**DSA Assignment No.13**

Title :-

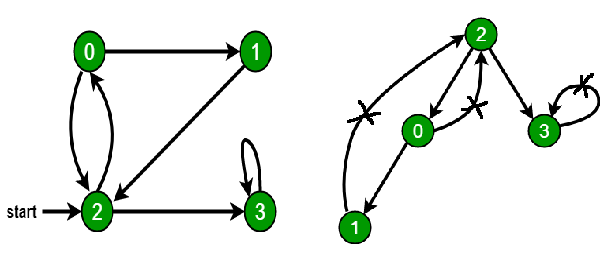
Store a graph using adjacency matrix or adjacency list representation and perform Depth First Traversal.(Recursive/Non Recursive)

Description :-

Depth First Search or DFS for a Graph

Unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array.

For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don’t mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Depth First Traversal of the following graph is 2, 0, 1, 3.

[](https://media.geeksforgeeks.org/wp-content/uploads/cycle.png)

SOURCE CODE :-

|  |
| --- |
| #include<iostream>  #include<list>  using namespace std;    class Graph  {      int V;        list<int> \*adj;        void DFSUtil(int v, bool visited[]);  public:      Graph(int V);          void addEdge(int v, int w);        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);  }    void Graph::DFSUtil(int v, bool visited[])  {        visited[v] = true;      cout << v << " ";          list<int>::iterator i;      for (i = adj[v].begin(); i != adj[v].end(); ++i)          if (!visited[\*i])              DFSUtil(\*i, visited);  }      void Graph::DFS(int v)  {        bool \*visited = new bool[V];      for (int i = 0; i < V; i++)          visited[i] = false;   DFSUtil(v, visited);  }    int main()  {        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;  } |
|  |

**Output:**

Following is Depth First Traversal (starting from vertex 2)

2 0 1 3

Conclusion :-

1. To know about the DFS (Depth First Search ) for A Graph how to taverse it using DFS.
2. To know about the representation of the of a graph using Adjecency list .