LAB SESSION 9: THREADED BINARY SEARCH TREE

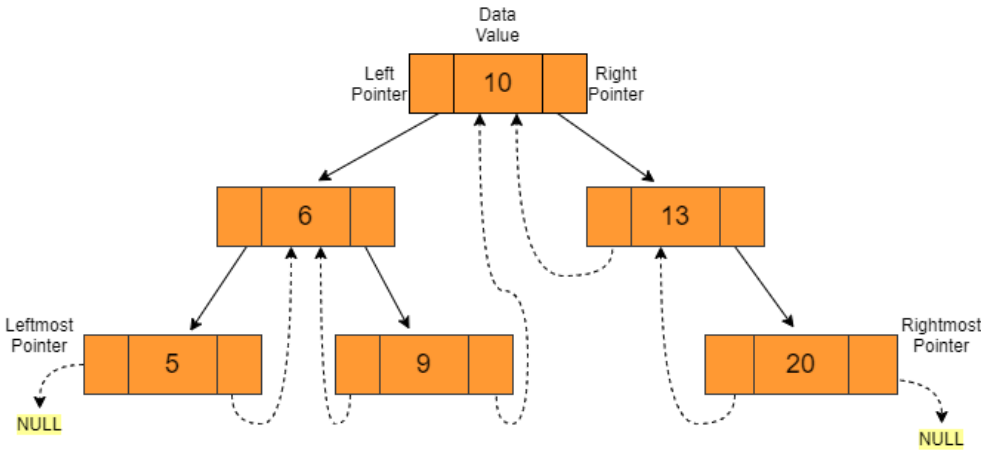
AIM: To implement THREADED BINARY SEARCH TREE and perform the listed operations on such trees.

PROBLEM DEFINITION:

Develop C program to create a Threaded BST through a series of insertions from a predefined array of elements. Provide the following options for the said operations to be performed on the tree:

1. Insertion of a new element
2. Deletion of an existing element
3. Searching for a given element
4. Performing inorder traversal on the tree.

THEORY: A Threaded Binary Tree is a variant of a normal Binary Tree that facilitates faster tree traversal and does not require a Stack or Recursion. It decreases the memory wastage by setting the null pointers of a leaf node to the in-order predecessor or in-order successor. In a Threaded Binary Tree, the nodes will store the in-order predecessor/successor instead of storing NULL in the left/right child pointers. So the basic idea of a threaded binary tree is that 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).



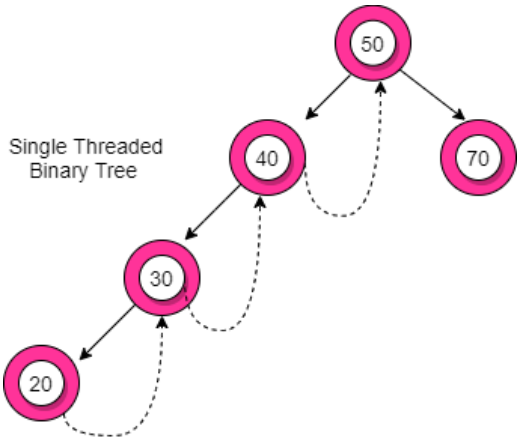
THREADED BINARY TREE

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.

The main idea behind setting such a structure is to make the inorder and preorder traversal of a binary tree faster without using any additional data structure (e.g. auxiliary stack) or memory for its traversal.

Types of Threaded Binary tree

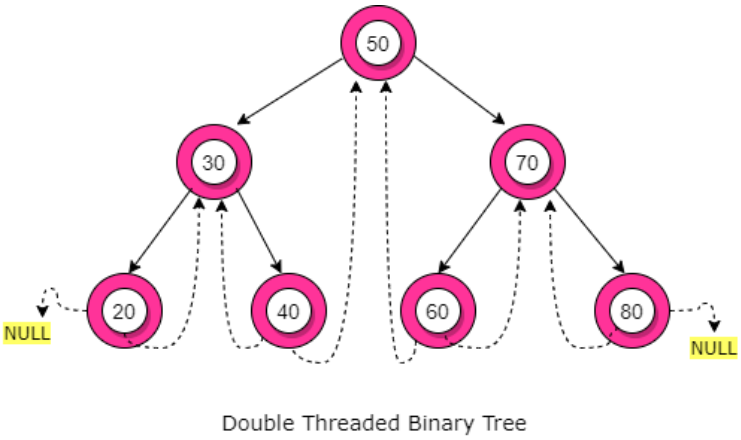
Single-Threaded Binary Tree



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.

Node Structure of Single-Threaded Binary Trees: The structure of a node in a binary threaded tree is quite similar to that of a binary tree, but with some modifications. In threaded binary trees, we need to use extra boolean variables in the node structure. For single-threaded binary trees, we use only the rightThread variable.

2. Double-Threaded Binary Tree



Double Threaded Binary 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. Here, the leftThread and rightThread boolean variables help us to differentiate whether the left/right pointer stores the in-order predecessor/successor or left child/right child.

Advantages of Threaded Binary Tree

- *No need for stacks or recursion*: Unlike binary trees, threaded binary trees do not require a stack or recursion for their traversal.
- *Optimal memory usage:* Another advantage of threaded binary tree data structure is that it 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.
- *Time complexity:* In-order traversal in a threaded binary tree is fast because we get the next node in O(1) time than a normal binary tree that takes O(Height). But insertion and deletion operations take more time for the threaded binary tree.
- *Backward traversal:* In a double-threaded binary tree, we can even do a backward traversal.

Disadvantages of Threaded Binary tree

- *Complicated insertion and deletion*: By storing the inorder predecessor/ successor for the node with a null left/right pointer, we make the insertion and deletion of a node more time-consuming and a highly complex process.
- *Extra memory usage*: We use additional memory in the form of rightThread and leftThread variables to distinguish between a thread from an ordinary link. (However, there are more efficient methods to differentiate between a thread and an ordinary link).

ALGORITHM AND DEMONSTRATION:

1. **Insertion in a Threaded Binary Search Tree**
2. **Deletion from a Threaded Binary Search Tree**