AIM:-

To Store a graph using adjacency matrix or adjacency list representation and perform

Depth First Traversal.

OBJECTIVE:-

To learn about depth first traversal of a graph.

Theory:-

Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.  
This means that we traverse from the first node to the deepest possible node from the same and bounce back to the next deepest node repeatedly, with the help of a stack. We push all of the nodes that we visit into the stack in the order of visiting, and when we face a dead end in traversal we just refer the previous nodes by popping the stack.



Code :-

#include<iostream>

#include<list>

using namespace std;

// Graph class represents a directed graph

// using adjacency list representation

class Graph

{

    int V;    // No. of vertices

    // Pointer to an array containing

    // adjacency lists

    list<int> \*adj;

    // A recursive function used by DFS

    void DFSUtil(int v, bool visited[]);

public:

    Graph(int V);   // Constructor

    // function to add an edge to graph

    void addEdge(int v, int w);

    // DFS traversal of the vertices

    // reachable from v

    void DFS(int v);

};

Graph::Graph(int V)

{

    this->V = V;

    adj = new list<int>[V];

}

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

{

    adj[v].push\_back(w); // Add w to v’s list.

}

void Graph::DFSUtil(int v, bool visited[])

{

    // Mark the current node as visited and

    // print it

    visited[v] = true;

    cout << v << " ";

    // Recur for all the vertices adjacent

    // to this vertex

    list<int>::iterator i;

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

        if (!visited[\*i])

            DFSUtil(\*i, visited);

}

// DFS traversal of the vertices reachable from v.

// It uses recursive DFSUtil()

void Graph::DFS(int v)

{

    // Mark all the vertices as not visited

    bool \*visited = new bool[V];

    for (int i = 0; i < V; i++)

        visited[i] = false;

    // Call the recursive helper function

    // to print DFS traversal

    DFSUtil(v, visited);

}

// Driver code

int main()

{

    // Create a graph given in the above diagram

    Graph g(4);

    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;

}

Conclusion:-

Hence we conclude that the program to store graph using adjacency matrix and performing

Depth First Traverse was completed successfully.

