Name: Aditi Helekar

UNC ID: 800966667

Project Report

Project Question:

Write a program to process a directed graph as follows:

1. Read in the number of vertices V and the number of edges E of the graph followed by its E edges, each in the form u, v, w where 1 <= u, v <= V & w > 0 representing a direct edge <u, v> with weight w.
2. Set up and print the adjacency matrix representation of the Graph.
3. Determine whether the graph is a dag. If it is a dag, print a topological order. If it is not a dag, print a cycle.
4. Find a shortest path tree from vertex 1. Print the adjacency matrix representation of the tree.

The program consists of two classes and a main file.

1. main.cpp
2. DirectedGraph.h & DirectedGraph.cpp
3. DFS.h & DFS.cpp

Algorithms used are:

1. Depth First Search using adjacency list technique
2. Topological Sort
3. All Pairs Shortest Path algorithm (Floyd Warshall Algorithm)

Steps followed in program as per files:

1. Main.cpp
    a. Creates object of DirectedGraph.
    b. Upon completion of processing deletes the above object.

2. DirectedGraph.h & DirectedGraph.cpp
    a. Takes the input for graph from user such as number of vertices, number of edges and the edges.
    b. Creates Adjacency matrix from the data and assigns 0 for no edge and 1 for edge existing.
    c. Creates object of DFS which performs Depth first search on graph to detect the cycle.
    d. Constructs graph for DFS
    e. Performs shortest path Dijkstra algorithm for Vertex 1 and print the shortest path tree.

3. DFS.h & DFS.cpp
    a. This class implements Depth first search and Topological sort technique.
    b. Creates graph by adding edges from adjacency matrix.
    c. Checks if the given graph is Cyclic if true then does not perform topological sort.
    d. If graph is not cyclic it applies Topological Sort to find the Topological order of the vertices visited.

Time complexity of Algorithms:

1. Depth First Search with adjacency list
   The time complexity of Depth first search is O(V + E) where
   V => number of vertices of graph
   E => number of edges of graph
   As the edges and vertices increase in number the complexity increases.

2. Topological Sort
   The number of operations is O(|E| + |V|) where
   V => number of vertices of graph
   E => number of edges of graph
   For Adjacency list representation: O(|E|)
   For Matrix representation: O(|V|^2)

3. Dijkstra Algorithm
   The time complexity is O(|V|^2)
   V => number of vertices of graph

Outputs:

Graph 1.

```
5 8

1 2 8

1 4 5

4 2 4

4 3 2

3 5 1

3 2 1

5 1 5

5 4 2
```

```
Enter the total number of vertices:
5
Enter the total number of edges:
8
Enter the directed edges in the form of <u v w>:
1,2,8
1,4,5
4,2,4
4,3,2
3,5,1
3,2,1
5,1,5
5,4,2
The adjacency matrix for the given graph is:
08050
00000
01001
04200
50020
Graph is cyclic.
```

```
Vertex  Distance from Source Vertex 1

 1          0

 2          8

 3          7

 4          5

 5          8


The adjacency matrix for the Shortest Path Tree is:

08050

00000

00001

00200

00000
```

Graph 2.

```
7 9

1 2 2

2 3 3

3 4 5

4 5 3

3 5 1

3 7 8

3 6 6

5 6 4

6 7 2

Enter the total number of vertices:

7
```

```
Enter the total number of edges:
9
Enter the directed edges in the form of <u v w>:
1,2,2
2,3,3
3,4,5
4,5,3
3,5,1
3,7,8
3,6,6
5,6,4
6,7,2
The adjacency matrix for the given graph is:
0200000
0030000
0005168
0000300
0000040
0000002
0000000
Topological order is:
3       7       6       5       4       2       1
Graph is not cyclic.
```

Vertex	Distance from Source Vertex 1
1	0
2	2
3	5
4	10
5	6
6	10
7	12

The adjacency matrix for the Shortest Path Tree is:

0200000

0030000

0005100

0000000

0000040

0000002

0000000