

**Assignment-10**

**Name: - Mann Kothari**

**Reg. No.: - 22BCB7064**

**Course Title: - Design and Analysis of Algorithm (Embedded Lab)**

**Course Code: - CSE3023**

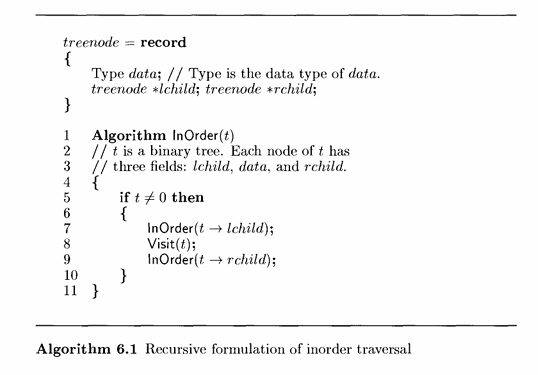
**Slot: - L21+L22**

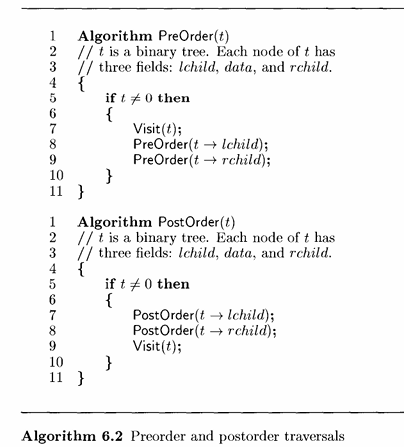
**Submitted to: - Prof. Tanikella Divya Naga Pavani**

* **INDEX**
* Tree Traversal
* Algorithm
* Code
* Output
* Complexity
* Topological Sorting
* Algorithm
* Code
* Output
* Complexity
* Complexity Chart

1. Write a C/C++ program to Implement Tree traversals

Algorithm: -





Time Complexity: -

O(n)

Code: -

#include <iostream>

using namespace std;

// A binary tree node has data, pointer to left child

// and a pointer to right child

struct Node {

    int data;

    struct Node \*left, \*right;

};

// Utility function to create a new tree node

Node\* newNode(int data)

{

    Node\* temp = new Node;

    temp->data = data;

    temp->left = temp->right = NULL;

    return temp;

}

// Given a binary tree, print its nodes in inorder

void printInorder(struct Node\* node)

{

    if (node == NULL)

        return;

    // First recur on left child

    printInorder(node->left);

    // Then print the data of node

    cout << node->data << " ";

    // Now recur on right child

    printInorder(node->right);

}

// Given a binary tree, print its nodes in preorder

void printPreorder(struct Node\* node)

{

    if (node == NULL)

        return;

    // First print data of node

    cout << node->data << " ";

    // Then recur on left subtree

    printPreorder(node->left);

    // Now recur on right subtree

    printPreorder(node->right);

}

// Given a binary tree, print its nodes according to the

// "bottom-up" postorder traversal.

void printPostorder(struct Node\* node)

{

    if (node == NULL)

        return;

    // First recur on left subtree

    printPostorder(node->left);

    // Then recur on right subtree

    printPostorder(node->right);

    // Now deal with the node

    cout << node->data << " ";

}

// Driver code

int main()

{

    struct Node\* root = newNode(1);

    root->left = newNode(2);

    root->right = newNode(3);

    root->left->left = newNode(4);

    root->left->right = newNode(5);

    root->right->left = newNode(6);

    root->right->right = newNode(7);

    // Function call

    cout << "Inorder traversal of binary tree is \n";

    printInorder(root);

    cout<<endl;

    cout << "Preorder traversal of binary tree is \n";

    printPreorder(root);

    cout<<endl;

    cout << "Postorder traversal of binary tree is \n";

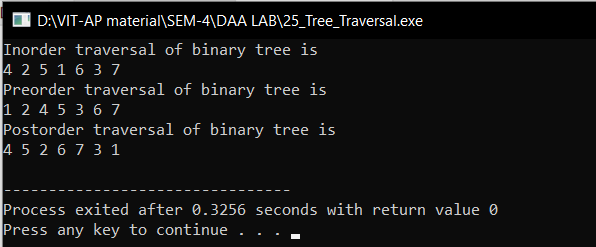
    printPostorder(root);

    cout<<endl;

    return 0;

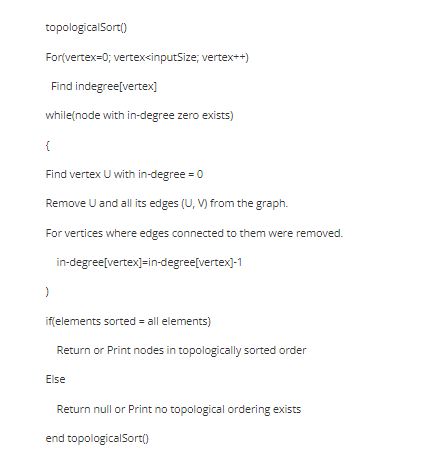
}

Output: -

****

1. Write a C/C++ program to Implement Topological sorting.

Algorithm: -



Time Complexity: -

O(V+E)

Code: -

#include <iostream>

#include <list>

#include <stack>

using namespace std;

class Graph {

    int V;

    list<int>\* adj;

    void topologicalSortUtil(int v, bool visited[], stack<int>& Stack);

public:

    Graph(int V);

    void addEdge(int v, int w);

    void topologicalSort();

};

Graph::Graph(int V)

{

    this->V = V;

    adj = new list<int>[V];

}

void Graph::addEdge(int v, int w)

{

    adj[v].push\_back(w);

}

void Graph::topologicalSortUtil(int v, bool visited[],

                                stack<int>& Stack)

{

    visited[v] = true;

    list<int>::iterator i;

    for (i = adj[v].begin(); i != adj[v].end(); ++i)

        if (!visited[\*i])

            topologicalSortUtil(\*i, visited, Stack);

    Stack.push(v);

}

void Graph::topologicalSort()

{

    stack<int> Stack;

    bool\* visited = new bool[V];

    for (int i = 0; i < V; i++)

        visited[i] = false;

    for (int i = 0; i < V; i++)

        if (visited[i] == false)

            topologicalSortUtil(i, visited, Stack);

    while (Stack.empty() == false) {

        cout << Stack.top() << " ";

        Stack.pop();

    }

}

int main()

{

    Graph g(6);

    g.addEdge(5, 2);

    g.addEdge(5, 0);

    g.addEdge(4, 0);

    g.addEdge(4, 1);

    g.addEdge(2, 3);

    g.addEdge(3, 1);

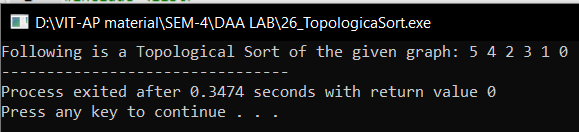
    cout << "Following is a Topological Sort of the given graph: ";

    g.topologicalSort();

    return 0;

}

Output: -



|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Tree Traversal | O(n) | O(h) |
| Topological soting | O(V+E) | O(V) |