Experiment Number 6

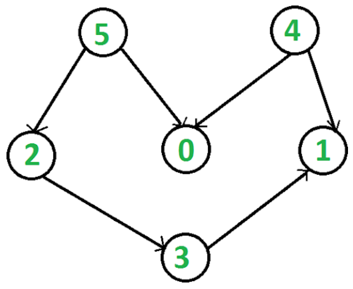
Name ::	Rishabh Anand	UID ::	19BCS4525
Branch ::	CSE - IoT	Sec/Grp ::	1/A
Semester ::	5th	Date ::	14th Nov, 2021
Subject ::	Adv Programming Lab	CODE ::	CSP-347

1. Aim :

Obtain the Topological ordering of vertices in a given digraph.

2. Task :

1. Obtain the Topological ordering of vertices in a given digraph.



3. Algorithm :

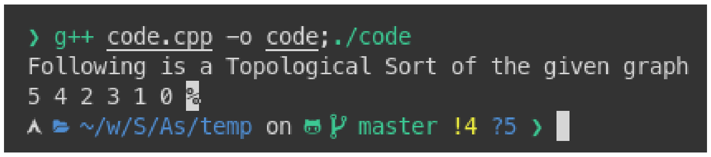
- Calculate the indegrees of all the vertices of the given graph.
- Find the vertex with indegree 0 and print it, if there is no such vertex in the given graph than it means that there is a cycle and the vertices cannot be ordered. Stop at this point if this case encountered.
- Remove that vertex with indegree 0 and all its associated edges as well.
- Update the indegrees of all the vertex after step 3.
- Repeat steps 2 to 4 till all the vertex goes through the same process.

4. Source 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.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;
}
```

5. Observations :



```
❯ g++ code.cpp -o code;./code
Following is a Topological Sort of the given graph
5 4 2 3 1 0 %
~/w/S/As/temp on master !4 ?5 ❯
```

Learning Outcomes :

- Come to know about graph and its concepts of indegree outdegree etc.
- Come to know about graphs and its different types and their classification on the basis of different parameters.
- Topological sorting, its advantages and disadvantages and its application in the real world.

S. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			