

## LAB ASSIGNMENT NO:01

**Name: Kshitija Sakhare**

**Class: BE(B2)**

**Roll No.: B212040**

### DFS PROGRAM

```
#include <iostream>

#include <vector>

#include <stack>

#include <omp.h>

using namespace std;

const int MAX = 100000;

vector<int> graph[MAX];

bool visited[MAX];

void dfs(int node) {

    stack<int> s;

    s.push(node);

    while (!s.empty()) {

        int curr_node = s.top();

        if (!visited[curr_node]) {

            visited[curr_node] = true;

            s.pop();

            cout<<curr_node<<" ";

#pragma omp parallel for

            for (int i = 0; i < graph[curr_node].size(); i++) {

                int adj_node = graph[curr_node][i];

                if (!visited[adj_node]) {

                    s.push(adj_node);

                } } }

    } }
```

```

int main() {

    int n, m, start_node;

    cout<<"Enter no. of Node,no. of Edges and Starting Node of graph:\n";

    cin >> n >> m >> start_node;

    //n: node,m:edges

    cout<<"Enter pair of node and edges:\n";

    for (int i = 0; i < m; i++) {

        int u, v;

        cin >> u >> v;

        //u and v: Pair of edges

        graph[u].push_back(v);

        graph[v].push_back(u);

    }

    #pragma omp parallel for

    for (int i = 0; i < n; i++) {

        visited[i] = false;

    }

    dfs(start_node);

    return 0;

}

```

### OUTPUT:

```

Enter no. of Node,no. of Edges and Starting Node of graph:
4 3 0
Enter pair of node and edges:
0 1
0 2
2 4
0 2 4 1

```

**BFS PROGRAM:**

```
#include<iostream>

#include<stdlib.h>

#include<queue>

using namespace std;

class node

{

    public:

    node *left, *right;

    int data;

};

class Breadthfs

{

    public:

    node *insert(node *, int);

    void bfs(node *);

};

node *insert(node *root, int data)

// inserts a node in tree

{

    if(!root)

    {

        root=new node;

        root->left=NULL;

        root->right=NULL;

        root->data=data;

        return root;

    }

}
```

```

    }
queue<node *> q;
q.push(root);
while(!q.empty())
{
    node *temp=q.front();
q.pop();
if(temp->left==NULL)
    {
        temp->left=new node;
        temp->left->left=NULL;
        temp->left->right=NULL;
        temp->left->data=data;
        return root;
    }
    else
    {
q.push(temp->left);
    }
if(temp->right==NULL)
    {
        temp->right=new node;
        temp->right->left=NULL;
        temp->right->right=NULL;
        temp->right->data=data;
        return root;
    } else

```

```

    {
q.push(temp->right);
    } }

}

void bfs(node *head)
{
queue<node*> q;

    q.push(head);

    int qSize;

    while (!q.empty())
    {
        qSize = q.size();

        #pragma omp parallel for
            //creates parallel threads
        for (int i = 0; i < qSize; i++)
        {
            node* currNode;

            #pragma omp critical
            {
                currNode = q.front();

                q.pop();

                cout<<"\t"<<currNode->data;

                }// prints parent node

            #pragma omp critical
            {

                if(currNode->left)// push parent's left node in queue

                    q.push(currNode->left);
            }
        }
    }
}

```

```

        if(currNode->right)
            q.push(currNode->right);
        } // push parent's right node in queue
    }
}

int main(){
    node *root=NULL;

    int data;

    char ans;

    do
    {
        cout<<"\n enter data=>";

        cin>>data;

        root=insert(root,data);

        cout<<"do you want insert one more node?";

        cin>>ans;

    } while(ans=='y' || ans=='Y');

    bfs(root);

    return 0;
}

```

### **OUTPUT:**

```

enter data=>5

do you want insert one more node?Y

enter data=>89

do you want insert one more node?Y

enter data=>43

do you want insert one more node?Y

```

enter data=>2

do you want insert one more node?N

5      89      43      2

### **BINARY SEARCH TREE:**

```
#include<iostream>
```

```
#include<stdlib.h>
```

```
#include<omp.h>
```

```
using namespace std;
```

```
int binary(int *, int, int, int);
```

```
int binary(int *a, int low, int high, int key)
```

```
{
```

```
    int mid;
```

```
    mid=(low+high)/2;
```

```
    int low1,low2,high1,high2,mid1,mid2,found=0,loc=-1;
```

```
    #pragma omp parallel sections
```

```
    {
```

```
        #pragma omp section
```

```
        {
```

```
            low1=low;
```

```
            high1=mid;
```

```
            while(low1<=high1)
```

```
            {
```

```
if(!(key>=a[low1] && key<=a[high1]))
```

```
        {
```

```
            low1=low1+high1;
```

```
            continue;
```

```

    }
    mid1=(low1+high1)/2;
    if(key==a[mid1])
    {
        found=1;
        loc=mid1;
        low1=high1+1;
    }
    else if(key>a[mid1])
    {
low1=mid1+1;
    }
    else if(key<a[mid1])
    {
        high1=mid1-1;
    }
}

#pragma omp section
{
    low2=mid+1;
    high2=high;
    while(low2<=high2)
    {
if(!(key>=a[low2] && key<=a[high2]))
    {
        low2=low2+high2;
        continue;
        cout<<"here2";
    }
}

```



```

        mid2=(low2+high2)/2;
        if(key==a[mid2])
        {
            found=1;
            loc=mid2;
            low2=high2+1;
        }
        else if(key>a[mid2])
        {
            low2=mid2+1;
        }
        else if(key<a[mid2])
            high2=mid2-1;
    }
    }}
return loc;
}int main()
{
    int *a,i,n,key,loc=-1;
    cout<<"\n enter total no of elements=>";
    cin>>n;
    a=new int[n];
    cout<<"\n enter elements=>";
    for(i=0;i<n;i++)
    {
        cin>>a[i];
    }

```

```

    cout<<"\n enter key to find=>";
    cin>>key;
    loc=binary(a,0,n-1,key);
    if(loc==-1)
        cout<<"\n Key not found.";
    else
        cout<<"\n Key found at position=>"<<loc+1;
    return 0;
}

```

## OUTPUT

```

enter total no of elements=>10

enter elements=>1
2
3
4
5
6
7
8
9
10

enter key to find=>8
here2
Key found at position=>8apr@C04L0801:~$ ./a.out

enter total no of elements=>12

enter elements=>1
2
3
4
5
6
7
8
9

```

```
10
```

```
11
```

```
12
```

```
enter key to find=>15
```

```
Key not found
```