# DATA STRUCTURES AND ALGORITHMS LABORATORY

Group B

Assignment No. 3

NAME :- OJUS PRAVIN JAISWAL
ROLL NO. :- SACO19108
DIVISION :- A

**DATA STRUCTURE LABORATORY**

# Group B

# Assignment 3

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

**Objectives:**

- To explain the concept of Threaded Binary Tree.

**Outcome:**

- Implement Threaded Binary Tree.

**Theory:**

The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

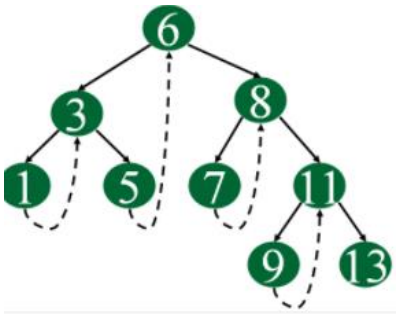
There are two types of threaded binary trees.

**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.

Following diagram shows an example Single Threaded Binary Tree. The dotted lines represent threads.




How to convert a Given Binary Tree to Threaded Binary Tree?

The idea is based on the fact that we link from inorder predecessor to a node. We link those inorder predecessor which lie in subtree of node. So we find inorder predecessor of a node if its left is not

NULL. Inorder predecessor of a node (whose left is NULL) is a rightmost node in the left child. Once we find the predecessor, we link a thread from it to the current node.

**Software Required:** g++ / gcc compiler- / 64 bit Fedora, eclipse IDE

**Input:** Binary Tree

**Program:**

```
//============================================================================
// Name        : ThreadedBinaryTree.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : C++ program to convert a Binary Tree to Threaded Tree
//============================================================================

#include <iostream>
using namespace std;

/* Structure of a node in threaded binary tree */
struct Node
{
	int key;
	Node *left, *right;

	// Used to indicate whether the right pointer
	// is a normal right pointer or a pointer
	// to inorder successor.
	bool isThreaded;
};

// Converts tree with given root to threaded
// binary tree.
// This function returns rightmost child of
// root.
Node *createThreaded(Node *root)
{
	// Base cases : Tree is empty or has single node
	if (root == NULL)
		return NULL;
	if (root->left == NULL &&
		root->right == NULL)
		return root;

	// Find predecessor if it exists
	if (root->left != NULL)
	{
		// Find predecessor of root (Rightmost child in left subtree)
		Node *l = createThreaded(root->left);

		// Link a thread from predecessor to root.
		l->right = root;
		l->isThreaded = true;
	}
```

```
	// If current node is rightmost child
	if (root->right == NULL)
		return root;

	// Recur for right subtree.
	return createThreaded(root->right);
}

// A utility function to find leftmost node in a binary tree rooted with 'root'.
// This function is used in inOrder()
Node *leftMost(Node *root)
{
	while (root != NULL && root->left != NULL)
		root = root->left;
	return root;
}

// Function to do inorder traversal of a threaded binary tree
void inOrder(Node *root)
{
	if (root == NULL)
		return;

	// Find the leftmost node in Binary Tree
	Node *cur = leftMost(root);

	while (cur != NULL)
	{
		cout << cur->key << " ";

		// If this Node is a thread Node, then go to inorder successor
		if (cur->isThreaded)
			cur = cur->right;

		else // Else go to the leftmost child in right subtree
			cur = leftMost(cur->right);
	}
}

// A utility function to create a new node
Node *newNode(int key)
{
	Node *temp = new Node;
	temp->left = temp->right = NULL;
	temp->key = key;
	return temp;
}

// Driver program to test above functions
int main()
{
	/*	1
		/	\
		2	3
		/ \	/ \
		4	5 6	7 */
	cout << "\n----------THREADED BINARY TREE----------\n";
	int e, c;
```

```
	cout << "\nInsert root node : ";
	cin >> e;
	Node *root = newNode(e);
	Node *parent = root;
	while (1)
	{
		cout << "\n-----Menu-----\n0) Go to root node\n1) Change parent node
to left child\n2) Change parent node to right child\n3) Insert left child\n4)
Insert right child\n5) Inorder traversal of created threaded trees\n6) Exit
program\n";
		cout << "\nEnter your choice : ";
		cin >> c;
		switch (c)
		{
		case 0:
			parent = root;
			cout << "\nParent node changed to root node\n";
			break;
		case 1:
			parent = parent->left;
			cout << "\nParent node changed to left child\n";
			break;
		case 2:
			parent = parent->right;
			cout << "\nParent node changed to right child\n";
			break;
		case 3:
			cout << "\nEnter the left child of parent node : ";
			cin >> e;
			parent->left = newNode(e);
			break;
		case 4:
			cout << "\nEnter the right child of parent node : ";
			cin >> e;
			parent->right = newNode(e);
			break;
		case 5:
			createThreaded(root);
			cout << "Inorder traversal of created threaded tree is : ";
			inOrder(root);
			cout<<"\n";
			break;
		case 6:
			cout << "\nExitting Program!!!\n";
			exit(0);
		default:
			cout << "\nWrong choice entered!!!\n";
		}
	}
	return 0;
}
```

**Output:**

```
----------THREADED BINARY TREE----------

Insert root node : 1

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 3

Enter the left child of parent node : 2

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 4

Enter the right child of parent node : 3

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 1

Parent node changed to left child

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 3

Enter the left child of parent node : 4

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program
```

```
Enter your choice : 4

Enter the right child of parent node : 5

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 0

Parent node changed to root node

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 2

Parent node changed to right child

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 3

Enter the left child of parent node : 6

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 4

Enter the right child of parent node : 7

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 5
Inorder traversal of created threaded tree is : 4 2 5 1 6 3 7
```

```
-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 7

Wrong choice entered!!!

-----Menu-----
0) Go to root node
1) Change parent node to left child
2) Change parent node to right child
3) Insert left child
4) Insert right child
5) Inorder traversal of created threaded trees
6) Exit program

Enter your choice : 6

Exitting Program!!!

[Program finished]
```

**Conclusion:** This program implements Threaded Binary tree data structure