Oriental Journal of Computer Science & Technology Vol. **4**(1), 217-219 (2011)

Reconstruction of a Binary Search Tree from its Preorder Tree Traversal with the Unique Non-recursive Approach

MANOJ C. LOHANI, UPENDRA S. ASWAL and RAMESH S. RAWAT

Department of Computer Science, Graphic Era University, Dehradun (India).

(Received: March 10, 2011; Accepted: April 18, 2011)

ABSTRACT

This paper presents a new approach of reconstruction of Binary search tree using its Pre order tree-traversal only. There are many approaches given with the help of combination of two- tree traversals. But, in this paper we have not used any other combination of tree traversals to reconstruct the Binary search tree. Our work shows the implementation of this algorithm in C language. Our algorithm is found to be very simple and faster than other non recursive algorithms due to its unique implementation. Due to this reason the time and space complexities are significantly reduced.

Key words: Binary Search Tree Reconstruction, Non Recursive Algorithm, Pre order Traversal, Binary Search Tree, Struct.

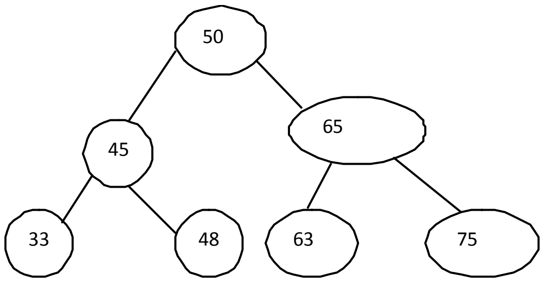
INTRODUCTION

The tree is a fundamental structure in computer science. Almost all operating systems store files in trees or tree like structures. Trees are also used in compiler design, text processing, and searching algorithms. A binary tree is an ordered tree in which each node has maximum of two children, referred to as a left and a right tree. A binary tree is different recursively as either empty or consists of a root, a left tree, and a right tree.

The left and right trees may themselves be empty, thus a node with one child could have a left or a right child. Commonly, there are three traversing methods: in order, pre order and post order traversals.

In preorder traversal, first, process the current node after this, the left child is processed, and finally the right child is processed. The output of a preorder traversal algorithm of the binary search tree shown in Figure 1is: 50,45,33,48,65,63,75.

This paper describes a non recursive algorithm for reconstructing the original binary search tree from its preorder traversal only and its implementation in C language.



Related reconstruction algorithms

It is well known that given the inorder traverse of a binary tree, along with one of its preorder or post order traversals, the original binary tree can be uniquely identified. It is not difficult to write a recursive algorithm to reconstruct the binary tree[1]. The computation time required is $O(n^2)$ where n is the number of nodes in the tree.

H A Burgdorff[5] presented a non recursive algorithm for reconstructing a binary tree from its inorder-preorder sequences (in short i-p sequences)[7] and their algorithm takes $O(n^2)$ computation time. Chen[4] has also proposed a non recursive algorithm from its i-p sequence array of time completing O(n) and inefficient space. The non-recursive algorithms for reconstructing the original binary tree from its inorder and preorder traversal are done in two stages in the algorithm proposed by G H Chen[4].

Vinu V Das[8] has also proposed a non recursive algorithm for reconstruction of binary tree with its preorder and in order tree traversal.

Proposed non-recursive reconstruction algorithm

We have proposed a non-recursive algorithm to reconstruct the binary search tree using a single tree traversal i.e., **preorder only** .Here we have developed the algorithm for Binary search tree and we have used only Preorder tree traversal of Binary search tree.

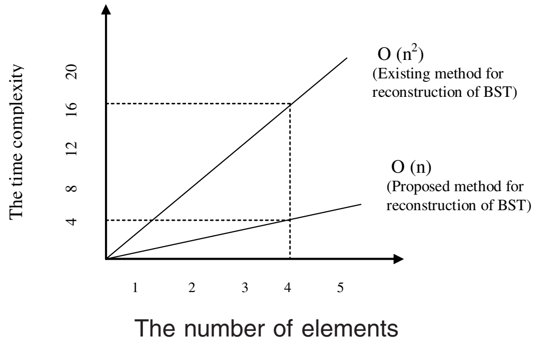
Here in this algorithm we have Preorder of the Binary search tree, which we have stored in an array say, preorder. Since in Preorder, the root node of the tree will be the first element from preorder, therefore this first element of an array preorder will be the root node for the resultant binary search tree.

