**Distance Vector Routing Protocol (RIP)**

**Program:-**

class GFG

{

static void BellmanFord(int graph[ ][ ], int V, int E, int src)

{

  **// Initialize distance of all vertices as infinite.**

 int []dis = new int[V];

 for (int i = 0; i < V; i++)

 dis[i] = Integer.MAX\_VALUE;

 **// initialize distance of source as 0**

 dis[src] = 0;

 for (int i = 0; i < V - 1; i++)

 {

 for (int j = 0; j < E; j++)

 {

 if (dis[graph[j][0]] != Integer.MAX\_VALUE && dis[graph[j][0]] + graph[j][2] <

 dis[graph[j][1]])

 dis[graph[j][1]] =

 dis[graph[j][0]] + graph[j][2];

 }

 }

 **//check for negative-weight cycles.**

 for (int i = 0; i < E; i++)

 {

 int x = graph[i][0];

 int y = graph[i][1];

 int weight = graph[i][2];

 if (dis[x] != Integer.MAX\_VALUE &&

 dis[x] + weight < dis[y])

 System.out.println("Graph contains negative" +" weight cycle");

 }

 System.out.println("Vertex Distance from Source");

 for (int i = 0; i < V; i++)

 System.out.println(i + "\t\t" + dis[i]);

}

**// Main program**

public static void main(String[] args)

{

 int V = 6; // Number of vertices in graph

 int E = 8; // Number of edges in graph

 /\* Every edge has three values (u, v, w) where the edge is from vertex u to v. And weight of the edge is w.

 int graph[][] = { { 0, 1, 10 }, { 0, 5, 8 },

 { 1, 3, 2 }, {2,1,1}, { 3, 2, -2 },

 { 4, 1, -4 }, { 4, 3, -1 }, {5, 4, 1} };

 BellmanFord(graph, V, E, 0);

 }

}



**Sample Output :-**

Vertex Distance from Source S

S 0

A 5

B 5

C 7

D 9

E 8