

Experiment Title- 3.1

STUDENT NAME :- Lipakshi

UID :- 20BCS5082

SECTION :-607-B

SEMESTER :- 5TH

SUBJECT:- DESIGN OF ANALYSIS AND ALGORITHM

AIM :- Code and analyze to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, OR (ii) to find a path from source to goal in a maze.

Program Code :-

a) Code and analyze to do a depth-first search (DFS) on an undirected graph

```
#include  
  
<bits/stdc++.h>          using  
  
namespace std;  
  
class Graph {  
  
public:  
  
    map<int, bool> visited;  
  
    map<int, list<int> > adj;  
  
    void addEdge(int v, int  
w);  
  
    void DFS(int v);  
  
};
```

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

```
{  
  
    adj[v].push_back(w);  
  
}  
  
void Graph::DFS(int v)  
  
{  
  
    visited[v] = true;  
    cout << v << " ";  
  
    list<int>::iterator i;  
    for (i = adj[v].begin(); i != adj[v].end();  
         ++i) if (!visited[*i])  
        DFS(*i);  
}  
  
int main()  
  
{  
  
    Graph g;  
  
    g.addEdge(0, 1);  
    g.addEdge(0, 2);  
    g.addEdge(1, 2);  
    g.addEdge(2, 0);  
    g.addEdge(2, 3);  
    g.addEdge(3, 3);  
  
    cout << "Following is Depth First Traversal"  
    " (starting from vertex 2) \n";
```

```
g.DFS(2);

return 0;
}
```

Output :-

The screenshot shows a web browser window with the URL `onlinegdb.com/online_c++_compiler`. The browser has several tabs open, including "Worksheet 3.1" and "Online C++ Com". The OnlineGDB interface includes a sidebar with navigation links like "IDE", "My Projects", "Classroom", "Learn Programming", "Programming Questions", "Sign Up", and "Login". The main editor area displays a C++ program for Depth First Search (DFS) on a graph. The code defines a graph with 4 vertices and 3 edges, and performs a DFS starting from vertex 2. The output window shows the result of the DFS: "Following is Depth First Traversal (starting from vertex 2)" followed by the sequence "2 0 1 3". The program finished with exit code 0.

```
main.cpp
26 visited[v] = true;
27 cout << v << " ";
28
29
30 list<int>::iterator i;
31 for (i = adj[v].begin(); i != adj[v].end(); ++i)
32     if (!visited[*i])
33         DFS(*i);
34 }
35 int main()
36 {
37
38     Graph g;
39     g.addEdge(0, 1);
40     g.addEdge(0, 2);
41     g.addEdge(1, 3);
42
43     DFS(2);
44
45     return 0;
46 }
```

input

```
Following is Depth First Traversal (starting from vertex 2)
2 0 1 3

...Program finished with exit code 0
Press ENTER to exit console.
```

b) to find the topological sort of a directed acyclic graph

Program Code :- #include <bits/stdc++.h>

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 \n";

g.topologicalSort();


return 0;

}
```

Output :-

```
54-   while (Stack.empty() == false) {
55-       cout << Stack.top() << " ";
56-       Stack.pop();
57-   }
58- }
59- int main()
60- {
61-
62-     Graph g(6);
63-     g.addEdge(5, 2);
64-     g.addEdge(5, 0);
65-     g.addEdge(4, 0);
66-     g.addEdge(4, 1);
67-     g.addEdge(2, 3);
```

input

```
Following is a Topological Sort of the given graph
5 4 2 3 1 0
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```


