Name : Mufaddal Diwan
Class : SE-3
Roll no : 21340

ASSIGNMENT NO. 3

TITLE	Threaded binary tree
PROBLEM STATEMENT /DEFINITION	Convert given binary tree into inordered and preordered threaded binary tree. Analyze time and space complexity of the algorithm.
OBJECTIVE	To understand construction of inordered and preordered Threaded binary tree from given binary tree.
OUTCOME	At the end of this assignment students will able to construct threaded binary tree and able to perform basic operations on Threaded binary tree.
S/W PACKAGES AND HARDWARE	• (64-bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS
APPARATUS USED	• Programming Tools (64-Bit) Latest Open source update of Eclipse Programming frame work, TC++, GTK++.

Concepts related Theory:

Binary Tree: is a special data structure used for data storage purposes. A binary tree has a special condition that each node can have a maximum of two children. A binary tree has the benefits of both an ordered array and a linked list as search is as quick as in a sorted array and insertion or deletion operation are as fast as in linked list. A

binary tree is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers each point to a binary tree.

In a binary search tree, there are many nodes that have an empty left child or empty right child or both. You can utilize these fields in such a way so that the empty left child of a node points to its inorder predecessor and empty right child of the node points to its inorder successor

Threaded Binary Tree: A binary tree is threaded by making all right child pointers that would normally be null point to the inorder successor of the node (if it exists), and all left child pointers that would normally be null point to the inorder predecessor of the node**.**

It is also possible to discover the parent of a node from a threaded binary tree, without explicit use of parent pointers or a stack. This can be useful where stack space is limited, or where a stack of parent pointers is unavailable.

One way threading:-A thread will appear in the right field of a node and will point to the next node in the inorder traversal.

Two way threading:- A thread will also appear in the left field of a node and will point to the preceding node in the inorder traversal.

Types of threaded binary tree

Single threaded: Each node is threaded towards either the in-order predecessor or successor (left or right) means all right null pointers will point to inorder successor or all left null pointers will point to inorder predecessor

Double threaded: Each node is threaded towards both the in-order predecessor and successor (left and right) means all right null pointers will point to inorder successor AND all left null pointers will point to inorder predecessor.

Threaded binary search tree: Threaded binary search tree is BST in which all right pointers of node which point to NULL are changed and made to point to inorder successor current node (These are called as single threaded trees). In completely threaded tree (or double threaded trees), left pointer of node which points to NULL is made to point to inorder predecessor of current node if inorder predecessor exists.

Now, there is small thing needs to be taken care of. A right or left pointer can have now two meanings : One that it points to next real node, second it is pointing inorder successor or predecessor of node, that means it is creating a thread. To store this

information, we added a bool in each node, which indicates whether pointer is real or thread.

Approach:

1. we can do the inorder traversal and store it in some queue. Do another inorder traversal and where ever you find a node whose right reference is NULL , take the front element from the queue and make it the right of the current node.

2.Now we will see the solution which will convert binary tree into threaded binary tree in one single traversal with no extra space required.

- Do the reverse inorder traversal, means visit right child first.
- In recursive call, pass additional parameter, the node visited previously.
- whenever you will find a node whose right pointer is NULL and previous visited node is not NULL then make the right of node points to previous visited node and mark the boolean right threaded as true.
- Important point is whenever making a recursive call to right subtree, do not change the previous visited not and when making a recursive call to left subtree then pass the actual previous visited node.

Algorithm:

```
public void convert(Node root){

inorder(root, null);

}
public void inorder(Node root, Node previous){

f(root==null){

return;

}else{

inorder(root.right, previous);

if(root.right==null && previous!=null){

root.right = previous;

root.rightThread=true;

}
```

```
inorder(root.left, root);
}
}
public void print(Node root){
//first go to most left node
Node current = leftMostNode(root);
//now travel using right pointers
while(current!=null){
System.out.print(" " + current.data);
//check if node has a right thread
if(current.rightThread)
current = current.right;
current = current.right;
else // else go to left most node in the right
subtree current =
leftMostNode(current.right);
```

```
System.out.println()
}
public Node leftMostNode(Node node){
if(node==null){
return null;
}else{
while(node.left!=null)
node = node.left
}
return node;
}
}
```

Test-Cases

Description	Input	Output	Result
Create TBT (Enter -1 for no node)	5 4 -1 -1 7 6 -1 -1 8 -1 -1	-	Pass
Traverse TBT inorder	-	4 5 6 7 8	Pass
Traverse TBT preoder	-	5 4 7 6 8	Pass

Conclusion: After successfully completing this assignment, Students have learned construction of Threaded binary tree and various operations on Threaded binary tree.