Name: Sapate Vaibhav Ramdas

Roll No: 307B055

Division: 2

Batch: C

Assignment No: 2

Problem Statement:

Write a program to implement Bellman-Ford Algorithm using Dynamic Programming and verify the time complexity

Program: (With proper comments):

```
#include <iostream>
#include <climits>
using namespace std;
// Structure for directed edge
struct DirectedEdge
{
   int source, destination, weight;
};
// Bellman-Ford function for directed graph
void BellmanFord(DirectedEdge edges[], int V, int E, int source)
{
   int dist[V];
   // Initialize distances to maximum value for all vertices except the source
   for (int i = 0; i < V; i++)
   {
      dist[i] = INT_MAX;
```

```
    }
    dist[source] = 0; // Distance from the source to itself is 0
    // Relax edges repeatedly (V - 1 times)
    for (int i = 1; i <= V - 1; i++)
    {
        for (int j = 0; j < E; j++)
        {
            int u = edges[j].source;
            int v = edges[j].destination;
            int w = edges[j].weight;
            // If relaxation condition is met, update the distance
            if (dist[u] != INT_MAX && dist[u] + w < dist[v])
            {
                dist[v] = dist[u] + w;
            }
        }
    }
    // Check for negative weight cycles
    for (int i = 0; i < E; i++)
    {
        int u = edges[i].source;
        int v = edges[i].destination;
        int w = edges[i].weight;
        // If relaxation condition is met after V - 1 iterations, a negative cycle
exists
        if (dist[u] != INT_MAX && dist[u] + w < dist[v])
        {
            cout << "Graph contains a negative weight cycle." << endl;
```

```
            return;
        }
    }
    // Print the result
    cout << "Vertex:\t\tDistance from Source:" << endl;
    for (int i = 0; i < V; i++)
    {
        cout << i << "\t\t" << (dist[i] == INT_MAX ? "INF" : to_string(dist[i])) <<
endl;
    }
}
int main()
{
    int V = 7;
    int E = 10;
    DirectedEdge edges[] = {
        // Start, Destination, Distance
        {0, 1, 6},
        {0, 2, 5},
        {0, 3, 5},
        {2, 1, -2},
        {3, 2, -2},
        {3, 4, -1},
        {1, 5, -1},
        {2, 5, -1},
        {4, 6, 3},
        {5, 6, 3}};
    int start;
```

```
    cout << "Enter the starting source: ";
    cin >> start;
    BellmanFord(edges, V, E, start);
    return 0;
}
```

Output:

Enter the starting source: 0

Vertex:	Distance from Source:
0	0
1	1
2	3
3	5
4	4
5	0
6	3

Enter the starting source: 3

Vertex:	Distance from Source:
0	INF
1	-4
2	-2
3	0
4	-1
5	-5
6	-2