Internat. J. Math. & Math. Sci.
VOL. 14 NO. 3 (1991) 545-552

SPLAYING A SEARCH TREE IN PREORDER TAKES LINEAR TIME

R. CHAUDHURI and H. HÖFT
Department of Computer Science
Eastern Michigan University
Ypsilanti, Michigan 48197.

(Received May 31, 1990 and in revised form November 1, 1990)

ABSTRACT. In this paper we prove that if the nodes of an arbitrary n-node binary search tree T are splayed in the preorder sequence of T then the total time is O(n). This is a special case of the splay tree traversal conjecture of Sleator and Tarjan.

KEY WORDS AND PHRASES. Binary search tree, preorder traversal, rotation, splay.

1980 AMS SUBJECT CLASSIFICATION CODE. 68P05

1. INTRODUCTION

A form of self-adjusting binary search tree, called the splay tree, has been studied by Sleator and Tarjan in [1]. On any sufficiently long access sequence, splay trees have been shown to be as efficient, to within a constant factor, as both dynamically balanced and static optimum search trees. It was conjectured that the splay trees are as efficient as any form of dynamically updated search trees. In particular, the following conjecture appeared in [1] as a special case of the dynamic optimality conjecture regarding splay trees:

TRAVERSAL CONJECTURE (Sleator and Tarjan)
Let T_1 and T_2 be any two n-node binary search trees containing exactly the same items. Suppose that we access the items in T_1 one after another using splaying, accessing them in the order they appear in T_2 in preorder (the item in the root of T_2 first, followed by the items in the left subtree of T_2 in preorder , followed by the items in the right subtree of T_2 in preorder). Then the total access time is O(n).

A special case of this traversal conjecture was proved by Tarjan in [3]. He showed that the nodes of an arbitrary search tree can be splayed in symmetric order (inorder) in linear time i.e. O(1) per access.

In this paper, we prove the traversal conjecture in the case when $T_1 = T_2$. That is , we prove that the nodes of a binary search tree can be splayed according to its own preorder sequence in time O(n). Note that a binary search tree T_1 can be transformed to another binary search tree T_2 using at most O(n) rotations [5]. Hence given any two binary search trees T_1 and T_2, if we first transform T_1 into T_2 by using rotations and then splay the resulting tree according to its preorder sequence then the total time is still O(n). Thus if we are faced with the task of accessing the nodes of one tree in the

preorder sequence of another tree then this can be done in linear time by the method described above.

2. PRELIMINARIES

A binary search tree is a binary tree whose nodes contain distinct items selected from a totally ordered universe such that for any node x, the items in the left subtree of x are less than x and the items in the right subtree of x are greater than x.

A path in a binary tree is defined as a sequence of nodes $t_1,t_2,....,t_k$ such that t_i is the parent of t_{i+1}, i=1,2,...,k-1. The depth of a node x is the length of the path from the root to x. For any node x in a binary tree, let left(x) and right(x) denote the left and right child of x respectively. A left chain in a binary tree is defined as a path $t_1,t_2,....,t_k$ such that t_{i+1} = left(t_i), i=1,2,...,k-1. A right chain is defined analogously. The left and right arms of a binary tree are the longest left and right chains originating at the root respectively.

A rotation in a binary search tree preserves the symmetric order of items in the tree. Figure 1 illustrates the rotation of the edge joining x and y.

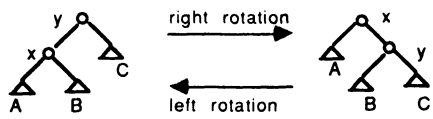

Figure 1. Rotations in a binary tree

A binary search tree in which we splay after each access at the node containing the accessed item is called a splay tree. Splaying is a restructuring operation consisting of a sequence of rotations. To splay a tree at a node x, we walk up the path from x to the root performing rotations along the path. The rotations are performed in pairs, unless the node x is a child of the root. To be specific, we splay at x by repeating one of the following steps until x is at the root of the tree.

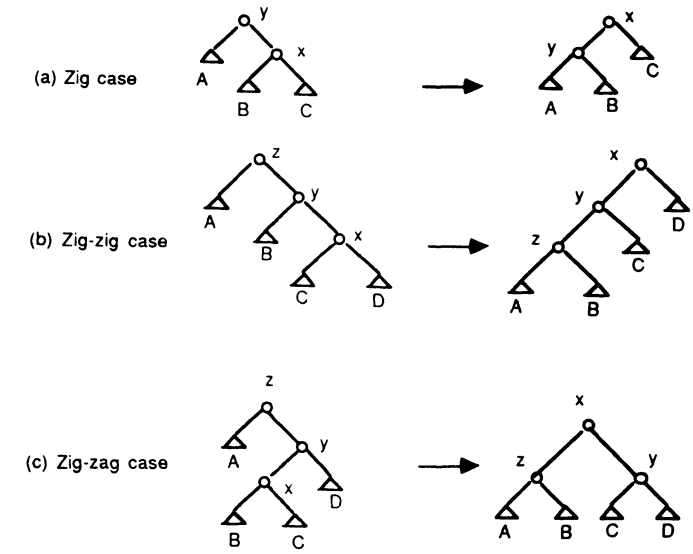

Fig 2. Three possible splayings at x. Each case has a symmetric variant (not shown here).

