Journal Of Harmonized Research (JOHR)

Journal Of Harmonized Research in Engineering
5(3), 2017, 127-132

ISSN 2347 – 7393

Original Research Article

INORDER AND PREORDER TRAVERSAL & DISPLAY OF EXISTING BINARY TREE IN COMPUTER MEMORY

P K Kumaresan

Professor Computer Science and Engineering, VMVK Engineering College
Vinayaka Missions University, Salem 636308 Tamil Nadu

Abstract: Depending on three different sequences of traversal of a binary tree, we get its preorder, postorder and inorder form. Similarly depending on preorder or postorder along with inorder, the binary tree can be formed. In this paper, we try to implement, through coding a binary tree is formed physically not virtually in computer memory and also give a realistic view of a binary tree, already created in computer memory. Using inorder and preorder traversal of binary tree and takes the value of the level of the tree we are able to graphically represent any binary tree.

Keywords: Binary Tree, Inorder, Preorder, Link List, Structure Pointer, Recursive Function, Data Structure.

1. Introduction*:* We are able to display a binary tree in the inorder preorder and postorder form. Binary tree is either empty or consist of a vertex, called root, together with two disjoint binary tree called left sub-tree and right sub-tree. There are three ways of traversal of a binary tree, preorder, inorder and postorder. Traversing with the following sequence, left sub-tree, Root and right sub-tree, Inorder form of binary tree is formed. Similarly post order of binary tree is formed for traversal, having the sequence left sub-tree, right sub-tree and the root.

For Correspondence:
pkkumaresan@hotmail.com
Received on: June 2017
Accepted after revision: July 2017
Downloaded from: www.johronline.com

To do it, at first we fetch one by one element from the left side of the preorder and then we search the position of that element in the inorder form, then break the link list into to parts and consider it as left and right child. Again the left and right child is treated as new inorder form, individually.

After creation of binary tree in computer memory, we are going to give a tree like view of an already created binary tree that is a graphical representation of a binary tree. Depending on the level of each node in the binary tree and sequence of nodes in inorder traversal of the binary tree, position of each node is calculated for plotting the node with respect to the graph paper. After calculation of position of each node, for each two nodes, pairs are created such that

they are having parent-child relationship. Lines are drawn between positions of two nodes in a pair.

Section 2 represents the algorithm for creation of binary tree from the correct inorder and preorder traversal and graphically representation of binary tree. To illustrate the algorithm more clearly an implementation is shown in section 3. Result of the execution of an example is shown in section 4. Section 5 analyses the concept. A conclusion is drawn in section 6.

2. The Scheme: In this section, algorithm of physically creation binary tree from given inorder and preorder form is described. For it a special kind of node, structure of which show is figure 2.1, is designed.

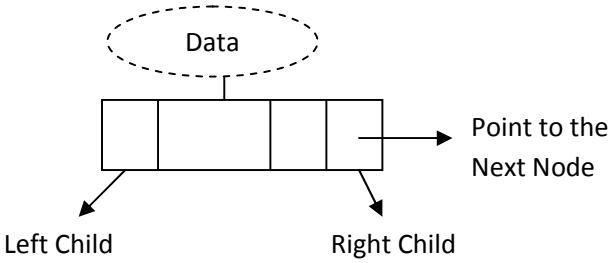

Figure 2.1: Structure of a Node

Section 2.1 and 2.2 describe the way to create a binary tree in computer memory from inorder and preorder traversal. Section 2.3 and 2.4 define the process for representing the binary tree with a realistic view.

2.1 Formation of binary tree from inorder and preorder traversal: Only from inorder and preorder traversal of the binary tree, the corresponding tree will be constructed. To construct the binary tree, following steps are used.

Step 1: A link list is created having first node as ROOT and same sequence, inorder form.

Step 2: Call the function, named b_tree, defined in section 2.2 with ROOT as passing argument.

Step 3: Now Binary tree is formed. The root node of which is set as ROOT.

2.2 Defining A function, Named b_tree(): After creating the link list with the same sequence of inorder traversal, the following steps are used to corresponding binary tree that is in function form.

Step 1: Start function, named b_tree taken an argument h as a first node of link list.

Step 2: If frist element of preorder has already considered, next element is kept into E. Otherwise first element is kept into E.

