Akanksha Koddra. CN - Lab 8.

1BM18CS007.

Date: Dec 7, 2020. Batch A1.

Date:
P. No:

Implement dijkstra's algorithm to compute shortest path through graph :

```
import java.util.*;
public class Dijkstra
{
    public static void main (String[] args)
    {
        System.out.println (" Enter no of vertices ");
        Scanner sc = new Scanner (System.in);
        int n = sc.nextInt();
        int arr[][] = new int [n][n];
        System.out.println (" Enter adjacency matrix ");
        for (int i=0; i<n; i++)
        {
            for (int j=0; j<n; j++)
            {
                arr[i][j] = sc.nextInt();
                if (arr[i][j] == 0)
                    arr[i][j] = 999;
            }
        }
        System.out.println (" Enter source vertex ");
        int start = sc.nextInt();
        dijkstra (n, arr, start);
    }
    public static void dijkstra (int n, int arr[][], int start)
    {
        int visited[n], parent = new int [n];
```

Date:
P. No:

```
int parent[] = new int[n];
int distance[] = new int[n];
int count=0;
distance[start]=0;
for(int i=0;i<n;i++)
{
    visited[i]=0;
    parent[i]=start;
    if(i!=start)
        distance[i]=arr[start][i];
}
parent[start]=-1;
visited[start]=1;
while(count<n-1)
{
    int min=999, index=0, i;
    for(i=0;i<n;i++)
    {
        if(visited[i]!=1 && distance[i]<min)
        {
            min=distance[i];
            index=i;
        }
    }
    visited[index]=1;
    for(int j=0;j<n;j++)
    {
        if(visited[j]!=1 && arr[index][j]!=999 && (
        distance[index]+arr[index][j]<distance[j]))
        {
```

```
                distance[j] = distance[index] + arr[index][j];
                parent[j] = index;
            }
        }

        count++;
    }
    System.out.println("Distance & path from source are");
    for(int i=0; i<n; i++)
    {
        System.out.print(i + ":" + distance[i] + "    " + start
            + "->");
        printPath(parent, i);
        System.out.println("\n");
    }
}

public static void printPath(int parent[], int j)
{
    if(parent[j] == -1)
        return;
    printPath(parent, parent[j]);
    System.out.print(j + "->");
}
}
```