International Journal of Computer Applications (0975 – 8887)
Volume 43– No.10, April 2012

Modified Non-Recursive Algorithm for Reconstructing a Binary Tree

Nitin Arora
Dept. of Computer Science & Engineering
G. B. Pant Engineering College, Pauri, Uttarakhand, INDIA

Vivek Kumar Tamta
Dept. of Computer Science & Engineering
G. B. Pant Engineering College, Pauri, Uttarakhand, INDIA

Suresh Kumar
Dept. of Computer Science & Engineering
G. B. Pant Engineering College, Pauri, Uttarakhand, INDIA

ABSTRACT

Binary tree traversal refers to the process of visiting each node in a specified order. Given the inorder traversal of a binary tree, along with one of its preorder or postorder traversals, the original binary tree can be uniquely identified. Many recursive and non recursive method of construction of the tree from inorder and any of the postorder or preorder traversal have been proposed. In this paper one of the proposed algorithms has been examined. This algorithm computes the wrong tree for some input sequences. We show a particular situation in which the algorithm fails and a solution for this situation is proposed. The proposed a modified non-recursive algorithm for reconstructing a binary tree which generates the correct tree otherwise an error has been reported.

Keywords

Binary Tree Reconstruction, Traversal, Non Recursive Algorithm.

1. INTRODUCTION

There are many data structures (linear or non-linear) used in computer science. The tree is one of the non-linear fundamental data structure. Almost all the operating systems store files in trees or tree like structures. We use trees in compiler design, text processing and searching algorithms [1]. A binary tree is an ordered tree in which each node has maximum of two children, referred to as left and right tree. [1, 2, 3, 4, 5] A binary tree can either be empty or recursively consists of a root, left sub tree and right sub tree. Commonly there are three traversing methods:

- Inorder Traversal
- Preorder Traversal
- Postorder Traversal

In an inorder traversal first the left child is processed recursively, then processes the current node followed by the right child. In preorder traversal first the root is processed followed by the left child and the right child. Similarly in postorder traversal first the left child, then right child followed by the current node processed recursively. [6]
The output of the different tree traversal algorithm of the binary tree shown in figure1 is [1]:
inorder traversal: F, J, G, A, H, K, I, B, C, L, D, E
preorder traversal: A, F, G, J, B, H, I, K, C, D, L, E
postorder traversal: J, G, F, K, I, H, L, E, D, C, B, A

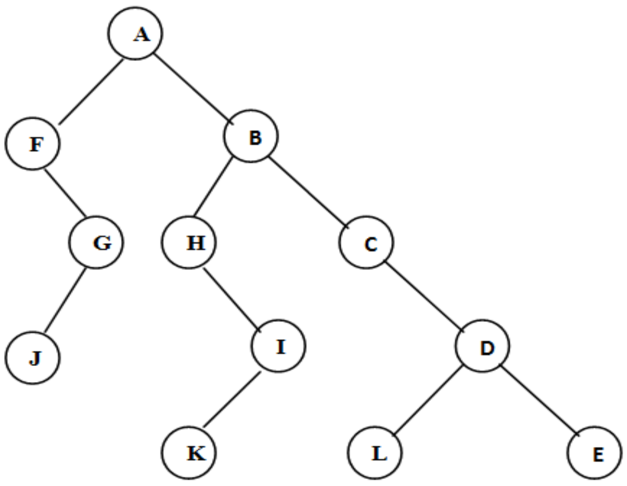


Fig 1: Binary Tree Containing 12 nodes

And output of the different tree traversal algorithm of the right skewed binary tree shown in figure2 is:
inorder traversal: A, B, C, D, E
preorder traversal: A, B, C, D, E
postorder traversal: E, D, C, B, A