Step 3: Element E is reached the link list, starting with h node.

Step 4: If E is found in link list, say in p node then following 4 steps are followed.

1. Link list is broken into two parts.
2. Making link list left position of p node is kept as left child and right position is kept as right child.
3. Call b_tree with first node of link list, kept left child of p as argument.
4. Call b_tree with first node of the link list kept as right child q as argument.

Step 5: End function.

2.3 Placing for every node of a binary tree properly: This is the first part of the total algorithm that calculates and puts each node in their proper position through the following steps.

Step 1: Start function to traverse all nodes of the created Binary tree.

Step 2: Assign a column value of each node with the position of particular node in inorder traversal of the binary tree.

Step 3: Assign row value of each node with its corresponding level.

Step 4: After counting the maximum co-ordinate along X and Y axis of the graph paper or for here computer screen, two arbitrary integer variable r_gap and c_gap are respectively assigned with integer part of maximum co-ordinate along Y-axis÷(total depth of the tree+1) and maximum co-ordinate along X-axis÷(total number of nodes+1).

Step5: Print the node in its corresponding position, calculate with (row× r_gap,column× c_gap).

Step 6: Call the function, named “drawline”, defined in the section 2.4 with root node of the binary tree.

2.4 Defining a function, named drawline(): For a complete tree like view connection between each parent node and child node is made through drawing lines with following steps which is recursively called.

Step 1: Start function, named "drawline", taking an argument, R as root node of a binary tree.

Step 2: If R has a left child, step 2.1 and 2.2 are followed.

2.1 A line is drawn between R and its right child where positions of both the nodes have already calculated in Step 5 in algorithm 2.3.

2.2 Call the function "drawline" with the passing the argument of its left child.

Step 3: If R has a right child, Step 3.1 and 3.2 are followed.

3.1 A line is drawn between R and its right child where positions of both the nodes have already calculated in Step 5 in the algorithm 2.3.

3.2 All the function "drawline" with the passing the argument of its right child.

Step 4: End function.

3. Implementation: To illustrate the algorithm, defined in section 2, an example is taken for real implementation. Respectively section 3.1 and section 3.2, describe the process of creation of binary tree from inorder and preorder and displaying the tree with its realistic view.

3.1: Creation of Binary Tree: Let us assume that {A, S, D, F, G, H, J} is considered as inorder of a binary tree and the preorder is {F, S, A, D, H, G, J}.

Step 1: A link list, with root as a node created with {A, S, D, F, G, H, J} that is shown in the figure. 3.1.1.

Step 2: Element F (1st element in preorder) is found in P node of link list. It is broken into left and right from F which is attached as left and right child of the node, containing with F. That is shown in the figure. 3.1.2.

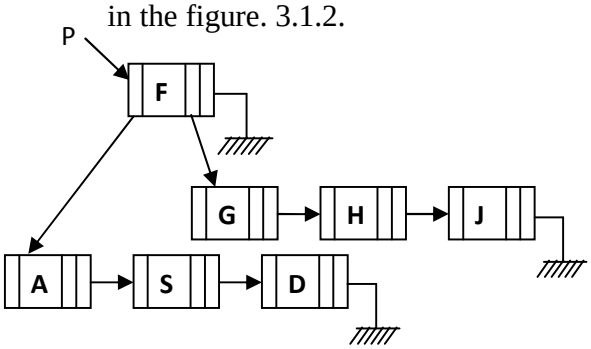

Figure 3.1.2: Creation of Left and Right Child of P-Node, contained with F

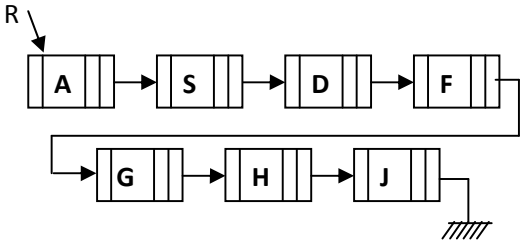

Figure 3.1.1: Creation of Link List with Sequence of the Data in Inorder

Step 3: The function b_tree is called with left child of P as argument; same process is applied on the link list (left child of P). Hence after finding the S, is similarly broken into left and right part. That is shown in figure. 3.1.3.

