G. H. RAISONI COLLEGE OF ENGG., NAGPUR
(An Autonomous Institute under UGC Act 1956)
Department of Computer Science & Engg.

Date: 20/08/2020

Practical Subject: Design and Analysis of Algorithms
Session: 2020-21

Student Details:

Roll Number	58
Name	Shivam Tawari
Semester	3
Section	A
Branch	Artificial Intelligence

Practical Details: Practical Number- 10

Practical Aim	To implement and analyze time complexity of Single Source shortest path Algorithm: Bellman-ford.
Theory & Algorithm	**Theory:** ***Bellman-Ford Algorithm:*** The Bellman-Ford algorithm is a graph search algorithm that finds the shortest path between a given source vertex and all other vertices in the graph. This algorithm can be used on both weighted and unweighted graphs. It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers.

	Algorithm: Step 1: START Step 2: Initialize distances from the source to all vertices as infinite and distance to the source itself as 0. Step 3: For each edge a->b of the graph If dis[b] > dis[a] + weight of edge (a->b) then dis[b] = dis[a] + weight of edge (a->b) Step 4: Repeat step 2 for nv-1 times (nv is no. of vertices). Step 5: If dis[b] > dis[a] + weight of edge (a->b) then report a negative cycle in the graph. Step 6: STOP
Complexity	Time Complexity of Bellman-Ford Algorithm is **O(VE)**.

Program	main.cpp
	1 #include "iostream"
	2 #include "vector"
	3
	4 using namespace std;
	5
	6 struct edge
	7 {
	8 int a, b, cost;
	9 };
	10
	11 int n, m, v;
	12 vector<edge> e;
	13 const int INF = 1000000000;
	14
	15 void solve()
	16 {
	17 vector<int> d (n, INF);
	18 d[v] = 0;
	19 while (true) {
	20 bool any = false;
	21
	22 for (int j=0; j<m; ++j)
	23 if (d[e[j].a] < INF)
	24 if (d[e[j].b] > d[e[j].a] + e[j].cost)
	25 {
	26 d[e[j].b] = d[e[j].a] + e[j].cost;
	27 any = true;
	28 }
	29

```
        if (!any) break;
    }
    cout << "\n Final Cost of Vertices from Starting Vertice " << v << ": ";
    for (auto x: d) {
        cout << x << " ";
    }
}

int main(int argc, char const *argv[]) {
    int a, b, cost;

    cout << "\n Name: Shivam Tawari";
    cout << "\n Roll no: A-58 Section: A";

    cout << "\n Enter Total Vertices: ";
    cin >> n;
    cout << " Enter Total Edges: ";
    cin >> m;

    for (int i=0; i<m; i++) {
        cout << " Enter start, end and cost of edge " << i+1 << ": ";
        cin >> a >> b >> cost;
        e.push_back({a, b, cost});
    }

    cout << " Enter starting vertex: ";
    cin >> v;

    solve();

    return 0;
}
```

Output

```
Name: Shivam Tawari
Roll no: A-58 Section: A
Enter Total Vertices: 2
Enter Total Edges: 4
Enter start, end and cost of edge 1: 2
3
6
Enter start, end and cost of edge 2: 4
9
2
Enter start, end and cost of edge 3: 6
2
6
Enter start, end and cost of edge 4: 3
7
8
Enter starting vertex: 2

Final Cost of Vertices from Starting Vertice 2: 1000000000 1000000000
```