

MIT WORLD PEACE UNIVERSITY

Advanced Data Structures
Second Year B. Tech, Semester 4

IMPLEMENTATION OF THREADED BINARY TREE
AND ITS TRAVERSALS

ASSIGNMENT NO. 4

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

March 26, 2023

Contents

1 Objectives	1
2 Problem Statement	1
3 Theory	1
3.1 Threaded Binary Tree	1
3.1.1 What is Threaded Binary Tree?	1
3.1.2 Advantages of Threaded Binary Tree	1
3.2 Space Utilization in Threaded Binary Tree	2
4 Platform	2
5 Input	2
6 Output	2
7 Test Conditions	2
8 Pseudo Code	2
8.1 Create	2
8.2 Inorder Traversal	3
8.3 PreOrder Traversal	4
9 Time Complexity	4
9.1 Creation of Threaded Binary Tree	4
9.2 Inorder Traversal	4
9.3 Preorder Traversal	4
10 Code	4
10.1 Program	4
10.2 Input and Output	7
11 Conclusion	9
12 FAQ	10

1 Objectives

1. To study the data Structure : Threaded Binary Tree
2. To study the advantages of Threaded Binary Tree over Binary Tree

2 Problem Statement

Implement threaded binary tree and perform inorder traversal.

3 Theory

3.1 Threaded Binary Tree

3.1.1 What is Threaded Binary Tree?

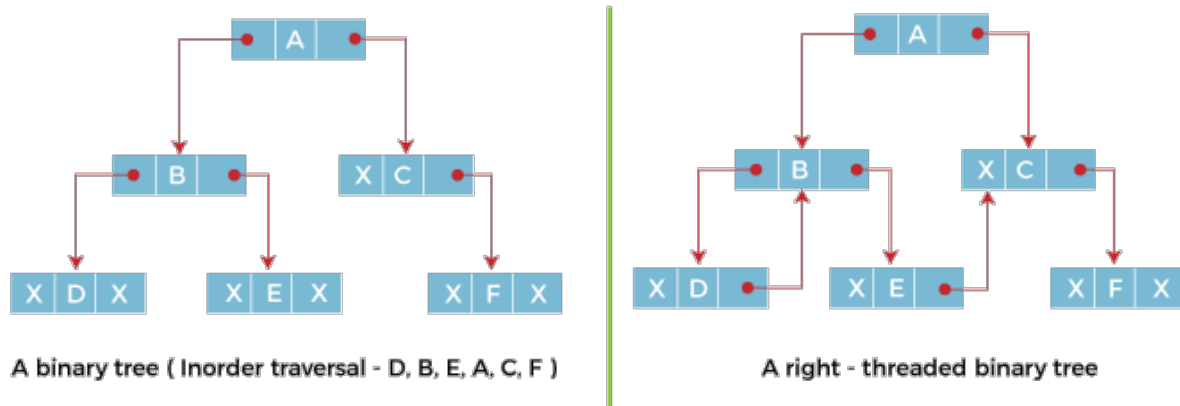


Figure 1: Threaded and a Normal Binary Tree

A Threaded Binary Tree is a binary tree where every node has an extra bit, and this bit is used to indicate whether the right pointer points to the right child or to the inorder successor of the node. In this way, we can traverse the tree without using the stack and without recursion. A Threaded Binary Tree is a binary tree where every node has an extra bit, and this bit is used to indicate whether the right pointer points to the right child or to the inorder successor of the node. In this way, we can traverse the tree without using the stack and without recursion.

3.1.2 Advantages of Threaded Binary Tree

1. Threaded Binary Tree is used to traverse the tree without using the stack and without recursion.
2. Inorder traversal of a binary tree is the most frequently used traversal. Inorder traversal of a threaded binary tree is done without using the stack and without recursion easier than the normal binary tree.

3.2 Space Utilization in Threaded Binary Tree

Space Utilized by a Normal Tree A Normal Binary Tree with N nodes each has data, left and right pointers. So, the space utilized by a normal binary tree is $3N$.

Space Utilized by a Threaded Tree

A Threaded Binary Tree with N nodes each has data, left and right pointers and 2 bits. So, the space utilized by a threaded binary tree is $3N + 2N = 5N$.

But they avoid the usage of a stack during traversal. So, the space utilized by a threaded binary tree is less than the space utilized by a normal binary tree.