Now we will fetch the next element from preorder, and compare this value with root and if this fetched element is smaller than the root then it will be placed in the left hand side of a root otherwise in the right hand side of a root. This process will continue for all elements of the array preorder.

The complete algorithm is given below

```
struct bst          //Node structure
  {

int data;

        struct bst *left,*right;

    };

    int     preorder[]={50, 45, 33, 48, 65, 63, 75};
//Preorder of Binary Search Tree
//shown above

    struct bst * RECONSTRUCT_BST(struct bst*,
int); //Function declaration for
//reconstruction of a binary //search tree

    struct bst * RECONSTRUCT_BST (struct bst
*root, int n) //Function definition
    {
        int i=0,data=0;
        struct bst *p,*nd,*x;
        for(i=0;i<n;i++)
       //Loop to fetch and
//reconstruct the BST until all //the element are
fetched
         {
           data=preorder[i];
                //Fetch the elements

            nd = (struct bst*) malloc(sizeof(struct
bst));    //Create new node
            nd->data=data;
            nd->left=NULL;
            nd->right=NULL;
            if(root==NULL)
             {
                root=nd;
                }
                else
                {
                          p=root;
                          x=p;
                     while(1)
                         {
                         x=p;
                         if(data > (p->data))
                          p=p->right;
                          else
                          p=p->left;
                           if(p==NULL && (x-
>data) > data)
                           {
                             x->left=nd;
                             break;
                //insert the element on the
//left side of BST
                            }
                            if(p==NULL && (x-
>data) < data)
                            {
```

```
                    x->right=nd;
//insert the element on the
                  break;
// right side of BST
  }
                  }
            }
  }
   return root;
 }
```



Time complexity of existing and proposed algorithms

The computation time required for executing the reconstruction algorithm is of O(N) and the space complexity is O(NlogN).

CONCLUSIONS

In this paper, a non-recursive algorithm for reconstructing the original binary search tree from its preorder traversal sequence is proposed. The new non-recursive algorithm is easy to understand and is more efficient than the algorithms which were given in the references[4,7-8]. While the algorithm in reference[4] has got poor time complexity, the proposed algorithm reduces the time complexity significantly though it is meant for binary search tree only.

REFERENCES

1. D.E. Kunth, "The Art of Computer Programming", Vol. 1: Fundamental Algorithm, Addison-Wesley, Reading, Mass., (1973).
2. D.E. Kunth, "The Art of Computer Programming", Vol. 3: Sorting and Searching, Addison-Wesley, Reading, *Mass,* (1973).
3. G. D Knott, "A Numbering System for Binary Trees", *Comm. of ACM,* **20**(2): 113 (1977).
4. G.H. Chen, M.S. Yu, and L.T. Liu, "Non-recursive Algorithms for Reconstructing a Binary Tree from its Traversals", *IEEE Comm.*, **88**: 490-492 (1988).
5. H.A. Burgdorff, S. Jojodia, F.N. Springsteel, and Y. Zalcstein, "Alternative Methods for the Reconstruction of Tree from their Traversals", *BIT,* **27**(2): 134 (1987).
6. J. Driscoll and Y. Lien, "A Selective Traversal Algorithm for Binary Search Tree", *Comm. of ACM*, **21**(6) 445-447 (1978).
7. T. Hikita, "Listing and Counting Subtrees of Equal size of a Binary Tree", *Inform Process Lett.,* **17**(4): 225 (1983).
8. Vinu. V. Das A new Non-Recursive Algorithm for Reconstructing a Binary Tree from its Traversals. *International Journal of Algorithms, Computing and Mathematics* **2**(1): (2009).