4. From a given source vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

```
import java.util.Scanner;
public class Dijakstras {
        static int a[][];
        static int n;
        public static void main(String args[])
        {
         Scanner in = new Scanner(System.in);
         System.out.println("Enter the number of vertices:");
         n = in.nextInt();
         System.out.println("Enter the cost adjacency matrix");
         a = new int[n][n];
         for (int i=0;i<n;i++)
         {
          for (int j=0;j<n;j++)
          {
           a[i][j] = in.nextInt();
          }
         }
         System.out.println("\nEnter the source vertex");
         int s=in.nextInt();
         Dijkstra(s);
         in.close();
        }

        public static void Dijkstra(int s)
        {
         int visited[] = new int[n];
         int d[] = new int[n];
         int i,u,v;
         for(i=0;i<n;i++)
         {
          visited[i]=0;
          d[i] = a[s][i];
         }
         visited[s]=1;
         d[s]=0;
         i=1;
         while(i<=n-1)
         {
          u = Extract_Min(visited,d);
          visited[u]=1;
          i++;
          for(v=0;v<n;v++)
          {
           if((d[u]+a[u][v]<d[v]) && visited[v]==0)
```

```
                d[v]= d[u]+a[u][v];
            }
          }
          for(i=0;i<n;i++)
          {
             if(i!=s)
                System.out.println(s+"->"+i+":"+d[i]);
          }
        }

        public static int Extract_Min(int visited[],int d[])
        {
          int i,j=0,min=999;
          for(i=0;i<n;i++)
            if(d[i]<min && visited[i]==0)
            {
              min = d[i];
              j=i;
            }
            return j;
        }
}
```