**import** java.util.\*;

**class** Dijkstra

{

**public** **static** **final** **int** ***V***=5;

// Function to find the vertex with minimum distance value

**public** **int** minDistance(**int** weight[], Boolean myArray[])

{

**int** min = 100, minindex=-1;

/\* System.out.println("Weight array is");

for(int i=0;i<weight.length;i++)

System.out.print(weight[i]+" ");

System.out.println();

System.out.println("my Array is ");

for(int i=0;i<myArray.length;i++)

System.out.print(myArray[i]+" ");

System.out.println();

\*/

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

**if** (myArray[i] == **false** && weight[i] <= min)

{

min = weight[i];

// System.out.println("min is "+min);

minindex = i;

// System.out.println("minumum index is: "+minindex);

}

**return** minindex;

}

// Function to find Shortest Path

**public** **void** dijkstra\_algo(**int** myGraph[][], **int** src)

{

**int** weight[] = **new** **int**[***V***];

Boolean myArray[] = **new** Boolean[***V***];

// Initialize all distances as 100 and array to all False

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

{

weight[i] = 100;

myArray[i] = **false**;

}

//Source vertex has weight of 0

weight[src] = 0;

// Find shortest path for all vertices

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

{

// Choose the minimum distance vertex from the set of vertices

// System.out.println("\n");

**int** num = minDistance(weight, myArray);

myArray[num] = **true**;

// System.out.println("my Array is ");

// for(int i=0;i<myArray.length;i++)

// System.out.print(myArray[i]+" ");

// System.out.println();

**for** (**int** k = 0; k < ***V***; k++)

{

**int** temp = weight[num]+myGraph[num][k];

// Update weight[v] if there is edge from u to v and weight is smaller than current value of weight[v]

**if** (!myArray[k] && myGraph[num][k]!=0 && temp < weight[k])

{

weight[k] = temp;

}

}

}

System.***out***.println();

// To print the Weight of all Vertices

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

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

System.***out***.println("Node is "+i+" distance is "+weight[i]);

}

// MAIN function where program starts Execution

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

{

/\* Let us create the example graph discussed above \*/

**int** myGraph[][] = {{0, 10, 5, 0, 0},{0, 0, 2, 1, 0},{0, 3, 0, 9, 2},{0, 0, 0, 0, 4},{7, 0, 0, 6, 0}};

System.***out***.println("Starting with Vertex S -> Node 0");

Dijkstra q = **new** Dijkstra();

q.dijkstra\_algo(myGraph, 0);

System.***out***.println();

System.***out***.println("Starting with Vertex Z -> Node 4");

Dijkstra w = **new** Dijkstra();

w.dijkstra\_algo(myGraph, 4);

}

}

Starting with Vertex S -> Node 0

Vertex Distance from Source

Node is 0 distance is 0

Node is 1 distance is 8

Node is 2 distance is 5

Node is 3 distance is 9

Node is 4 distance is 7

Starting with Vertex Z -> Node 4

Vertex Distance from Source

Node is 0 distance is 7

Node is 1 distance is 15

Node is 2 distance is 12

Node is 3 distance is 6

Node is 4 distance is 0