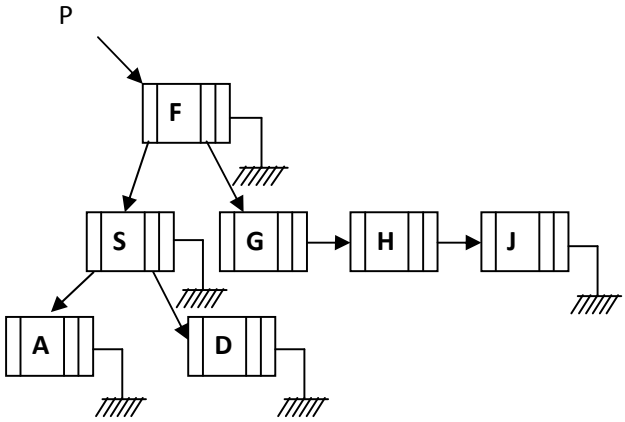

Figure 3.1.3: Creation of Left and Right Child of the Node, contained with S

Step 4: After applying the same things for right child of the node, containing F, finally we have got the binary tree, and its root node is assigned into ROOT. That is shown in figure. 3.1.4.

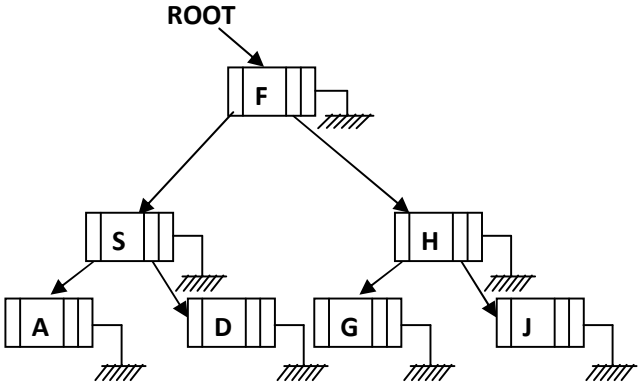

Figure 3.1.4: Final Stage of Creation of Binary Tree

Node	Row Value (R)	Column Value (C)	Position of the Node (R × r_gap, C × c_gap)
A	3	1	(6, 2)
S	2	2	(4, 4)
D	3	3	(6, 6)
F	1	4	(2, 8)
G	3	5	(6, 10)
H	2	6	(4, 12)
J	3	7	(6, 14)

3.2 Display of binary tree: To illustrate the algorithm 2.3 and 2.4, the process of displaying of a binary tree with tree like view is shown in this Section. Now the recently created binary tree, shows in figure. 3.2.1.

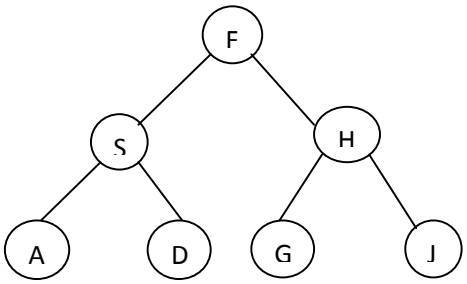

Figure 3.2.1: The Taken Binary

The inorder form of the binary tree is {A, S, D, F, G, H, J}. The column value of A node is 1 and for S, it is 2. Similarly we calculate column value for each node. Now row values for A and S node are calculated with its corresponding level that is 3 and 2. Similarly row values for other nodes are also calculated with its corresponding level. For this particular implementation, we assume that after calculation values of both, r_gap and c_gap are assigned with 2. Hence the position of the each node is calculated. That is shown in table 3.2.1.

Each node in binary tree is plotted according to its position, shown in table 3.2.1 in the computer screen or graph paper where X-axis is denoted as left to right direction and Y-axis is directed form top to bottom. This is shown in figure. 3.2.2.

After plotting each node of the tree according to its correct position, each pair of nodes having parent-child relationship is to be through a line. Due to that, we use a function, called "drawline", which takes root node of the binary tree, i.e. Anode, as argument for the first time. As this F-node has left child, S-node, a line is drawn between (2, 8) and (4, 4) which are respective position of F-node and S-node. That is shown in figure. 3.2.3.

