**Assignment No. 02**

Name:- Anirudha Shripad Kale.

Roll No. :- 223029

PRN No :- 22210288

Batch: C1

**Problem Statement:** Implement threaded binary search tree and perform Inorder traversal of that.

**Objectives:**

1. To understand the concept of Threaded Binary Search Tree.
2. To efficiently traverse the tree in sorted order without using recursion or maintaining an explicit stack.
3. To study types of threaded binary tree and how they are efficient for inorder traversal.

**Theory:**

A threaded binary tree is a type of binary tree data structure where the empty left and right child pointers in a binary tree are replaced with threads that link nodes directly to their in-order predecessor or successor, thereby providing a way to traverse the tree without using recursion or a stack.

Threaded binary trees can be useful when space is a concern, as they can eliminate the need for a stack during traversal. However, they can be more complex to implement than standard binary trees.

There are two types of threaded binary trees.

Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

The threads are also useful for fast accessing ancestors of a node.

static class Node

{

int data;

Node left, right;

boolean rightThread, leftThread;

}

**Algorithm:**

1. **Insert Method Algorithm**:
   * Create a new node with the given data.
   * Traverse the tree:
     + If data is less than the current node, move to the left child.
       - If left child is null, insert new node as left child with a thread pointing to its parent.
     + If data is greater than or equal to the current node:
       - If the current node has a thread, insert new node as its right child, updating threads accordingly.
       - Otherwise, move to the right child and continue traversing.
2. **Main Method Algorithm**:
   * Create a scanner object for user input.
   * Create an instance of the single-threaded binary tree.
   * Display a menu:
     + Option to insert data: Continuously insert data until -1 is entered.
     + Option to print inorder traversal.
     + Option to exit the program.
3. **End of Algorithm**:
   * The program exits when the user chooses to exit.

**Input(Source Code):**

import java.util.\*;

class Node {

    int data;

    Node left, right;

    boolean rightThread;

    public Node(int data) {

        this.data = data;

        this.left = right = null;

        this.rightThread = false;

    }

}

public class Assignment2 {

    Node root = null;

    // Function to perform inorder traversal

    public void inorder() {

        Node current = leftMost(root);

        while (current != null) {

            System.out.print(current.data + " ");

            if (current.rightThread)

                current = current.right;

            else

                current = leftMost(current.right);

        }

    }

    // Function to find the leftmost node in a subtree

    private Node leftMost(Node node) {

        if (node == null)

            return null;

        while (node.left != null)

            node = node.left;

        return node;

    }

    // Function to insert a node in the binary tree

    public void insert(int data) {

        Node newNode = new Node(data);

        if (root == null) {

            root = newNode;

            return;

        }

        Node current = root;

        while (true) {

            if (data < current.data) {

                if (current.left == null) {

                    current.left = newNode;

                    newNode.rightThread = true;

                    newNode.right = current;

                    return;

                }

                current = current.left;

            } else {

                if (current.rightThread) {

                    newNode.right = current.right;

                    current.right = newNode;

                    current.rightThread = false;

                    newNode.rightThread = true;

                    return;

                }

                if (current.right == null) {

                    current.right = newNode;

                    return;

                }

                current = current.right;

            }

        }

    }

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        Assignment2 tree1 = new Assignment2(); // Assuming initial root data as 0

        System.out.println("\t\*\*\*\*\*Threaded BST operations\*\*\*\*\*\t");

        System.out.println("\n-----------------------------------------");

        boolean x = true;

        while (x) {

            System.out.println("Operations Available : ");

            System.out.println(

                    "1. Insert the data to TBT \n2. Inorder print \n3. Exit");

            System.out.println("First insert the data then perform another operations.");

            System.out.println("\n-----------------------------------------");

            System.out.print("Enter your choice : ");

            int choice = sc.nextInt();

            if (choice < 1 || choice > 3) {

                System.out.println("Please, Enter the choice between 1 to 3.");

            }

            switch (choice) {

                case 1:

                    System.out.print("Enter the data (-1 to stop) : ");

                    int data = sc.nextInt();

                    while (data != -1) {

                        tree1.insert(data);

                        data = sc.nextInt();

                    }

                    System.out.println("The data get successfully inserted.");

                    System.out.println("\n-----------------------------------------");

                    break;

                case 2:

                    System.out.println("The inorder traversal is : ");

                    tree1.inorder();

                    System.out.println("\n-----------------------------------------");

                    break;

                case 3:

                    x = false;

                    break;

            }

        }

    }

}

**Output:**

\*\*\*\*\*Threaded BST operations\*\*\*\*\*

-----------------------------------------

Operations Available :

1. Insert the data to TBT

2. Inorder print

3. Exit

First insert the data then perform another operations.

-----------------------------------------

Enter your choice : 1

Enter the data (-1 to stop) : 2 6 5 9 8 7 4 1 25 65 13 -1

The data get successfully inserted.

-----------------------------------------

Operations Available :

1. Insert the data to TBT

2. Inorder print

3. Exit

First insert the data then perform another operations.

-----------------------------------------

Enter your choice : 2

The inorder traversal is :

1 2 4 5 6 7 8 9 13 25 65

-----------------------------------------

Operations Available :

1. Insert the data to TBT

2. Inorder print

3. Exit

First insert the data then perform another operations.

-----------------------------------------

Enter your choice : 3