(a) zig-case: If the parent p(x) of x is the tree root, rotate the edge joining x to p(x). This makes x the root and terminates splaying.

(b) zig-zig case: If p(x) is not the root and x and p(x) are both left children or right children, rotate the edge joining p(x) to its parent and then rotate the edge joining x to p(x).

(c) zig-zag case: If p(x) is not the root and x is a left child and p(x) is a right child or vice-versa, rotate the edge joining x to its parent and then rotate the edge joining x to its new parent.

The splay depth of a node x is the depth of x at the time we start splaying at x. Let T be a n-node binary search tree such that the preorder sequence of its nodes is $t_1,t_2,.....,t_n$. As we splay T according to its own preorder sequence, we generate a sequence of binary search trees $T_1,T_2,.....,T_n$ such that $T_1 = T(t_1) = T$ and $T_i = T(t_i) = SPLAY(T_{i-1},t_i)$ and $T(t_i)$ is the binary search tree with t_i at the root (obtained by splaying the original tree T in the sequence $t_1,t_2,....,t_i$). The notation SPLAY(T,x) denotes splaying x in the tree T.

Let z be a node in a binary search tree T. A node y in T is a right ancestor of z if y is an ancestor of z and $z \leq y$. As in [3], we denote the set of all the right ancestors of z by A(z). Note that $z \in A(z)$. Furthermore, the set of all the right ancestors of z in the subtree rooted at x will be denoted by A(z,x). Clearly, the node x belongs to A(z,x) if and only if z is an element of the left subtree of x.

3. MAIN RESULTS

We are now ready to state and prove our main results.

THEOREM 1.

Let T be a binary search tree whose nodes are being splayed according to its own preorder sequence. Let x be a node of T and assume that the splay depth (SD) of x is d. Then

(a) $SD(left(x)) \leq d/2 + 3/2$ and

(b) $SD(right(x)) \leq 1 + |A(z,x)|$

where z is the preorder predecessor of right(x) in T and $|A(z,x)|$ denotes the cardinality of the set A(z,x).

PROOF:

(a) We note that each splay step consists of two rotations except possibly for the last step when the node being splayed is the child of the root. At each splay step, the distance of x from the root (which is the preorder predecessor of x in T) is reduced by 2 while that of left(x) is reduced by 1 (except possibly at the last zig step where left(x) does not move up if x was the right child of the root before it became the new root). Since the original depth of x was d and that of left(x) was d+1, it is easy to see that

$$SD(left(x)) = d/2 + 1 \text{ , if d is even}$$

and $$SD(left(x)) \leq (d-1)/2 + 2$$

$$= d/2 + 3/2 \text{ , if d is odd.}$$

This completes the proof of part(a).

(b) Assume that x does not belong to the right arm of T. Thus A(x) contains at least one element other than x. We will show that the bound on the right hand side is attained in such a case.

Assume first that x is the left child of its parent p(x). Figure 3 depicts the subtree of T rooted at p(x).

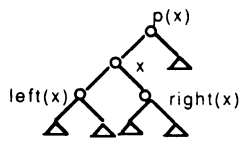

Figure 3. Subtree rooted at p(x) in the original tree T.

Note that x is the preorder successor of p(x) in T. Now consider the search tree T(p(x)) which is obtained by splaying $t_1,t_2,\ldots,p(x)$ successively in the original tree T. It must have the subtree rooted at x on the right arm of its left subtree (figure 4) since in the symmetric order (inorder) traversal of T, p(x) follows immediately after the symmetric order (inorder) node sequence of the subtree rooted at x .

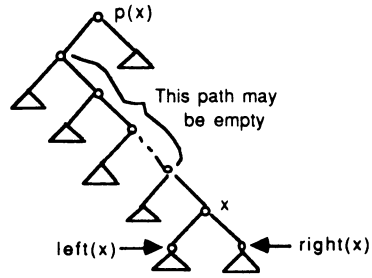

Figure 4. Subtree rooted at x is embedded along the right arm of the left subtree of T(p(x))

Since x is the preorder successor of p(x) in T, it is the node to be splayed next. Note that the splay path of x consists of a number (may be zero) of zig-zigs followed by a zig-zag or a zig at the end depending on whether the splay path is of even or odd length. In either case, the structure of T(x) is depicted by figure 4.

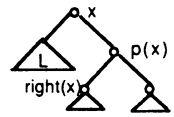

Figure 5. Splay tree T(x) with x at the root

Clearly, the current left subtree L of x now contains the previous left subtree of x rooted at left(x). Let S be the set of all nodes in the subtree rooted at left(x). Each element of S is clearly greater than every element of L-S (figure 5). Hence, as the elements of S are splayed the elements in L-S never appear on the right subtree of the root. Note that the elements of S are to be splayed next in preorder.