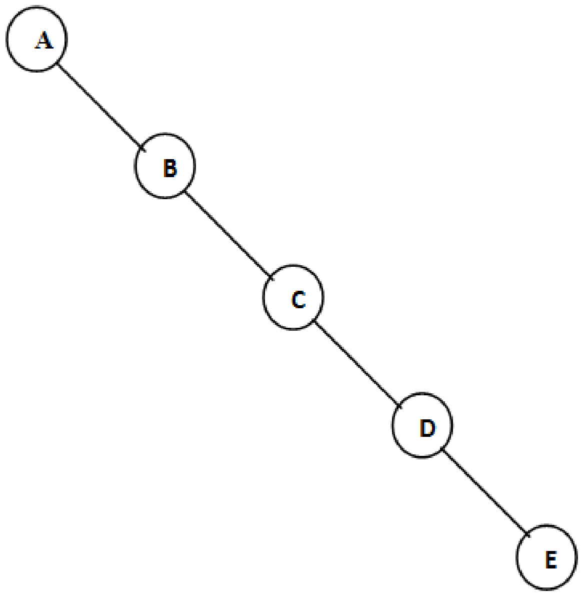


Fig 2: Right Skewed Binary Tree Containing 5 nodes

For a correct constructed tree the same inorder, preorder and postorder sequences must be generated from the constructed tree. In this paper we have point out a correction in the algorithm proposed by Vinu V Das "*A new non-recursive algorithm for reconstructing a binary tree from its traversals*" [1]. This algorithm computes the wrong binary tree for some input sequences. We have shown a particular situation in which the algorithm used by Vinu V Das fails and a solution for this situation is proposed. The proposed solution generates the correct tree if the tree can be constructed otherwise an error has been reported i.e. tree can't be constructed from the given input sequences.

The remainder of this paper is organized as follows. Section 2 introduces the related work; Section 3 describes our proposed solution, Section 4 describes performance evaluation, conclusion and future scope in Section 5, given acknowledgments in Section 6 and all the used references are given in section 7.

2. RELATED WORK

A unique original binary tree can be constructed from the given inorder traverse of a binary tree, along with one of its preorder or postorder traversals. Binary tree reconstruction from its traversals is not a difficult task [7]. The computation time required is O(n^2) where n is the number of nodes in the tree. A non-recursive algorithm for reconstructing a binary tree from its inorder-preorder sequences (in short i-p sequences) [5] had been presented by H A Burgdorff [8] and their algorithm takes O(n^2) computation time. A non recursive algorithms from for constructing a binary tree has been proposed by Chen [9] and it takes of time completing O(n) and inefficient space. Vinu V Das [1] presented a non-recursive algorithm for reconstructing a binary tree from its inorder-preorder sequences (in short i-p sequences) and their algorithm takes O(n) time and O(nlogn) space. This non-recursive algorithm works on the following lemma. The algorithm proposed by Vinu V Das is based on following lemma [1]:

Lemma: *Let P[j] be the parent of P[i], (where j < i) then Lp[j] be the corresponding index in the inorder sequence of node P[j] and Lp[i] be the corresponding index in the inorder sequence of node P[i]. If Lp[j] is greater than Lp[i] then P[i] is the left child of P[j] otherwise it is the right child. The binary tree can be reconstructed by exploiting the following three properties of the above lemma.*
1. First node in the preorder sequence will be the root of the binary tree
2. The node(s) which comes to the left of root node in the inorder sequence will be the nodes in the left sub tree and node(s) in the right are the nodes in the right sub tree of the reconstructed root node.
3. Apply the second property recursively to the left and right sub tree of the root node to obtain the complete binary tree.

Following is the reconstruction algorithm for binary tree from its traversals.[1]

```
/*Variable Declarations*/
left = 0; right = 0; count1 = 0; count2 = 0; data = P[0]; root
= NULL;
for ( count1 = 0  ; count1<n  ;  count1++  )
{
        /*Create a new node*/
        newnode->info = data;
        newnode->lchild = NULL;
        newnode->rchild = NULL;

        If ( root == NULL )
        root=newnode;
        /*Intially the linked list is empty*/
    while ( I[count2] is present in the linked  list )
        {
                presentnode = address of I[count2];
                remove the I[count2] from the linked list;
                right=1;left=0;count2++;
        }
        If( data != I[count2] )
        {
                Add the data into the linked list;
                If ( left == 1)
        Place the newnode as left child of presentnode
                If( right == 1)
        Place the newnode as right child of presentnode
                left=1;right=0;
        }
        else
        {
                If( left == 1)
        Place the newnode as a left child of presentnode
                elseif( right == 1)
        Place the newnode as right child of presentnode
                left=0;right=1;count2++;
        }
        data=P[count1+1];
        presentnode = newnode;
} /* for loop end */
```

