Assignment No - 5 (Group B)

Problem statement:

Convert given binary tree into threaded binary tree. Analyze time and space complexity of the algorithm.

Pre-requisite

- Knowledge of C++ programming
- Knowledge of threaded binary tree in data structure

Objective

- Understanding the effective use of NULL pointers to improve the performance in tree traversal operations

Input

- Sequence of numbers to construct binary tree

Output

- Traversal of binary tree

Software and Hardware requirements:-

1. **Operating system:** Linux- Ubuntu 16.04 to 17.10, or Windows 7 to 10,
2. **RAM-** 2GB RAM (4GB preferable)
3. C++ / gcc compiler- / 64 bit Fedora, eclipse IDE

Theory-

Threaded binary search tree:

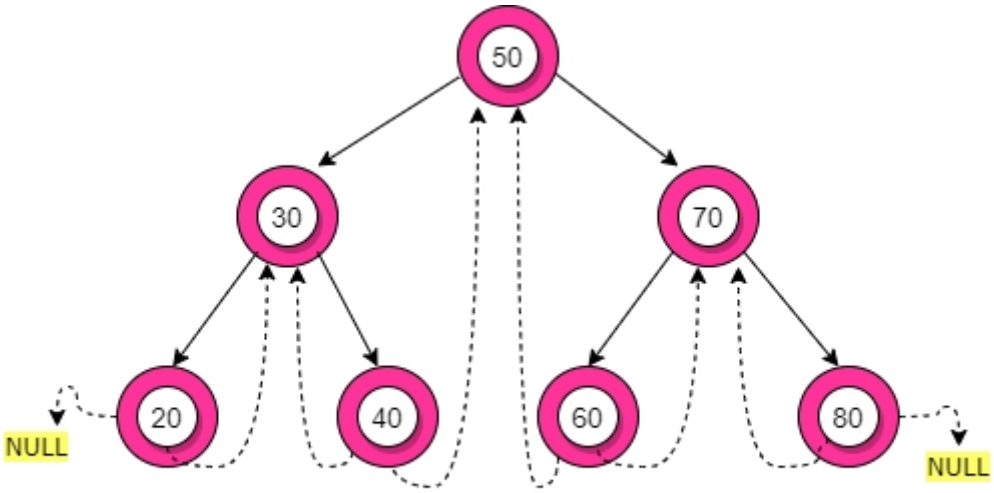
A threaded binary search tree is a binary search tree with additional links (threads) between nodes, optimizing in-order traversal by eliminating the need for recursion or stacks.

In a Threaded Binary Search Tree, the nodes will store the in-order predecessor/successor instead of storing NULL in the left/right child pointers.

In a threaded binary search tree for the nodes whose right pointer is null, we store the in-order successor of the node (if-exists), and for the nodes whose left pointer is null, we store the in-order predecessor of the node(if-exists).

One thing to note is that the leftmost and the rightmost child pointer of a tree always points to null as their in-order predecessor and successor do not exist.

To understand this, following is an example of a Threaded Binary search Tree.



Double Threaded Binary Tree

Why do we need Threaded Binary Search Tree?

Binary trees have a lot of wasted space: the leaf nodes each have 2 null pointers. We can use these pointers to help us in inorder traversals. Threaded binary search tree makes the tree traversal faster since we do not need stack or recursion for traversal.

Types of Threaded Binary Search tree:

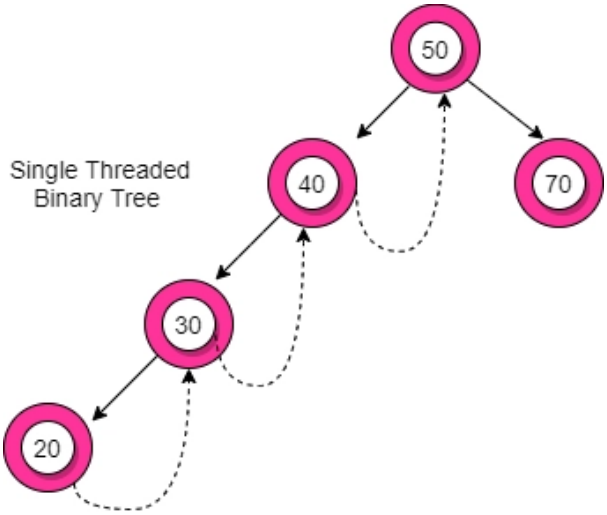
There are two types of Threaded Binary Search Trees:

- One-way Threaded Binary Search Tree
- Two-way Threaded Binary Search Tree

1. One-way Threaded Binary Search Tree

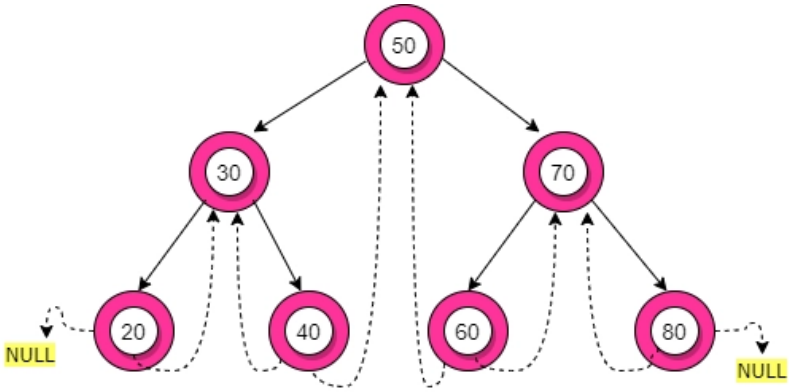
In this type, if a node has a right null pointer, then this right pointer is threaded towards the in-order successor's node if it exists.

The following diagram depicts an example of a Single-Threaded Binary Search Tree. Dotted lines represent threads.



2. Two-way Threaded Binary Search Tree

In this type, the left null pointer of a node is made to point towards the in-order predecessor node and the right null pointer is made to point towards the in-order successor node.



Double Threaded Binary Tree

Advantages of Threaded Binary Search Tree

1. **No stack or recursion needed :** Unlike binary trees, threaded binary trees do not require a stack or recursion for their traversal.
2. **Decreases memory wastage**. In normal binary trees, whenever a node's left/right pointer is NULL, memory is wasted. But with threaded binary trees, we are overcoming this problem by storing its inorder predecessor/successor.
3. **Time complexity:** In-order traversal in a threaded binary tree is fast because we get the next node in O(1) time as we have thread for inorder successor
4. **Backward traversal:** In a double-threaded binary tree, we can even do a backward traversal using threads to predecessor

Conclusion:

The need of implicit or explicit stack in the traversal operation was found eliminated

Sample Oral Questions :

1. Why do we need Threaded Binary Tree?
2. What are the Types of threaded binary trees?
3. What are the advantages of Threaded Binary Tree?