ANALYSIS REPORT FOR ASSIGNMENT-4

NAME- MAREESH KUMAR ISSAR
NET ID- mi251
COURSE- 16:332:573 DATA STRUCTURE AND ALGORITHMS
DATE OF SUBMISSION- 30TH APRIL,2019

ANSWER 1

GRAPH CYCLE DETECTION

The program Cycle_undirected.cpp checks whether the graph is acyclic or not (for an undirected graph). DFS is used for detecting cycles. The graph in file mediumEWG.txt is cyclic i.e. it has cycle 211,222,225,211.

ANSWER 2

KRUSKAL AND PRIM ALGORITHM

Weight of the MST is 10.4635 in all the three implementations- Kruskal, Prim(Lazy implementation), and Prim (Eager implementation) for the file mediumEWG.txt. The running time of these algorithms is shown in Table 1. Theoretically, prim algorithm (eager approach) should be the fastest as it maintains a priority queue of vertices that are connected with an edge to the tree but in my implementation it takes the longest running time due to the use of complex underlying data structures like map (for priority queue implementation).

TIME IN MICROSECONDS	ALGORITHM	THEORITICAL WORST CASE RUNNING TIME
105365	Kruskal	E Log E
689217	Prim(Lazy approach)	E Log E
821153	Prim(Eager approach)	E Log V

Table:1

ANSWER 3

LONGEST AND SHORTEST PATH IN EDGE WEIGHTED DAG

The topological order of the given graph is 5 1 3 6 4 7 0 2

The computation of the shortest path is as follows:-

V	edgeTo[]	distanceTo[]
0	4-0	0.73
1	5-1	0.32

2	7-2	0.62
3	1-3	0.61
4	5-4	0.35
5	-	0.0
6	3-6	1.13
7	5-7	0.28

Table:2

1. Add 5 to tree and relax all the edges pointing from 5 i.e. 5-1,5-7,5-4.
2. Add 1 to tree and relax all the edges pointing from 1 i.e. 1-3.
3. Add 3 to tree and relax all the edges pointing from 3 i.e. 3-7,3-6.
4. Add 6 to tree and relax all the edges pointing from 6 i.e. 6-2,6-0,6-4.
5. Add 4 to tree and relax all the edges pointing from 4 i.e. 4-7,4-0.
6. Add 7 to tree and relax all the edges pointing from 7 i.e. 7-2.
7. Add 0 to tree and relax all the edges pointing from 0 i.e. 0-2.
8. Add 2 to tree.

The computation of the longest path is as follows:-

V	edgeTo[]	distanceTo[]
0	4-0	2.44
1	5-1	0.32
2	7-2	2.77
3	1-3	0.61
4	6-4	2.06
5	-	0.0
6	3-6	1.13
7	4-7	2.43

Table:3

Here, we first negate all the weights present in the graph, compute the shortest path and then negate all the weights in the final result.

1. Add 5 to tree and relax all the edges pointing from 5 i.e. 5-1,5-7,5-4.
2. Add 1 to tree and relax all the edges pointing from 1 i.e. 1-3.
3. Add 3 to tree and relax all the edges pointing from 3 i.e. 3-7,3-6.
4. Add 6 to tree and relax all the edges pointing from 6 i.e. 6-2,6-0,6-4.
5. Add 4 to tree and relax all the edges pointing from 4 i.e. 4-7,4-0.
6. Add 7 to tree and relax all the edges pointing from 7 i.e. 7-2.
7. Add 0 to tree and relax all the edges pointing from 0 i.e. 0-2.

8. Add 2 to tree.

ANSWER 4

BELLMAN FORD ALGORITHM

4(a)

Pass 1

Relax all edges in the following order for finding the SPT for source vertex 0:-

0-2,0-4,2-7,4-5,4-7,7-3,7-5,5-1,5-4,5-7,3-6,1-3,6-0,6-2,6-4

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	1.05
2	0-2	0.26
3	7-3	0.99
4	6-4	0.26
5	4-5	0.73
6	3-6	1.51
7	2-7	0.6

Table:4

Pass 2

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	0.93
2	0-2	0.26
3	7-3	0.99
4	6-4	0.26
5	4-5	0.61
6	3-6	1.51
7	2-7	0.6

Table:5

Pass 3

V	edgeTo[]	distanceTo[]
0	-	0.0

1	5-1	0.93
2	0-2	0.26
3	7-3	0.99
4	6-4	0.26
5	4-5	0.61
6	3-6	1.51
7	2-7	0.6

Table:6

No change in the table occurs for the rest of the passes (till pass 8). Hence we have obtained the shortest path tree for source vertex 0.

