PREPRINT. Computer Journal, 33(5):471-472, 1990.

A Note on the Expected Behaviour of Binary Tree Traversals

Arne Andersson
Department of Computer Science
Lund University
Lund, Sweden

Abstract

Brinck and Foo [1, 2] have analyzed tree traversal algorithms using threads and stack. In this note we show that, contrary to the results in the two referred papers, a preorder traversal is performed faster with a stack than with threads, in terms of pointer assignments. Moreover, the preorder-stack traversal is faster than any of the other standard traversals analyzed in [1] and [2].

1 Introduction

It is well known that adding threads to a binary tree makes it possible to traverse the tree in preorder, inorder or postorder without using any stack. Although the asymptotic complexity are the same for the stack and thread algorithms the time constant may differ. A comparison of the two methods has been made by Brinck and Foo [1, 2]. They present efficient algorithms for each traversal and analyze their average behaviour for uniformly distributed trees and randomly generated (i. e. generated by insertions from a random permutation) binary search trees. As a measure of the cost they use the number of pointer assignments required, including pushing and popping a pointer on and off a stack. According to their results the thread algorithm runs about 25% faster for inorder traversal. For preorder traversal the difference is small when traversing trees of uniform distribution while the difference for randomly generated trees, which are more likely to occur in reality, is 40%.

In this note we show that a preorder traversal with a stack can be performed faster than what is computed by Brinck and Foo. We are only concerned with the preorder traversal and for a detailed presentation and analysis of the other traversals we refer to [1] and [2].

2 The Original Preorder Algorithm

The preorder-stack algorithm analyzed in [1, 2] is given in Figure 1. The algorithm uses a stack containing all visited nodes wich have a non-visited right-child. The successor of a node v is found as follows.

1. *v has two children.* We push v on the stack and proceed to the left-child.

2. *v has one child.* We proceed to this child.

3. *v is a leaf.* We pop a node from the stack and proceed to the right-child of this node.

During a traversal each node in the tree except the root is retrieved by a pointer assignment and each node with two children is pushed and popped once. Let n be the number of nodes in the tree and let n_2 be the number of nodes with two children. The traversal cost can be expressed as

$$n - 1 + 2n_2 \tag{1}$$

From [1, 2] we know that the average value of n_2 is $\frac{(n-1)(n-2)}{2(2n-1)}$ for uniformly distributed trees and $\frac{n-2}{3}$ for randomly generated binary search trees. Thus the average cost for the uniform distribution is asymptotic to

$$\frac{3n}{2} - \frac{9}{4} \tag{2}$$

as n increases. For randomly generated search trees the average cost is

$$\frac{5n}{3} - \frac{7}{3} \tag{3}$$

3 An Improved Algorithm

A closer look at the preorder algorithm by Brinck and Foo shows that we may reduce the required number of pointer assignments. Instead of keeping the nodes with unvisited right-childs on the stack we can keep the right-childs themselves. In this way we avoid taking the circuitous route over a node when passing from its left to its right subtree. We find the successor of v in the following way:

1. *v has two children.* We push v's right-child on the stack and proceed to v's left-child.

2. *v has one child.* We proceed to this child.

3. *v is a leaf.* We pop a node from the stack and proceed to this node.

A complete traversal algorithm with this modification is given in Figure 2. The difference between the two algorithms is small but gives a significant decrease in the number of pointer assignments, since we save one assignment per node with two children. We have the following traversal cost:

1. Each node is retrieved either by a pointer assignment or by popping the stack.

2. Each right-child node with a sibling is pushed on the stack once.

The cost for this can be expressed as

$$n - 1 + n_2 \tag{4}$$

Thus the expected number of pointer assignments for the uniform distribution is

$$\begin{aligned} & n - 1 + \frac{(n-1)(n-2)}{2(2n-1)} \\ &= \frac{5n}{4} - \frac{13}{8} + \frac{3}{16n-8} \end{aligned} \tag{5}$$

which is asymptotic to

$$\frac{5n}{4} - \frac{13}{8} \tag{6}$$

as n increases. For randomly generated trees the cost is

$$\begin{aligned} & n-1+\frac{n-2}{3} \\ & =\frac{4n}{3}-\frac{5}{3} \end{aligned} \tag{7}$$

4 Comments

In Table 1 the performance of the modified traversal algorithm is compared to the cost of other algorithms. The maybe surprising result is that a preorder traversal may be performed faster (at least in the sense of pointer assignments) using a stack instead of threads. This is not true only for randomly generated and uniformly distributed trees but for all trees, which is shown below.