4 Platform

Operating System: Arch Linux x86-64

IDEs or Text Editors Used: Visual Studio Code

Compilers : g++ and gcc on linux for C++

5 Input

1. Input at least 10 nodes.
2. Display inorder traversal of binary tree with 10 nodes.

6 Output

1. The traversal of the Threaded binary tree in different ways.

7 Test Conditions

1. Input at least 10 nodes.
2. Display all traversals of binary tree with 10 nodes.(recursive and nonrecursive)

8 Pseudo Code

8.1 Create

```
1 // Allocate memory for root
2 root = new ThreadedBinaryTreeNode()
3
4 // Assign root->leftc and rightc to head
5 root->left = head
6 root->right = head
7
8 // Accept root data
9 print "Enter the root data: "
10 cin >> root->data
11
```

```
12 // Assign head lbit to 0
13 head->left = root
14 // Assign head->leftc to root
15 head->isLeftNodeAThread = false
16
17 ThreadedBinaryTreeNode *temp, *curr
18 temp = root
19 bool flag = true
20 int again = 0
21 int choice = 0
22
23 print "Do you want to enter another node? (1 or 0)" << endl
24 cin >> again
25 while (again != 0)
26     temp = root
27     flag = true
28     print "Enter 1 for Entering a new node to the Left, and 2 for Entering it to
the right" << endl
29     cin >> choice
30     while (flag)
31         if (choice == 1)
32             if (temp->isLeftNodeAThread == true)
33                 curr = new ThreadedBinaryTreeNode()
34                 print "Enter the Data: "
35                 cin >> curr->data
36                 curr->right = temp
37                 temp->left = curr
38                 temp->isLeftNodeAThread = false
39                 flag = false
40             else
41                 temp = temp->left
42
43             else if (choice == 2)
44                 if (temp->isRightNodeAThread == true)
45                     curr = new ThreadedBinaryTreeNode()
46                     print "Enter the Data: "
47                     cin >> curr->data
48                     curr->left = temp
49                     curr->right = temp->right
50                     temp->right = curr
51                     temp->isRightNodeAThread = false
52                     flag = false
53                 else
54                     temp = temp->right
55
56     print "Do you want to enter another node? (1 or 0)" << endl
57     cin >> again
```

8.2 Inorder Traversal

```
1 inorder_traversal()
2     ThreadedBinaryTreeNode *temp
3     temp = head
4     while (true)
5         temp = inorder_successor(temp)
6         if (temp == head)
7             break
8     print temp->data << " "
```

```
9
10 ThreadedBinaryTreeNode *inorder_successor(ThreadedBinaryTreeNode *temp)
11     ThreadedBinaryTreeNode *x = temp->right
12     if (!temp->isRightNodeAThread)
13         while (x->isLeftNodeAThread == false)
14             x = x->left
15     return x
```

8.3 PreOrder Traversal

```
1 void preorder_traversal()
2     ThreadedBinaryTreeNode *temp = head->left
3     while (temp != head)
4         print temp->data << " "
5         while (temp->isLeftNodeAThread == false)
6             temp = temp->left
7         print temp->data << " "
8         while (temp->isRightNodeAThread == true)
9             temp = temp->right
10        temp = temp->right
```

