Report

Objective:

To obtain the shortest path from the source node to each of the other nodes in the network.

Algorithm:

Step 1: Obtain the date from the input file line by line in the string format.

Step 2: Arrange the input data in a structure graph that contains number of nodes, number of edges and structure edge as the attributes. Edge structure contains source, destination and edge cost as the attributes.

```
typedef struct edges {
int src;
int dest;
int weight;
} Edge;

typedef struct graphs {
int num_vert;
int num_edge;
Edge *edge;
} Graph;
```

Step 3: Using the structure values, apply Bellman Ford algorithm to the graph.

Step 4: In Bellman Ford function, the shortest distance to each of the node is updated. Distance update takes place inside two loops. Outer loop runs V – 1 times where V is the number of edges. Inner loop runs E times where E represents total number of edges.

Step 5: Also, maintain a preceding array that contains the previous node for each of the node. It represents the shortest path to that node.

```
if (distance[graph->edge[j].src] != INT_MAX && distance[graph->edge[j].src] +
graph->edge[j].weight < distance[graph->edge[j].dest])
{
distance[graph->edge[j].dest] = distance[graph->edge[j].src] + graph-
>edge[j].weight;
preceeding[graph->edge[j].dest] = graph->edge[j].src;
stop_iter = 0;
}
```

Step 6: Terminate early if there is no change in distance array. This is done using the stop_iter flag.

Step 7: Print the distance array into the output file.

Step 8: Using the preceding array, print the shortest path to each of the node in the graph starting from the source node.

Input and Output:

N7:

7

0,*,100,10,*,32,*

4,0,*,*,17,*,5

5,*,0,30,*,42,*

,23,3,0,14,,*

,10,,26,0,2,*

,,9,13,3,0,*

,6,,*,12,12,0

output-N7 file:

0,33,13,10,24,26,38
0
0->3->1
0->3->2
0->3
0->3->4
0->3->4->5
0->3->1->6

N10:

10
0,*,3,*,2,*,*,1,*,*
4,0,3,*,*,3,8,*,2,*
2,*,0,*,5,*,4,8,*,*

5,*,*,0,*,4,*,*,7,4
,3,8,,0,*,*,3,*,*
,1,,*,4,0,*,*,*,*
,,5,3,*,*,0,*,*,1
,,7,*,2,*,*,0,*,*
,2,,*,*,6,7,1,0,*
,,4,*,*,3,1,2,*,0

output-N10:

0,5,3,10,2,8,7,1,7,8
0
0->4->1
0->2
0->2->6->3
0->4
0->4->1->5
0->2->6
0->7
0->4->1->8
0->2->6->9

N20:

20

0,6,9,*,6,4,*,*,9,7,*,*,*,*,*,3,*,*,4,*

,0,2,9,5,,*,*,9,3,9,*,*,*,*,*,2,*,8,2

,,0,2,*,*,*,*,8,*,*,*,*,5,1,*,*,7,2,5

,6,,0,*,4,*,6,*,6,8,4,*,*,*,*,*,9,*,2

4,*,*,4,0,7,5,7,*,1,*,4,*,2,8,6,*,2,7,9

8,*,*,6,*,0,*,3,*,*,8,9,*,*,1,5,*,4,4,1

,,*,9,*,4,0,*,*,5,*,*,*,*,*,*,*,*,5,*

,7,1,,1,*,3,0,*,9,*,*,*,*,4,*,*,7,*,4

4,*,2,*,*,5,*,*,0,*,8,*,1,*,3,9,4,*,*,2

,3,7,,8,*,*,*,3,0,5,7,*,*,5,5,8,3,*,7

,,*,*,*,6,*,*,*,*,0,*,*,*,3,*,3,*,*,*

8,*,*,*,1,*,*,*,2,8,*,0,*,*,4,2,1,*,3,3

4,*,9,*,*,4,*,*,*,*,*,7,3,0,*,1,3,*,*,*,*

9,6,*,*,9,6,8,*,*,*,*,3,*,0,*,3,7,*,*,7

,,*,5,*,2,*,*,*,9,9,*,*,1,0,4,6,6,*,6

3,1,*,*,*,*,9,*,*,1,*,*,5,3,*,0,7,9,4,2

,9,,6,7,*,*,5,8,7,*,4,*,*,*,*,0,*,*,6

5,7,7,*,*,*,*,3,*,9,4,*,*,*,*,2,*,0,6,1

8,4,3,*,*,9,6,3,*,6,*,3,*,*,*,*,1,*,0,*

,4,,2,1,*,*,2,*,4,*,2,*,5,9,*,*,3,5,0

output-N20:

0,4,6,7,6,4,10,7,7,4,9,7,8,6,5,3,5,7,4,5
0
0->15->1
0->15->1->2
0->5->19->3
0->4
0->5
0->5->7->6
0->5->7
0->15->9->8
0->15->9
0->15->9->10
0->18->11
0->15->12
0->5->14->13
0->5->14
0->15
0->18->16
0->15->9->17
0->18
0->5->19

Output file format:

Line 1 contains the shortest path distance from source node to each node in the graph.
Line 2 to end of the file describes the shortest path from source node.