4(b) Pass 1

Relax all edges in the following order for finding the SPT for source vertex 0:-

0-2,0-4,2-7,4-5,4-7,7-3,7-5,5-1,5-4,5-7,3-6,1-3,6-0,6-2,6-4

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	1.05
2	0-2	0.26
3	7-3	0.99
4	5-4	0.07
5	4-5	0.73
6	3-6	1.51
7	2-7	0.6

Table:7

Pass 2

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	0.74
2	0-2	0.26
3	7-3	0.83
4	5-4	-0.24
5	4-5	0.42
6	3-6	1.35
7	4-7	0.44

Table:8

Pass 3

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	0.32
2	0-2	0.26
3	7-3	0.52
4	5-4	-0.55
5	4-5	0.11
6	3-6	1.04
7	4-7	0.13

Table:9

Pass 4

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	0.12
2	0-2	0.26
3	7-3	0.21
4	5-4	-0.86
5	4-5	-0.2
6	3-6	0.73
7	4-7	-0.18

Table:10

Pass 5

V	edgeTo[]	distanceTo[]
0	-	0.0
1	5-1	-0.19
2	0-2	0.26
3	7-3	-0.1
4	5-4	-1.17
5	4-5	-0.51
6	3-6	0.42
7	4-7	-0.49

Table:11

We can clearly see that with each pass the value of distanceTo[] for vertex 4 becomes smaller and smaller due to the presence of the negative edge from 5->4.

This change in value in turn triggers the updates in the values of all the other connected vertices and the change propagates. The value of distanceTo[] for vertex 4 continues decreasing for 3 more passes and in pass 8 we will find there is change in the distanceTo[] values for several vertices including vertex 4, which signifies the presence of a negative cycle in the graph.

ANSWER 5

BFS AND DFS TRAVERSAL

Output on running BFS

Source vertex-23

Destination vertex-100

Shortest hop count distance from 23 to 100 is 42

The program also displays the complete path from 23 to 100

Output on running DFS

After processing the edges from the file and making the graph when I enter the source vertex (to run DFS algorithm), the program gives a runtime error. A major reason for this error can be the inadequate size of the function call stack as the DFS implemented in the program is of recursive nature.

ANSWER 6

DIJKSTRA ALGORITHM

For source vertex 0, the output for file 4a.txt is:-

no path exists

0->2 w 0.26 2->7 w 0.34 7->3 w 0.39 3->6 w 0.52 6->4 w -1.25 4->5 w 0.35 5->1 w 0.32

0->2 w 0.26

0->2 w 0.26 2->7 w 0.34 7->3 w 0.39

0->2 w 0.26 2->7 w 0.34 7->3 w 0.39 3->6 w 0.52 6->4 w -1.25

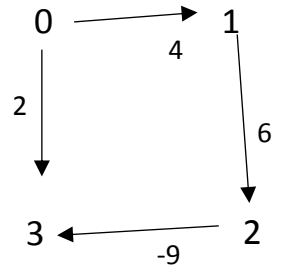
0->2 w 0.26 2->7 w 0.34 7->3 w 0.39 3->6 w 0.52 6->4 w -1.25 4->5 w 0.35

0->2 w 0.26 2->7 w 0.34 7->3 w 0.39 3->6 w 0.52

0->2 w 0.26 2->7 w 0.34

For the above case when we run Dijkstra algorithm on Q4(a) (the graph has negative weights but no negative cycle)

We cannot guarantee that the algorithm will always give us the right result in the presence of negative weights because the algorithm relaxes each edge exactly once. After an edge has been relaxed it does not revisit to check whether another path (possibly longer but with negative weights) exists. For, example



In the graph shown above Dijkstra's algorithm will select the 0->3 edge as it is the smallest outgoing edge from vertex 0 but it is not the shortest path from 0->3. The shortest path from 0->3 is 0->1->2->3.

For source vertex 0, the output for file 4b.txt is stuck in an endless loop due to the presence of a negative cycle (for which the Dijkstra algorithms keeps relaxing the edges in the cycle).