C. to find a path from source to goal in a maze.

Program Code :-

```
#include <iostream>

#include <vector>

#include <climits>

#include <cstring> using

namespace std;

bool isSafe(vector<vector<int>> &mat, vector<vector<bool>> &visited, int x, int y)

{

    return (x >= 0 && x < mat.size() && y >= 0 && y < mat[0].size())

        && mat[x][y] == 1 && !visited[x][y];

}

void findShortestPath(vector<vector<int>> &mat, vector<vector<bool>>

    &visited, int i, int j, int x, int y, int &min_dist, int dist)

{

    if (i == x && j == y)

    {

        min_dist = min(dist,

            min_dist); return;

    }

    visited[i][j] = true;

    if (isSafe(mat, visited, i + 1, j)) {
```

```
        findShortestPath(mat, visited, i + 1, j, x, y, min_dist, dist + 1);
    }

    if (isSafe(mat, visited, i, j + 1)) {
        findShortestPath(mat, visited, i, j + 1, x, y, min_dist, dist + 1);
    }

    if (isSafe(mat, visited, i - 1, j)) {
        findShortestPath(mat, visited, i - 1, j, x, y, min_dist, dist + 1);
    }

    if (isSafe(mat, visited, i, j - 1)) {
        findShortestPath(mat, visited, i, j - 1, x, y, min_dist, dist + 1);
    }

    visited[i][j] = false;
}

int findShortestPathLength(vector<vector<int>> &mat, pair<int, int> &src,
                           pair<int, int> &dest)
{
    if (mat.size() == 0 || mat[src.first][src.second] == 0 ||
        mat[dest.first][dest.second] == 0) {
        return -1;
    }
}
```

```
int M = mat.size();

int N =

mat[0].size();

vector<vector<bool>> visited;

visited.resize(M, vector<bool>(N));

int min_dist = INT_MAX;

findShortestPath(mat, visited, src.first, src.second, dest.first, dest.second,

    min_dist, 0);


if (min_dist != INT_MAX) {

    return min_dist;

}


return -1;

}


int main()

{

    vector<vector<int>> mat =

    {

        { 1, 1, 1, 1, 1, 0, 0, 1, 1, 1 },

        { 0, 1, 1, 1, 1, 1, 0, 1, 0, 1 },

        { 0, 0, 1, 0, 1, 1, 1, 0, 0, 1 },

        { 1, 0, 1, 1, 1, 0, 1, 1, 0, 1 },

        { 0, 0, 0, 1, 0, 0, 0, 1, 0, 1 },

        { 1, 0, 1, 1, 1, 0, 0, 1, 1, 0 },

        { 0, 0, 0, 0, 1, 0, 0, 1, 0, 1 },

        { 0, 1, 1, 1, 1, 1, 1, 1, 0, 0 },
```

```
{ 1, 1, 1, 1, 1, 0, 0, 1, 1, 1 },
```

```
{ 0, 0, 1, 0, 0, 1, 1, 0, 0, 1 },
```

```
};
```

```
pair<int, int> src = make_pair(0, 0);
```

```
pair<int, int> dest = make_pair(7, 5);
```

```
int min_dist = findShortestPathLength(mat, src, dest);
```

```
if (min_dist != -1)
```

```
{
```

```
    cout << "The shortest path from source to destination "
```

```
        "has length " << min_dist;
```

```
}
```

```
else {
```