Theorem 1 *Let P_S be the number of pointer assignments required by our stack algorithm and P_T the number required by the thread algorithm used in [1, 2]. Then $P_S \leq P_T$ for any binary tree.*

Proof: The algorithm given in [1, 2] to find the preorder successor in a threaded tree is as follows:

1. *The node has at least one child.* Proceed to the leftmost child.
2. *The node is a leaf.* Follow right-threads until a node with two children is found. Proceed to the right-child of this node.

It is not hard to see that the number of pointer assignments required by this algorithm is

$$n-1+n_2+\textit{number of nodes with only a left-child} \tag{8}$$

Clearly, this is $\geq n-1+n_2$, which is the cost for our stack algorithm. This completes the proof. □

In fact, from the table we can conclude that the cost for a preorder-stack traversal is lower than for any of the other traversals analyzed in the papers referred. However, as pointed out by Brinck and Foo the algorithms do not consist only of pointer assignments and therefore the differences in the values in Table 1 do not necessarily correspond to a difference in real execution time.

References

[1] K. Brinck. The expected performance of traversal algorithms in binary trees. *Computer Journal*, 28(4), 1985.

[2] K. Brinck and N. Y. Foo. Analysis of algorithm on threaded trees. *Computer Journal*, 24(2), 1981.

```
type     Pointer = ↑Node;
         Node =  record
                       (* data *)
                       left, right: Pointer;
                 end;

procedure  PreScan (p: Pointer);
var  S: Stack;
begin
     Clear (S);
     while  p ≠ nil do
     begin
          while  p↑.left ≠ nil do
          begin
               Visit (p);
               if  p↑.right ≠ nil then
                    Push (S, p);
               p := p↑.left;
          end;
          Visit (p);
          if  p↑.right = nil then
               Pop (S, p);
          p := p↑.right;
     end;
end;
```

Figure 1: Preorder-stack traversal by Brinck and Foo. We assume the tree to be declared with the type definitions above. We also assume an abstract data type Stack with the operations Clear, Push and Pop to be defined. The procedure Visit(p) processes the information in the node p.

```
procedure PreScan (p: Pointer);
var S: Stack;
begin
    Clear (S);
    while p ≠ nil do
    begin
        while p↑.left ≠ nil do
        begin
            Visit (p);
            if p↑.right ≠ nil then
                Push (S, p↑.right);
            p := p↑.left;
        end;
        Visit (p);
        if p↑.right = nil then
            Pop (S, p);
        else
            p := p↑.right;
    end;
end;
```

Figure 2: The improved algorithm for preorder traversal.

	stack algorithm	thread algorithm
uniform distribution		
preorder by Brinck and Foo	$\frac{3n}{2} - \frac{9}{4}$	$\frac{3n}{2} + \frac{3}{2}$
new preorder	$\frac{5n}{4} - \frac{13}{8}$	
inorder	$2n - 2$	$\frac{3n}{2} + \frac{1}{2}$
postorder	$4n - 3$	$4n - 6$
random search trees		
preorder by Brinck and Foo	$\frac{5n}{2} - \frac{7}{3}$	$\frac{3n}{2} + \frac{3}{2}$
new preorder	$\frac{4n}{3} - \frac{1}{3}$	
inorder	$2n - 2$	$\frac{3n}{2} + \frac{1}{2}$
postorder	$4n - 3$	$4.19n - 2h_n$

Table 1: *Average costs for traversal algorithms. All values except for the new preorder algorithm are taken from Brinck and Foo.* h_n *is the nth harmonic number* $(= 1 + 1/2 + 1/3 + ... + 1/n)$.