3. MODIFIED NON-RECURSIVE RECONSTRUCTING ALGORITHM

The algorithm reference [1] is implemented in C language and checked for some different input sequences. The algorithm worked correctly for some sequences but on the other side it generated a wrong binary tree for some other input sequences. We made some changes in the algorithm according to the proposed solution as: If before printing the binary tree we first generate the related sequences and compare with in input sequences supplied. If both the sequences are same then only the correct binary tree can be generated otherwise an error message can be reported.

The modified Non-Recursive algorithm for generating a tree from its inorder and preorder traversals is:

```
/*Variable Declarations*/
left=0; right=0; count1=0; count2=0; data=P[0]; root=NULL;
m=0;  /* New variable in our program which keeps a count of
the number of nodes having 2 children */
for ( count1 = 0  ; count1<n  ;  count1++  )
{
        /*Create a new node*/
        newnode->info = data;
        newnode->lchild = NULL;
        newnode->rchild = NULL;
        If ( root == NULL )
        root=newnode;

        /*Intially the linked list is empty*/
        while ( I[count2] is present in the linked list )
        {
                presentnode = address of I[count2];
                remove the I[count2] from the linked list;
                right=1;left=0;count2++;
                /*Here a small modification is done*/
                m++;
        }

        If( data != I[count2] )
        {
                Add the data into the linked list;
                If ( left == 1)
                Place the newnode as left child of
presentnode
                If( right == 1)
                Place the newnode as right child of
presentnode
                left=1;right=0;
        }
        else
        {
                If( left == 1)
                Place the newnode as a left child of
presentnode
                elseif( right == 1)
                Place the newnode as right child of
presentnode
                left=0;right=1;count2++;
        }
        data=P[count1+1];
        presentnode = newnode;
} /*For loop end*/

If (m is greater than zero or check if the inorder traversal of
the tree formed matches with the given inorder sequence )
{
        Print the tree ;
}
else {
        Print that a legitimate tree cannot be constructed
with the sequences;
}
```

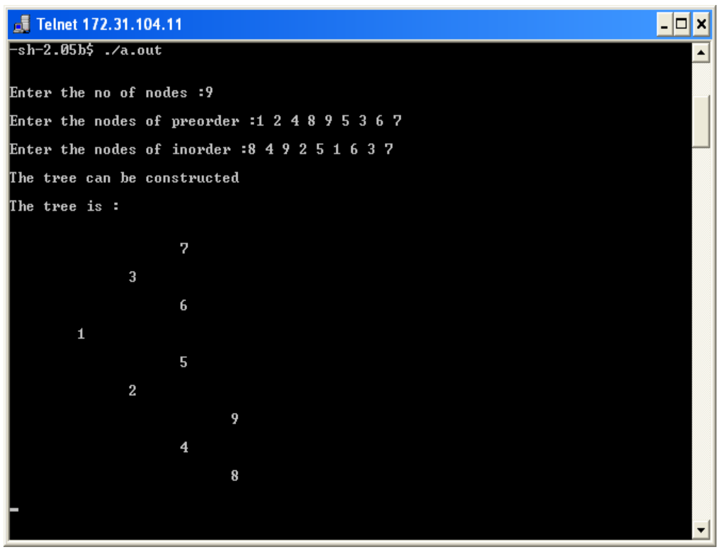
The time complexity of our modified algorithm is O(n) where n is the number of nodes.

4. PERFORMANCE EVALUATION

We have checked both the algorithms for different types of input sequences and generated the binary tree.

4.1.Sample input sequences and output tree in vertical fashion:

Supplied input sequence preorder: 1 2 4 8 9 5 3 6 7 and inorder: 8 4 9 2 5 1 6 3 7 to the previous algorithm and our proposed algorithm. Both the algorithm generates the same output correct tree (shown in vertical fashion) as shown in figure 3.

