**7. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. Write the program in Java.**

import java.util.\*;

public class DijkstrasClass {

final static int MAX = 20;

final static int infinity = 9999;

static int n; // No. of vertices of G

static int a[][]; // Cost matrix

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

public static void main(String[] args) {

ReadMatrix();

int s = 0; // starting vertex

System.out.println("Enter starting vertex: ");

s = scan.nextInt();

Dijkstras(s); // find shortest path

}

static void ReadMatrix() {

a = new int[MAX][MAX];

System.out.println("Enter the number of vertices:");

n = scan.nextInt();

System.out.println("Enter the cost adjacency matrix:");

for (int i = 1; i <= n; i++)

for (int j = 1; j <= n; j++)

a[i][j] = scan.nextInt();

}

static void Dijkstras(int s) {

int S[] = new int[MAX];

int d[] = new int[MAX];

int u, v;

int i;

for (i = 1; i <= n; i++) {

S[i] = 0;

d[i] = a[s][i];

}

S[s] = 1;

d[s] = 1;

i = 2;

while (i <= n) {

u = Extract\_Min(S, d);

S[u] = 1;

i++;

for (v = 1; v <= n; v++) {

if (((d[u] + a[u][v] < d[v]) && (S[v] == 0)))

d[v] = d[u] + a[u][v];

}

}

for (i = 1; i <= n; i++)

if (i != s)

System.out.println("The distance from"+ s + "to "+ i + ":" + d[i]);

}

static int Extract\_Min(int S[], int d[]) {

int i, j = 1, min;

min = infinity;

for (i = 1; i <= n; i++) {

if ((d[i] < min) && (S[i] == 0)) {

min = d[i];

j = i;

}

}

return (j);

}

}

**OUTPUT**