9 Time Complexity

9.1 Creation of Threaded Binary Tree

- Time Complexity: $O(n)$
- Space Complexity: $O(n)$

9.2 Inorder Traversal

- Time Complexity: $O(n)$
- Space Complexity: $O(1)$

9.3 Preorder Traversal

- Time Complexity: $O(n)$
- Space Complexity: $O(1)$

10 Code

10.1 Program

```
1 #include <iostream>
2 #include <queue>
3 #include <stack>
4 #include <string.h>
5 using namespace std;
6
7 class ThreadedBinaryTreeNode
8 {
```

```
9     string data;
10     ThreadedBinaryTreeNode *left;
11     ThreadedBinaryTreeNode *right;
12     bool isLeftNodeAThread;
13     bool isRightNodeAThread;
14
15 public:
16     ThreadedBinaryTreeNode()
17     {
18         left = NULL;
19         right = NULL;
20         isLeftNodeAThread = true;
21         isRightNodeAThread = true;
22     }
23     friend class ThreadedBinaryTree;
24 };
25
26 class ThreadedBinaryTree
27 {
28 public:
29     ThreadedBinaryTreeNode *head;
30     ThreadedBinaryTreeNode *root;
31     ThreadedBinaryTree()
32     {
33         head = new ThreadedBinaryTreeNode();
34         head->left = head;
35         head->right = head;
36         head->isLeftNodeAThread = false;
37         head->isRightNodeAThread = false;
38         root = NULL;
39     }
40
41     void create()
42     {
43         // Allocate memory for root;
44         root = new ThreadedBinaryTreeNode();
45
46         // Assign root->leftc and rightc to head;
47         root->left = head;
48         root->right = head;
49
50         // Accept root data;
51         cout << "Enter the root data: ";
52         cin >> root->data;
53
54         // Assign head lbit to 0;
55         head->left = root;
56         // Assign head->leftc to root;
57         head->isLeftNodeAThread = false;
58
59         ThreadedBinaryTreeNode *temp, *curr;
60         temp = root;
61         bool flag = true;
62         int again = 0;
63         int choice = 0;
64
65         cout << "Do you want to enter another node? (1 or 0)" << endl;
66         cin >> again;
67         while (again != 0)
```

```
68     {
69         temp = root;
70         flag = true;
71
72         cout << "Enter 1 for Entering a new node to the Left, and 2 for
Entering it to the right" << endl;
73         cin >> choice;
74         while (flag)
75         {
76             if (choice == 1)
77             {
78                 if (temp->isLeftNodeAThread == true)
79                 {
80                     curr = new ThreadedBinaryTreeNode();
81                     cout << "Enter the Data: ";
82                     cin >> curr->data;
83                     curr->right = temp;
84                     temp->left = curr;
85                     temp->isLeftNodeAThread = false;
86                     flag = false;
87                 }
88                 else
89                 {
90                     temp = temp->left;
91                 }
92             }
93             else if (choice == 2)
94             {
95                 if (temp->isRightNodeAThread == true)
96                 {
97                     curr = new ThreadedBinaryTreeNode();
98                     cout << "Enter the Data: ";
99                     cin >> curr->data;
100                    curr->left = temp;
101                    curr->right = temp->right;
102                    temp->right = curr;
103                    temp->isRightNodeAThread = false;
104                    flag = false;
105                }
106                else
107                {
108                    temp = temp->right;
109                }
110            }
111        }
112        cout << "Do you want to enter another node? (1 or 0)" << endl;
113        cin >> again;
114    }
115 }
116 void inorder_traversal()
117 {
118     ThreadedBinaryTreeNode *temp;
119     temp = head;
120     while (true)
121     {
122         temp = inorder_successor(temp);
123         if (temp == head)
124             break;
125         cout << temp->data << " ";
```



```
126     }
127 }
128 ThreadedBinaryTreeNode *inorder_successor(ThreadedBinaryTreeNode *temp)
129 {
130     ThreadedBinaryTreeNode *x = temp->right;
131     if (!temp->isRightNodeAThread)
132     {
133         while (x->isLeftNodeAThread == false)
134             x = x->left;
135     }
136     return x;
137 }
138
139 void preorder_traversal()
140 {
141     ThreadedBinaryTreeNode *temp = head->left;
142     while (temp != head)
143     {
144         cout << temp->data << " ";
145         while (temp->isLeftNodeAThread == false)
146         {
147             temp = temp->left;
148             cout << temp->data << " ";
149         }
150         while (temp->isRightNodeAThread == true)
151         {
152             temp = temp->right;
153         }
154         temp = temp->right;
155     }
156 }
157 };
158
159 int main()
160 {
161     ThreadedBinaryTree tree;
162     tree.create();
163     cout<<"The inorder traversal is:" << endl;
164     tree.inorder_traversal();
165     cout << endl;
166     cout << "The preorder traversal is:" << endl;
167     tree.preorder_traversal();
168 }
```

10.2 Input and Output

```
1 Enter the root data: 1
2 Do you want to enter another node? (1 or 0)
3 1
4 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
5 1
6 Enter the Data: 3
7 Do you want to enter another node? (1 or 0)
8 1
9 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
10 1
11 Enter the Data: 5
12 Do you want to enter another node? (1 or 0)
13 1
```

Advanced Data Structures - Assignment 4

```
14 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
15 2
16 Enter the Data: 69
17 Do you want to enter another node? (1 or 0)
18 1
19 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
20 1
21 Enter the Data: 544
22 Do you want to enter another node? (1 or 0)
23 1
24 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
25 1
26 Enter the Data: 245
27 Do you want to enter another node? (1 or 0)
28 1
29 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
30 2
31 Enter the Data: 36
32 Do you want to enter another node? (1 or 0)
33 1
34 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
35 1
36 Enter the Data: 41
37 Do you want to enter another node? (1 or 0)
38 1
39 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
40 1
41 Enter the Data: 255
42 Do you want to enter another node? (1 or 0)
43
44 1
45 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
46 2
47 Enter the Data: 64
48 Do you want to enter another node? (1 or 0)
49 1
50 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
51 1
52 Enter the Data: 21
53 Do you want to enter another node? (1 or 0)
54 1
55 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
56 1
57 Enter the Data: 47
58 Do you want to enter another node? (1 or 0)
59 1
60 Enter 1 for Entering a new node to the Left, and 2 for Entering it to the right
61 2
62 Enter the Data: 65
63 Do you want to enter another node? (1 or 0)
64 0
65 The inorder traversal is:
66 47 21 255 41 245 544 5 3 1 69 36 64 65
67 The preorder traversal is:
68 1 3 5 544 245 41 255 21 47 69 36 64 65
```