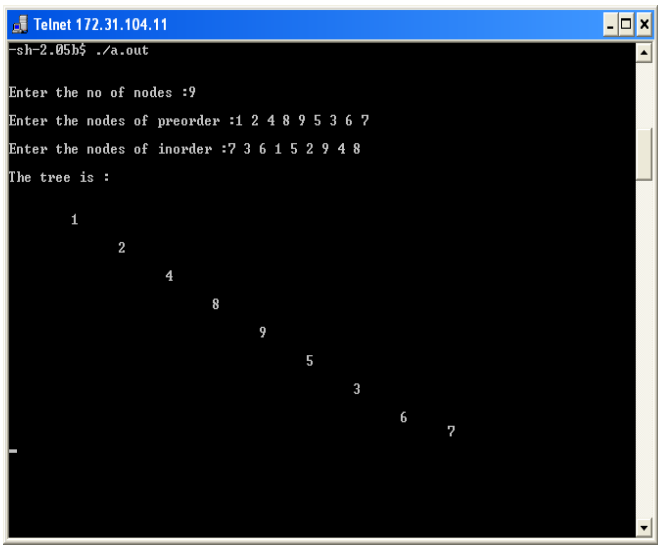


```
-sh-2.05b$ ./a.out

Enter the no of nodes :9
Enter the nodes of preorder :1 2 4 8 9 5 3 6 7
Enter the nodes of inorder :8 4 9 2 5 1 6 3 7
The tree can be constructed
The tree is :

                    7
              3
                    6
        1
                    5
              2
                          9
                    4
                          8
_
```

Fig 3: Correct Generated tree by the algorithm

Above produced binary tree is correct binary tree because we can generate the same given preorder and inorder sequences from it.

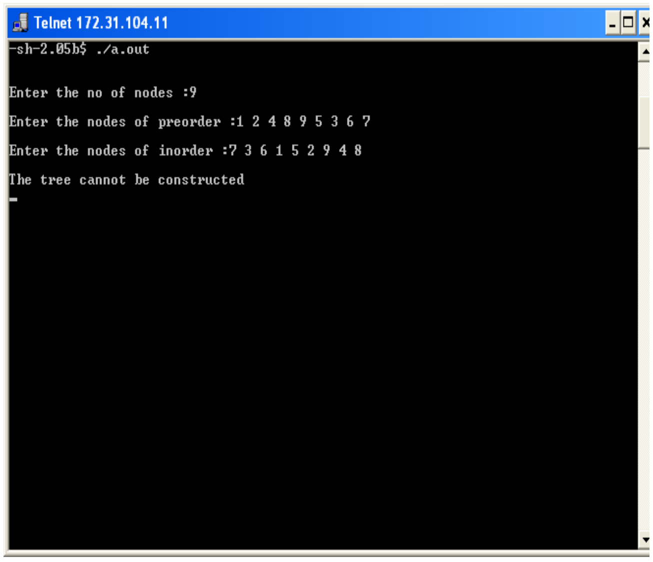
If we supplied the different input sequence preorder: 1 2 4 8 9 5 3 6 7 and inorder: 7 3 6 1 5 2 9 4 8 to both the algorithms. We got a binary tree from the previous algorithm as shown in figure 4. This tree is wrongly constructed. We are not able to generate the same sequences from the generated binary tree.



```
-sh-2.05b$ ./a.out

Enter the no of nodes :9
Enter the nodes of preorder :1 2 4 8 9 5 3 6 7
Enter the nodes of inorder :7 3 6 1 5 2 9 4 8
The tree is :

        1
              2
                    4
                          8
                                9
                                      5
                                            3
                                                  6
                                                        7
_
```

Fig 4.A wrong binary tree has been constructed from the previous algorithm.