Table 3.2.1: Position of Each Node of the Binary Tree

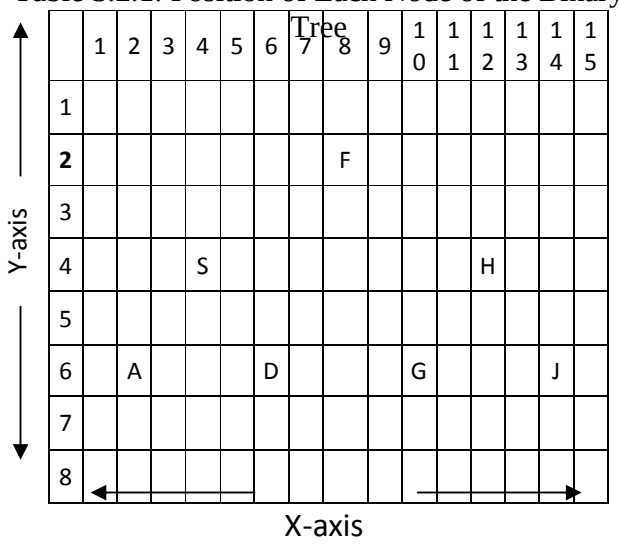

Figure 3.2.2: Plotting of Each Node of Binary Tree According to its Position

After creating F and S, through a recursively calling the function "drawline" with passing all nodes one by one, similarly left child for each node are individually searched. Then connecting each parent-child pair through drawing a line depending on their position, shown in table 3.1, the taken binary tree is being gotten a realistic tree like view, shown in fig. 3.2.4.

4. Result: In this section, the output of the tree, which is taken as an example in section 3, is displayed in figure 4.1. The inorder and preorder traversal of the tree are respectively {A, S, D, F, G, H, J} and {F, S, A, D, H, G, J} The program has been coded with C-Language.

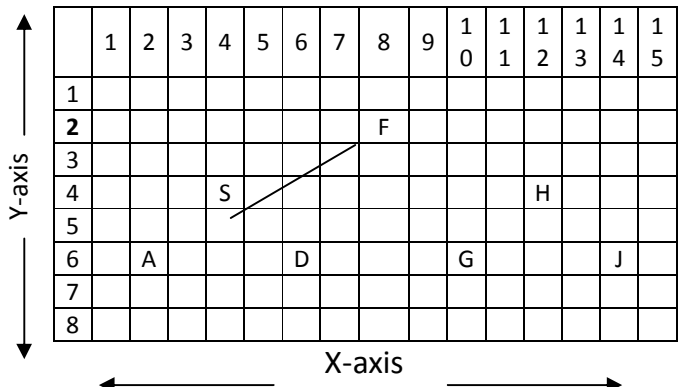

Figure 3.2.3:Connection between F-node and S-node through a Line According to its Position

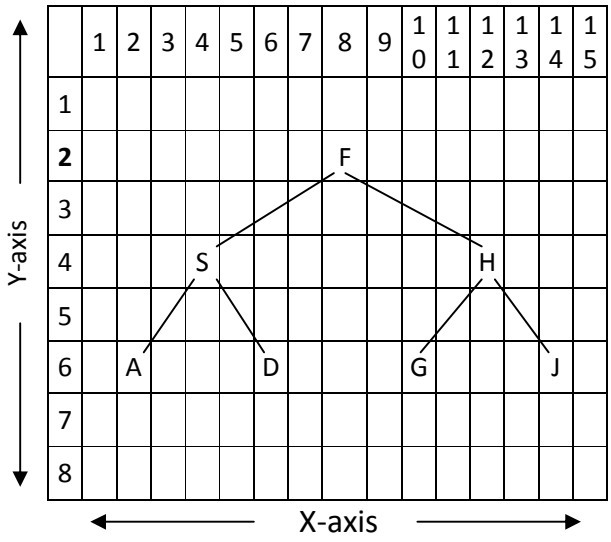

Figure 3.2.4: Complete Tree like View for the Taken Binary Tree

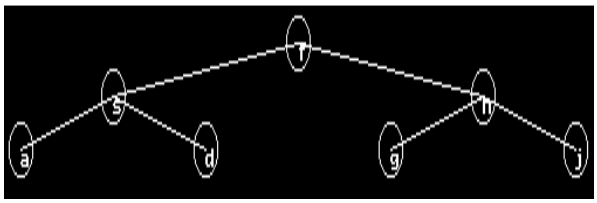

Figure 4.1

5. Analysis: Here we are physically, not virtually creating a binary tree in computer memory depending on preorder and inorder form. We are creating binary tree, actually through a special kind of link list. We use preorder of that binary tree only for traversing to find out the element, from which node link list is broken into its left and right child. Again the left child and right child are taken as two individual link lists, so this same process has to be applied on each and every node along with its left child and right child. Hence we are using recursion**.**