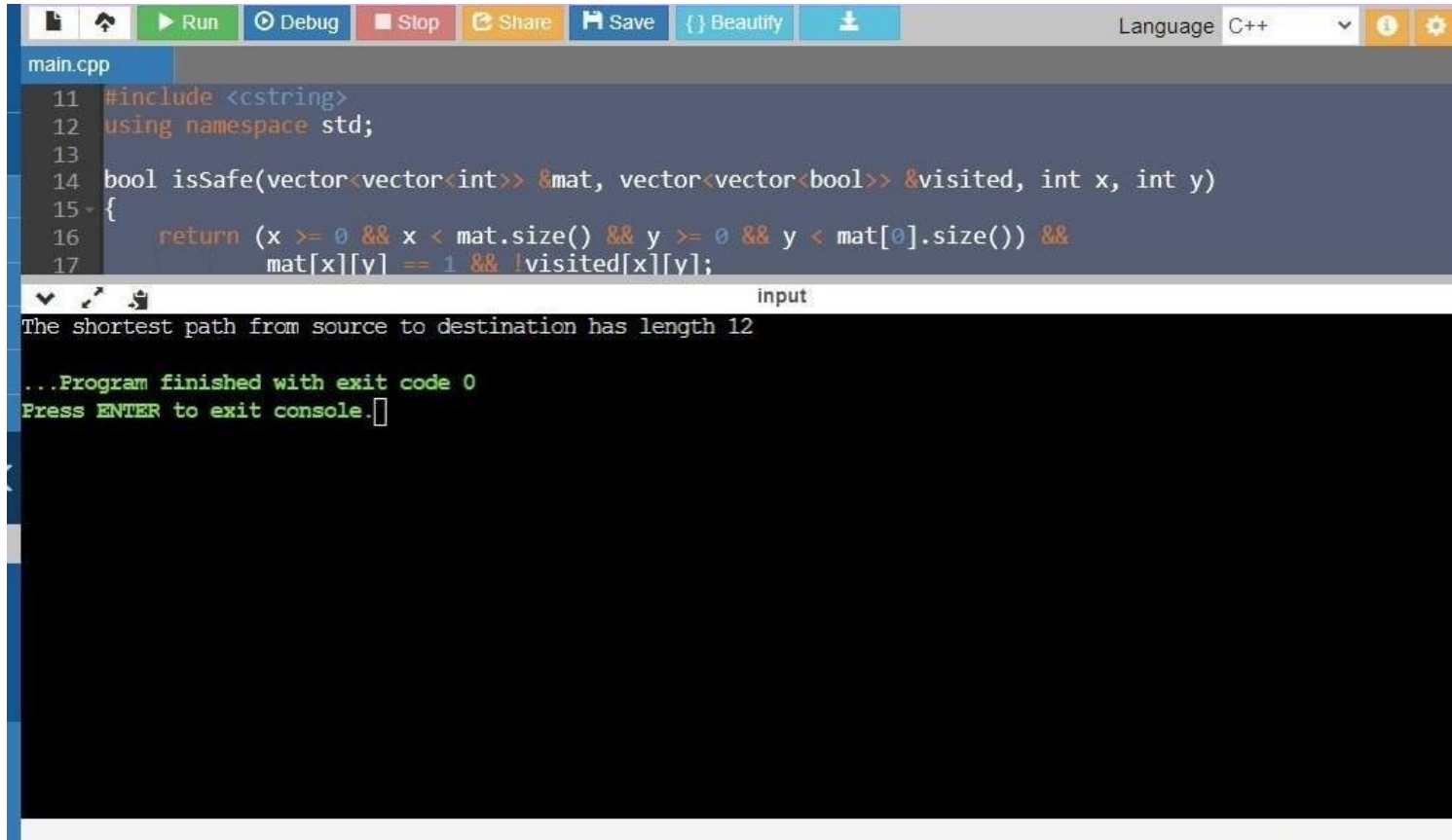
```
    cout << "Destination cannot be reached from a given source";
```

```
}
```

```
return 0;
```

```
}
```

OUTPUT:



The screenshot shows a C++ IDE with a toolbar at the top containing icons for file operations, running, debugging, stopping, sharing, saving, and beautifying code. The language is set to C++. The code in the editor is as follows:

```
main.cpp
11 #include <cstring>
12 using namespace std;
13
14 bool isSafe(vector<vector<int>> &mat, vector<vector<bool>> &visited, int x, int y)
15 {
16     return (x >= 0 && x < mat.size() && y >= 0 && y < mat[0].size()) &&
17         mat[x][y] == 1 && !visited[x][y];
```

The output window shows the result of the program execution:

```
input
The shortest path from source to destination has length 12
...Program finished with exit code 0
Press ENTER to exit console.
```

