```
/* Bellman-Ford */
import java.util.*;
public class Belmanford
 {
   private  int D[];
   private  int n;
   public static final int max_value=999;
   public Belmanford(int n)
   {
      this.n=n;
      D=new int[n+1];
   }
   public void shortest(int s,int a[][])
   {
      for(int i=1;i<=n;i++)
      {
         D[i]=max_value;
      }
      D[s]=0;
      for(int k=1;k<=n-1;k++)
      {
         for(int i=1;i<=n;i++)
         {
            for(int j=1;j<=n;j++)
            {
               if(a[i][j]!=max_value)
               {
                  if(D[j]>D[i]+a[i][j])

                     D[j]=D[i]+a[i][j];
               }
            }
         }
      }
      for (int i=1;i<=n;i++)
      {
         for (int j=1;j<=n;j++)
         {
          if(a[i][j]!=max_value)
          {
              if(D[j]>D[i]+a[i][j])
              {
                 System.out.println("the graph contains -ve edge cycle");
                 return;
              }
```

```
            }
           }
        }
        for (int i=1;i<=n;i++)
        {
           System.out.println("distance of source"+s+"to"+i+"is"+D[i]);
        }
    }
        public static void main(String[] args)
{
        int n=0,s;
        Scanner sc=new Scanner(System.in);
        System.out.println("enter the no.of values");
        n=sc.nextInt();
        int a[][]=new int [n+1][n+1];
        System.out.println("enter the weighted matrix:");
        for (int i=1;i<=n;i++)
        {
           for (int j=1;j<=n;j++)
           {
              a[i][j]=sc.nextInt();
              if(i==j)
              {
                 a[i][j]=0;
                 continue;
              }
              if(a[i][j]==0)
                 a[i][j]=max_value;
           }
        }
    System.out.println("enter the source vertex:");
    s=sc.nextInt();
    Belmanford b=new Belmanford(n);
    b.shortest(s,a);
    sc.close();
    }

}
```

Output1

```
enter the no.of values
4
enter the weighted matrix:
0 999 999 999
5 0 3 4
```