average rotation distance* of $R(n)$, which is defined as follows:

$$\mu(n) = \frac{\sum_{T,T' \in R_n; T \neq T'} dist(T,T')}{2\binom{B_n}{2}}$$

$$= \frac{\sum_{T,T' \in R_n; T \neq T'} dist(T,T')}{B_n(B_n - 1)}$$

where $dist(T, T')$ denotes the distance (i.e., the number of arcs of a shortest path) between two binary trees T and T' in $R(n)$. Note that $dist(T, T') \neq dist(T', T)$. For example $\mu(3) = 1.5$ if the computation of rotation distance only uses the LL-rotation and RR-rotation. Contrastively, if the excess LR-rotation and RL-rotation are allowed to use in the computation of rotation distance, we have the result $\mu(3) = 1.4$. Thus, the study of the variation of $\mu(n)$ for these two cases is interesting, especially for the larger value of n.

References

[1] G. M. Adelson-Velsky and E, M. Landis, An algorithm for organization of information, *Soviet Mathematics Doklady* 3 (1962) 1259–1263.

[2] A. Andersson, General balanced trees, *Journal of Algorithms* 30 (1999) 1–18.

[3] R. Bayer, Symmetric binary B-trees: data structure and maintenance algorithms, *Acta Informatica* 1 (1972) 290–306.

[4] A. Bonnin and J. Pallo, A shortest path metric on unlabeled binary trees, *Pattern Recognition Letters* 13 (1992) 411–415.

[5] S. Cleary, Restricted rotation distance between binary trees, *Information Processing Letters* 84 (2002) 333–338.

[6] S. Cleary and J. Taback, Bounding restricted rotation distance, *Information Processing Letters* 88 (2003) 251–256.

[7] K. Culik and D. Wood, A note on some tree similarity measures, *Information Processing Letters* 15 (1982) 39–42.

[8] C. Germain and J. Pallo, The number of coverings in four Catalan lattices, *International Journal of Computer Mathematics* 61 (1996) 19–28.

[9] S. Hanke, T. Ottmann and S. Schuierer, The edge-flipping distance of triangulations, *Journal of Universal Computer Science* 2 (1996) 570–579.

[10] F. Hurtado and M. Noy, Graph of triangulations of a convex polygon and tree of triangulations, *Computational Geometry* 13 (1999) 179–188.

[11] F. Hurtado, M. Noy and J. Urrutia, Flipping edges in triangulations, *Discrete Computational Geometry* 22 (1999) 333–346.

[12] D. E. Knuth, Sorting and Searching, in: *The Art of Computer Programming*, Vol. 3, Addison-Wesley, Reading, MA, 1973.

[13] J. M. Lucas, The rotation graph of binary trees is hamiltonian, *Journal of Algorithms* 8 (1987) 503–535.

[14] J. M. Lucas, D. Roelants van Baronaigien, and F. Ruskey, On rotations and the generation of binary trees, *Journal of Algorithms* 15 (1993) 343–366.

[15] J. M. Lucas, A direct algorithm for restricted rotation distance, *Information Processing Letters* 90 (2004) 129–134.

[16] J. M. Lucas, Untangling binary trees via rotations, *The Computer Journal* 47 (2004) 259–269.

[17] F. Luccio and L. Pagli, On the upper bound on the rotation distance of binary trees, *Information Processing Letters* 31 (1989) 57–60.

[18] E. Mäkinen, On the rotation distance of binary trees, *Information Processing Letters* 26 (1987/88) 271–272.

[19] J. Pallo, Enumerating, ranking and unranking binary trees, *The Computer Journal* 29 (1986) 171–175.

[20] J. Pallo, On the rotation distance in the lattice of binary trees, *Information Processing Letters* 25 (1987) 369–373.

[21] J. Pallo, Some properties of the rotation lattice of binary trees, *The Computer Journal* 31 (1988) 564–565.

[22] J. Pallo, An efficient upper bound of the rotation distance of binary trees, *Information Processing Letters* 73 (2000) 87–92.

[23] J. Pallo, Generating binary trees by Glivenko classes on Tamari lattices, *Information Processing Letters* 85 (2003) 235–238.

[24] J. Pallo, Right-arm rotation distance between binary trees, *Information Processing Letters* 87 (2003) 173–177.

[25] R. O. Rogers and R. D. Dutton, Properties of the rotation graph of binary trees, *Congressus Numerantium* 109 (1995) 51–63.

[26] R. O. Rogers and R. D. Dutton, On distance in the rotation graph of binary trees, *Congressus Numerantium* 120 (1996) 103–113.

[27] R. O. Rogers, On finding shortest paths in the rotation graph of binary trees, *Congressus Numerantium* 137 (1999) 77–95.

[28] D. D. Sleator and R. E. Tarjan, Self-adjusting binary search trees, *Journal of the ACM* 32 (1985) 652–686.

[29] D. D. Sleator, R. E. Tarjan, and W. R. Thurston, Rotation distance, in: T.M. Cover, B. Gopinath (Eds.), *Open Problems in Communication and Computation*, Springer, Berlin, 1987, pp. 130–137.

[30] D. D. Sleator, R. E. Tarjan, and W. R. Thurston, Rotation distance, triangulations and hyperbolic geometry, *Journal of the American Mathematical Society* 1 (1988) 647–681.

[31] R. Sundar, On the deque conjecture for the splay algorithm, *Combinatorica* 12 (1992) 95–124.

[32] V. Vajnovszki, On the loopless generation of binary tree sequences, *Information Processing Letters* 68 (1998) 113–117.