**Exercise – 7a**

**Bellman Ford Algorithm**

**Aim:** To write a python code to implement Bellman Ford Algorithm

**Algorithm:**

1. This step initializes distances from the source to all vertices as infinite and distance to the source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.
2. This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph. Do following for each edge u-v
   * If dist[v] > dist[u] + weight of edge uv, then update dist[v] to
   * dist[v] = dist[u] + weight of edge uv
3. This step reports if there is a negative weight cycle in the graph. Again traverse every edge and do following for each edge u-v   
   ……If dist[v] > dist[u] + weight of edge uv, then “Graph contains negative weight cycle”

**Source Code:**

class Graph:

    def \_\_init\_\_(self, vertices):

        self.V = vertices

        self.graph = []

    def addEdge(self, u, v, w):

        self.graph.append([u, v, w])

    def printArr(self, dist):

        print("Vertex Distance from Source")

        for i in range(self.V):

            print("{0}\t\t{1}".format(i, dist[i]))

    def BellmanFord(self, src):

        dist = [float("Inf")] \* self.V

        dist[src] = 0

        for \_ in range(self.V - 1):

            for u, v, w in self.graph:

                if dist[u] != float("Inf") and dist[u] + w < dist[v]:

                    dist[v] = dist[u] + w

        for u, v, w in self.graph:

            if dist[u] != float("Inf") and dist[u] + w < dist[v]:

                print("Graph contains negative weight cycle")

                return

        self.printArr(dist)

**Sample Input and Output:**

Input:    g = Graph(5)

    g.addEdge(0, 1, -1)

    g.addEdge(0, 2, 4)

    g.addEdge(1, 2, 3)

    g.addEdge(1, 3, 2)

    g.addEdge(1, 4, 2)

    g.addEdge(3, 2, 5)

    g.addEdge(3, 1, 1)

    g.addEdge(4, 3, -3)

    g.BellmanFord(0)

Output:

A black text on a white background

Description automatically generated with low confidence

**Result:**

Thus, Bellman Ford algorithm has been successfully implemented using Python code and the output is verified.