Here preorder is used to supply element one by one, that is searched in link list or incomplete binary tree which is under developing. For some wrong input either in preorder or inorder, after a certain stage element, supplied by preorder for searching, will not be found in the incomplete binary tree. Hence further, creation of binary tree will not be possible. Binary tree is created up to the position of correct input.

In computer we are habituate to display a binary tree either of preorder, postorder or inorder form. In this project we are also trying to display a binary tree with a realistic view. So for clear display of a tree like view, a gap between two printed nodes is to be calculated such that no two nodes are overlapped during the printing. For it, we have to calculate proper position of each node in a binary tree at the time of tree like displaying.

After taking a binary tree, the tree is traversed in inorder form for getting the column value of each node which is actually the position of the node in the inorder sequence and the level of each node is also calculated to assign row value. Here measuring and considering every thing, calculated gaps between two nodes along with X-axis and Y-axis are respectively kept into r_gap and c_gap. Hence proper position of every node is calculated through row and column value along with r_gap and c_gap such that tree is clearly displayed.

To display the Binary tree, at first, root node of the tree is connected with its two child-nodes through drawing lines. Then it is treated that these two child-nodes are individually also roots of two sub-trees. Making connection the same process is applied on these two sub-trees. Hence, a recursive function, “drawline” is used.

6. Conclusion: A binary tree can be constructed with only pen and paper from its inorder and preorder form. Apart from this, display of a binary tree in inorder and preorder form can be implemented trough coding, if the binary tree has already been created in computer memory. In spite of displaying the inorder, preorder or postorder form, a binary tree is displayed here with a realistic tree like view. At first, all nodes are placed in their proper positions such that no two nodes are overlapped each other. Finally, connecting each parent-child node through drawing line, the display of binary tree is completed with tree like view.

7. Reference:

[1]. Russell Impagliazzo, Lecture notes from algorithms courses 101 (Spring 2004; undergraduate), and 202 (Spring 2004, Fall 2004; graduate). University of California, San Diego. Used almost everywhere.

[2]. Fabulous Adventures In Coding, 21 July 2004. Used with permission, http://weblogs.asp.net/ericlippert/archive/2004/07/21/189974.aspx, Used in the backtracking and dynamic programming chapters.

[3]. Wikipedia, the free encyclopedia. http://www.wikipedia.org, Quoted/ appropriated pervasively.

[4]. C. A. R. Hoare. "Algorithm 63: Partition", "Algorithm 64: Quicksort", "Algorithm 65: Find", from Communications of the ACM. Volume 4, Issue 7 (July 1961), Page 321, ISSN: 0001-0782, http://doi.acm.org/10.1145/366622.366642

[5]. Yashavant Kanitker, *Graphics under C*, B P B Publication, 1998

[6]. Seymour Lipschutz, "Data Structure, Adopted by JAVPAI, TATA Mc GRAW HILL PUBLISHING CO. LTD.

[7]. Pranam Paul, Saurabh Dutta, A. K. Bhattacherjee, "Enhancement of Security through an Efficient Substitution-based Block Cipher of Bit-level Implementation with Possible Lossless Compression", International Journal of Computer Science and Network Security, Vol. 8 No. 4, pp 318 – 326.

[8]. R.S. Salaria, "A Simplified Text cum-workbok on Data Structure & Algorithm Using C", Theory Design and Implementation, second edition, KHANNA BOOK PUBLISHING CO.(P)LTD.

[9]. G.S.Baluja, "Data Structure Through C (A Practical Approach)", DHANPAT RAI &CO.(PVT.)LTD.EDUCATIONAL&TECHNICAL PUBLISHERS.

[10]. Pranam Paul, Saurabh Dutta, A. K. Bhattacherjee, "An Approach to ensure Security through Bit-level Encryption with Possible Lossless Compression", International Journal of Computer Science and Network Security, Vol. 8 No. 2, pp 291 – 299.