Experiment: 6

Write a program to implement topological ordering for a directed acyclic graph.

Student Name: Nitish Rai

UID: 23MCA20326

Branch: Computers Application

Section/Group: 4(A)

Semester: 2nd

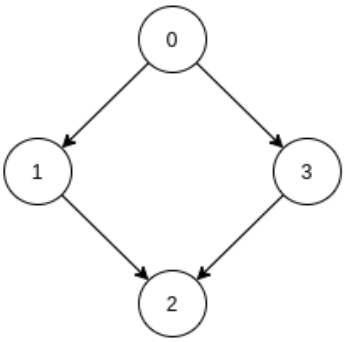
Date of Performance: 11/03/2024

Subject Name: Design and Analysis of Algorithms

Subject Code: 23CAH-511

A. The task is to:

Implement topological sort and find out any topological sort for the given directed acyclic graph.



B. Steps of Experiment:

- Understand and divide the problem into parts.
- Install JDK: If Java isn't installed on your system, download and install it.
- Open IDE(Integrated development environment) like VS code.
- Start creating a new Java file with a .java extension (e.g., program.java).
- Write your code in a structural manner taking care of indentation to maintain readability.
- Execute your code

C. Algorithm:

1. Create a Stack (say st) which will be used to store the topological ordering.
2. Create a boolean array (say visited), it will be used to mark the vertices that have been visited.
3. For each unvisited vertex (say node) from 0 to V-1 call a recursive helper function which will do the following:
    - Mark node as visited.
    - For each adjacent vertex of node call the recursive function.
    - Push node in St.

D. Pseudocode:

```
topologicalSort()

For(vertex=0; vertex<inputSize; vertex++)

  Find indegree[vertex]

while(node with in-degree zero exists)

{

Find vertex U with in-degree = 0

Remove U and all its edges (U, V) from the graph.

For vertices where edges connected to them were removed.

    in-degree[vertex]=in-degree[vertex]-1

)

if(elements sorted = all elements)

    Return or Print nodes in topologically sorted order

Else

    Return null or Print no topological ordering exists

end topologicalSort()
```

E. Code:

```
#include <bits/stdc++.h>
using namespace std;

void solveDFS(unordered_map<int, list<int> > &adjList, stack<int> &st, vector<bool> &visited, int node) {
   visited[node] = true;

   for(auto x : adjList[node]) {
      if(!visited[x]) {
         solveDFS(adjList, st, visited, x);
      }
   }

   st.push(node);
}

vector<int> topologicalSort(vector< vector<int> > &edges, int v, int e) {
   unordered_map<int, list<int> > adjList;
   for(int i=0; i<e; i++) {
      int u = edges[i][0];
      int v = edges[i][1];
      adjList[u].push_back(v);
   }

   vector<int> solution;
   stack<int> st;
   vector<bool> visited(v,0);

   for(int i=0; i<v; i++) {
      if(!visited[i]) {
         solveDFS(adjList, st, visited, i);
      }
   }

   while(!st.empty()) {
      solution.push_back(st.top());
      st.pop();
   }

   return solution;
}

int main() {
   vector< vector<int> > edges;
   int n, m;

   cout << "Enter number of nodes : ";
   cin >> n;

   cout << "Enter number of edges : ";
   cin >> m;

   cout << "Enter edges : " << endl;
   for(int i=0; i<m; i++) {
      int u, v;
      cin >> u >> v;
      edges.push_back({u,v});
   }
```

```
    vector<int> topSort = topologicalSort(edges, n, m);

    cout << "Topological Sort : ";
    for(int x : topSort) {
        cout << x << " ";
    }

    return 0;
}
```

F. Output:

```
PS C:\Users\Nitish\Desktop\JAVA\javaComeBack> cd "c:\Users\Nitish\Desktop\JAVA\javaComeBack\" ; if ($?) { javac TopologicalSort.java } ; if ($?) { java TopologicalSort}
Enter number of nodes: 4
Enter number of edges : 4
Enter edges :
0 1
1 2
0 3
3 2
Topological Sort : 0 3 1 2
PS C:\Users\Nitish\Desktop\JAVA\javaComeBack>
```

G. Time Complexity:

O(V+ E)

Learning outcomes:

- Understand how to use sets.
- Understand the concept of the creation of a graph.
- Understanding how to deal with errors..
- Understand what topological sort is and how to implement it.
- Understand the adjacency list.