11 Conclusion

Thus, implemented threaded binary tree with inorder traversal.

12 FAQ

1. Why TBT can be traversed without stack?

A threaded binary tree is a binary tree that has additional links between nodes to allow for traversal without the use of a stack or recursion. These extra links, called threads, connect nodes that would otherwise be null pointers in a traditional binary tree.

There **are two types of threaded binary trees**: inorder threaded binary trees and preorder threaded binary trees.

- (a) In an inorder threaded binary tree, the threads are used to connect nodes to their inorder predecessor and inorder successor. These threads allow for a traversal of the tree in inorder without using a stack or recursion. To traverse the tree in inorder without a stack, you start at the root node and follow the leftmost thread until you reach a node without a left child. Then, you visit that node and follow the thread to its inorder successor. You repeat this process until you have visited all nodes in the tree.
- (b) In a preorder threaded binary tree, the threads are used to connect nodes to their preorder successor. These threads allow for a traversal of the tree in preorder without using a stack or recursion. To traverse the tree in preorder without a stack, you start at the root node and visit each node in the tree in preorder. When you visit a node, you follow the thread to its preorder successor and visit that node next.
- (c) In both cases, the threads provide a way to traverse the tree without using a stack or recursion, which can be useful in situations where memory usage is a concern or where recursion or stack-based algorithms are not allowed. However, constructing the threads requires additional memory and processing time, so the benefits of threaded binary trees depend on the specific use case.

2. What are the advantages and disadvantages of TBT?

Threaded binary trees have several advantages and disadvantages that should be considered when deciding whether to use them in a specific scenario. Here are some of the main advantages and disadvantages:

Advantages:

- (a) **Traversal without stack or recursion:** As mentioned before, threaded binary trees allow for traversal without using a stack or recursion. This can be useful in situations where memory usage is a concern or where recursion or stack-based algorithms are not allowed.
- (b) **Efficient Inorder and Preorder Traversal:** The use of threads can significantly improve the performance of inorder and preorder traversal. In fact, threaded binary trees can perform these traversals in $O(n)$ time complexity, which is faster than the $O(n \log n)$ complexity of the recursive approach.
- (c) **Space Efficiency:** Compared to traditional binary trees, threaded binary trees require less space to store the threads. This can be useful in situations where memory usage is a concern.

Disadvantages:

- (a) **Construction Overhead:** The process of constructing threads can be time-consuming and requires additional memory. In some cases, the overhead of constructing the threads can negate any performance benefits.
- (b) **Complexity:** Threaded binary trees can be more complex than traditional binary trees due to the additional links. This can make them harder to understand and maintain.
- (c) **Limited Applications:** Threaded binary trees are not suitable for all types of problems. They are typically used in scenarios where traversal performance is critical, but in other cases, traditional binary trees or other data structures may be more appropriate.

3. Write application of TBT

- (a) **Expression Trees:** Threaded binary trees can be used to represent arithmetic expressions in a compact form. By using threads, the expression tree can be traversed efficiently without using a stack or recursion. This can be useful in compilers and other applications that deal with arithmetic expressions.
- (b) **Database Indexing:** Threaded binary trees can be used to implement indexing in databases. By using threads, the binary tree can be traversed efficiently, which can improve the performance of database queries.
- (c) **Text Editors:** Threaded binary trees can be used in text editors to efficiently search for words in a document. By using threads, the binary tree can be traversed efficiently, which can improve the speed of searches.
- (d) **Spell Checking:** Threaded binary trees can be used in spell checking applications to efficiently search for misspelled words. By using threads, the binary tree can be traversed efficiently, which can improve the speed of spell checking.
- (e) **Image Processing:** Threaded binary trees can be used in image processing applications to efficiently process pixels in an image. By using threads, the binary tree can be traversed efficiently, which can improve the speed of image processing algorithms.