Let z be the preorder predecessor of right(x). The path from x to z is the right most path in L. All the right ancestors of z in the tree rooted at x lie along this path. Let A(z,x) = $\{z,m_1,m_2,\ldots,m_k,x\}$, where $z<m_1<m_2<\ldots<m_k<x$. As the nodes of S are splayed in preorder, $m_k,m_{k-1},\ldots,m_1,z$ will appear successively on the right arm of the tree and will form a chain. Furthermore, when z gets to the root the tree will look like as it appears in figure 6.

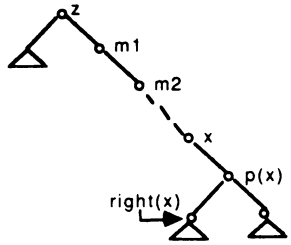

Figure 6. Splay tree with z (preorder predecessor of right(x) at the root

Since right(x) is the preorder successor of z in T, it is clear that the splay depth of right(x) is given by SD(x) = 1 + |A(z,x)| . Note that the splay path of right(x) will be a zig-zag followed by a series of zig-zigs and possibly a zig at the end if the splay path is of odd length. In case the original left subtree of x rooted at left(x) is empty, SD(right(x))=2 , since |A(z,x)| = 1.

In the case where x is the right child of its parent, consider the path from x to the root of T. Let n_k be the very first node where the path turns right. Let h be the parent of n_k. Then $n_k < n_{k-1} < .....< n_1 < x$ and h is greater than all the elements in the subtree rooted at n_k (figure 7).

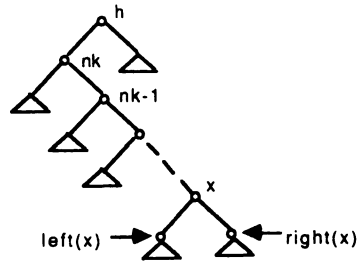

Figure 7. Subtree rooted at h in the original tree T.

Arguing as before, we see that the subtree rooted at n_k will be positioned on the right arm of the left subtree of T(h). It is not difficult to see that our argument in the previous case applies successively to each of the nodes n_{k-1}, n_{k-2},... , n_1, x and finally to right(x).

Next consider the case where |A(x)|=1. The node x must be situated on the right arm of original tree T and hence x is the right child of its parent p(x). Also, when x is at the root its right subtree is the same as the subtree rooted at right(x).Clearly, when it is time for the right(x) to be splayed the splay path consists of the right chain with z, m_1,m_2, m_3,.....,m_k,x and right(x) along the path.Therefore, SD(right(x)) = | A(z,x) | in this case.

THEOREM 2.

The time to splay an n-node binary search tree T according to its own preorder sequence is at most 8n.

PROOF: Clearly the time to splay each node on the left arm of T is 1. Let

$$X = \{x_1, x_2, \ldots\ldots, x_s\}$$

be the nodes of T such that each is a right child of its parent. Let d_i be the splay depth of x_i, i=1,2,.....,s . Using theorem 1(b), we have

$$d_i \leq 1 + |A(z_i, p(x_i))|$$

where z_i is the preorder predecessor of x_i in T. Since the sets $A(z_i, p(x_i))$ are pairwise disjoint, it follows that

$$\sum d_i \leq s + n \leq 2n.$$

Let the subtrees rooted at $x_1, x_2, \ldots\ldots, x_s$ have $k_1, k_2, \ldots\ldots, k_s$ elements along their left arms respectively (excluding the root in each case). Some k_i's may be zero. Using theorem1(a), since each of these nodes is the left child of its parent, the splay depths of the k_i nodes along the left arm of x_i are bounded respectively by

$$d_i/2 + 3/2$$

$$(d_i/2 + 3/2)/2 + 3/2 = d_i/4 + 3/4 + 3/2$$

$$(d_i/4 + 3/4 + 3/2)/2 + 3/2 = d_i/8 + 3/8 + 3/4 + 3/2$$

. . . . . .

. . . . . .

. . . . . .

and so on and therefore adding the above expressions vertically it follows that the sum of the splay depths of the k_i elements is bounded by $(d_i + 3k_i)$. Thus the sum of the splay depths of all the elements in all such left chains is bounded by

$$\sum d_i + 3 \sum k_i \leq 2n + 3n = 5n.$$

Since the time to splay the elements on the left arm of T is at most n, the total time to splay all the elements of T is necessarily bounded by 8n. This completes the proof.

REFERENCES

1. D.D.Sleator and R.E.Tarjan: Self-adjusting binary search trees, J.A.C.M , 32, 1985, 652-686.
2. R.E.Tarjan: Data Structures and Network Algorithms, C.B.M.S 44 (1983),S.I.A.M , Philadelphia, Pa.
3. R.E.Tarjan: Sequential access in splay trees takes linear time, Combinatorica, 5(4),1985, 367-378.
4. R.E.Tarjan: Amortized Computational Complexity, S.I.A.M Journal of Applied Discrete Methods, 6, 1985, 545-568.
5. D.D.Sleator, R.E.Tarjan and W.Thurston: Rotation distance, Triangulations and Hyperbolic Geometry, J. American Mathematical Society, 1 ,1988, 647-681.