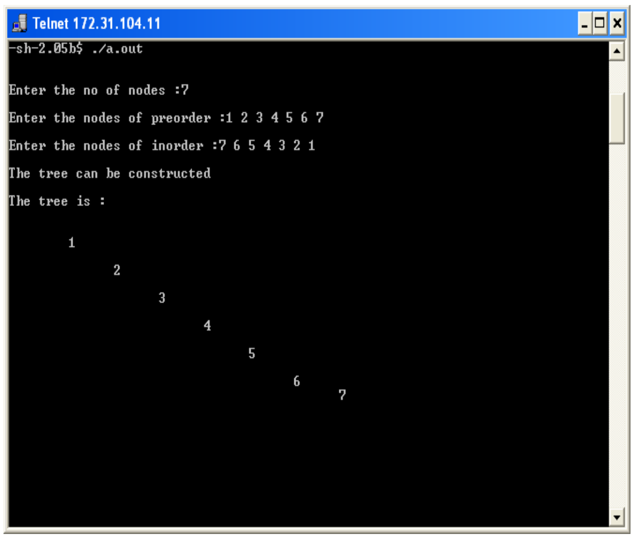
In the same case our proposed algorithm performs better. It generates an error message if the supplied input sequences are not correct as shown in figure 5.



```
Telnet 172.31.104.11
-sh-2.05b$ ./a.out

Enter the no of nodes :9

Enter the nodes of preorder :1 2 4 8 9 5 3 6 7

Enter the nodes of inorder :7 3 6 1 5 2 9 4 8

The tree cannot be constructed
_
```

Fig 5: Error message has been generated if the supplied input sequences are not correct.

If the input sequences are preorder: 1 2 3 4 5 6 7 and inorder: 7 6 5 4 3 2 1 then the correct binary tree can be constructed using both the algorithms as shown in figure 6.



```
Telnet 172.31.104.11
-sh-2.05b$ ./a.out

Enter the no of nodes :7

Enter the nodes of preorder :1 2 3 4 5 6 7

Enter the nodes of inorder :7 6 5 4 3 2 1

The tree can be constructed

The tree is :


        1

              2

                    3

                          4

                                5

                                      6
                                            7
```

Fig 6: constructed binary tree.

5. CONCLUSION AND FUTURE SCOPE

The proposed solution has been successfully implemented in C language and checked for the different input sequences. Our proposed solution did not create a tree for a combination which is not correct. Also in the future better algorithm can be find out than the above proposed one which would further reduce the complexity of the algorithm.

6. ACKNOWLEDGMENTS

Our thanks to Dr. A.K. Swami, Principal, G. B. Pant Engineering College, Ghurdauri, for providing necessary infrastructure for the research work. We would also like to thank Mr. Sashi Kant Verma, Head, Department of Computer Science and Engineering, G. B. Pant Engineering College, Ghurdauri, for his unconditional and valuable support in writing this paper.

7. REFERENCES

[1] Vinu V Das, “A new Non-Recursive Algorithm for Reconstructing a Binary Tree from its Traversals”, IEEE Comm., pp. 261-263, 2010

[2] Vinu V Das, “Principles of Data Structures Using C and C++”, New Age International Publishers, Reading, Mass., 2005.

[3] M. Weiss, Data Structures & Problem Solving Using Java, 2"d ed., Addison Wesley, 2002

[4] D. E. Knuth, The Art of Computer Programming, Vol. 3 (2nd ed.): Sorting and Searching, Addison Wesley, 1998.

[5] J. Driscoll, and Y. Lien, A Selective Traversal Algorithm for Binary Search Trees, Communications of the ACM, Number 6, Vol. 21, 1978, pp. 445-447.

[6] R. Sedgewick, Algorithms in Java, 3d edition, Addison Wesley, 2003

[7] D.E. Kunth, “The Art of Computer Programming”, Vol. 1:Fundamental Algorithm, Addison-Wesley, Reading, Mass., 1973

[8] H.A. Burgdorff, S. Jojodia, F.N. Springsteel, and Y.Zalcstein, “Alternative Methods for the Reconstruction of Tree from their Traversals”, BIT, Vol. 27, No. 2, p. 134, 1987

[9] G.H. Chen, M.S. Yu, and L.T. Liu, “Non-recursive Algorithms for Reconstructing a Binary Tree from its Traversals”, IEEE Comm., Vol. 88, pp. 490-492, 1988.