**import** java.util.Scanner;

**public class** bellmanford

{ **public int** distance[];

**public int** numb\_vert;

**public static final int** *MAX\_VALUE*=999;

**public** bellmanford(**int** numb\_vert)

{ **this**.numb\_vert = numb\_vert; distance = **new int**[numb\_vert+1]; }

**public void** BellmanfordpEvaluation(**int** source,**int** adj\_matrix[][])

{ **for**(**int** node=1;node<=numb\_vert;node++) distance[node]=*MAX\_VALUE*;

distance[source]=0;

**for**(**int** node=1;node<=numb\_vert-1;node++)

{ **for**(**int** src\_node=1;src\_node<=numb\_vert;src\_node++)

{ **for**(**int** dest\_node=1;dest\_node<=numb\_vert; dest\_node++)

{ **if**(adj\_matrix[src\_node][dest\_node]!=*MAX\_VALUE*)

{ **if**(distance[dest\_node] > distance[src\_node] +

adj\_matrix[src\_node][dest\_node])

distance[dest\_node] = distance[src\_node] +

adj\_matrix[src\_node][dest\_node]; } } } }

**for**(**int** src\_node=1;src\_node<=numb\_vert;src\_node++)

{ **for**(**int** dest\_node=1;dest\_node<=numb\_vert; dest\_node++)

{ **if**(adj\_matrix[src\_node][dest\_node]!=*MAX\_VALUE*)

{ **if**(distance[dest\_node] > distance[src\_node] +

adj\_matrix[src\_node][dest\_node])

{ System.*out*.println("The graph contains negative edge cycle"); } } } }

System.*out*.println("Routing Table for Router " + source+" is");

System.*out*.println("Destination Distance\t");

**for**(**int** vertex=1;vertex<=numb\_vert;vertex++) System.*out*.println(+vertex+"\t\t\t"+distance[vertex]); }

**public static void** main(String args[])

{ **int** numb\_vert=0;

**int** source;

Scanner scan = **new** Scanner(System.*in*);

System.*out*.println("Enter the number of vertices"); numb\_vert = scan.nextInt();

**int** adj\_matrix[][] = **new int**[numb\_vert+1][numb\_vert+1]; System.*out*.println("Enter the adjacency matrix");

**for**(**int** src\_node=1;src\_node<=numb\_vert;src\_node++)

**for**(**int** dest\_node=1;dest\_node<=numb\_vert;dest\_node++)

{ adj\_matrix[src\_node][dest\_node] = scan.nextInt();

**if**(src\_node==dest\_node)

{ adj\_matrix[src\_node][dest\_node]=0;

**continue**; }

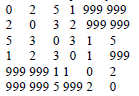
**if**(adj\_matrix[src\_node][dest\_node]==0) adj\_matrix[src\_node][dest\_node]=*MAX\_VALUE*; }

**for**(**int** i=1;i<=numb\_vert;i++)

{ bellmanford bellmanford = **new** bellmanford(numb\_vert); bellmanford.BellmanfordpEvaluation(i,adj\_matrix);

}

scan.close(); } }

**Output 1** –

Enter the number of vertices 6

Enter the adjacency matrix