WEEK 8

PROGRAM-1:

AIM:

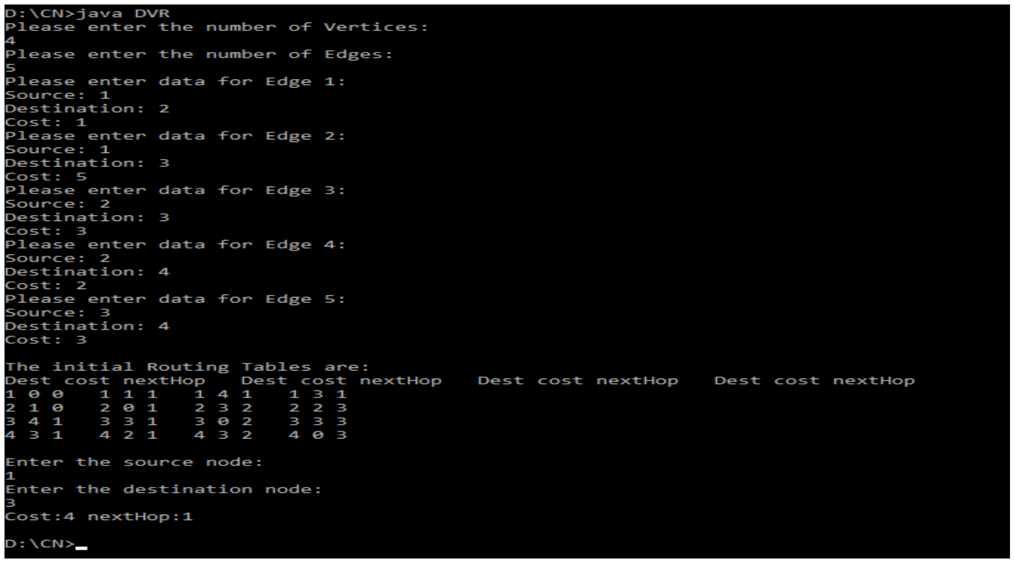
To implement Distance Vector Routing Algorithm.

Program:

```
import java.io.*;
public class DVR
{
 static int graph[][];
 static int via[][];
 static int rt[][];
 static int v;
 static int e;

 public static void main(String args[]) throws IOException
 {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

  System.out.println("Please enter the number of Vertices: ");
  v = Integer.parseInt(br.readLine());

  System.out.println("Please enter the number of Edges: ");
  e = Integer.parseInt(br.readLine());

  graph = new int[v][v];
  via = new int[v][v];
  rt = new int[v][v];
  for(int i = 0; i < v; i++)
   for(int j = 0; j < v; j++)
   {
    if(i == j)
     graph[i][j] = 0;
   else
    graph[i][j] = 9999;
   }

  for(int i = 0; i < e; i++)
  {
   System.out.println("Please enter data for Edge " + (i + 1) + ":");
   System.out.print("Source: ");
   int s = Integer.parseInt(br.readLine());
   s--;
   System.out.print("Destination: ");
   int d = Integer.parseInt(br.readLine());
   d--;
   System.out.print("Cost: ");
   int c = Integer.parseInt(br.readLine());
   graph[s][d] = c;
   graph[d][s] = c;
  }

  dvr_calc_disp("The initial Routing Tables are: ");
  System.out.println("Enter the source node:");
  int source=Integer.parseInt(br.readLine());
  System.out.println("Enter the destination node:");
  int dest=Integer.parseInt(br.readLine());
  System.out.println("Cost:"+rt[source-1][dest-1]+" nextHop:"+via[source-1][dest-
1]+1);
 }

 static void dvr_calc_disp(String message)
 {
  System.out.println();
  init_tables();
  update_tables();
  System.out.println(message);
  print_tables();
  System.out.println();
 }
 static void update_table(int source)
 {
  for(int i = 0; i < v; i++)
  {
   if(graph[source][i] != 9999)
   {
   int dist = graph[source][i];
   for(int j = 0; j < v; j++)
   {
    int inter_dist = rt[i][j];
    if(via[i][j] == source)
     inter_dist = 9999;
    if(dist + inter_dist < rt[source][j])
    {
     rt[source][j] = dist + inter_dist;
     via[source][j] = i;
    }
   }
   }
  }
 }

 static void update_tables()
 {
  int k = 0;
  for(int i = 0; i < 4*v; i++)
  {
   update_table(k);
   k++;
   if(k == v)
   k = 0;
  }
 }

 static void init_tables()
 {
  for(int i = 0; i < v; i++)
  {
   for(int j = 0; j < v; j++)
   {
   if(i == j)
   {
    rt[i][j] = 0;
    via[i][j] = i;
   }
   else
   {
    rt[i][j] = 9999;
    via[i][j] = 100;
   }
   }
  }
 }
 static void print_tables()
 {
  for(int i=0;i<v;i++){
    System.out.print("Dest cost nextHop"+"   ");
  }
  System.out.println();
```

```
  for(int i = 0; i < v; i++)
  {
   for(int j = 0; j < v; j++)
   {
    System.out.print(i+1+" "+ rt[i][j] +" "+via[i][j]+1+"   ");
   }
   System.out.println();
  }
 }
}
```

Output:



```
D:\CN>java DVR
Please enter the number of Vertices:
4
Please enter the number of Edges:
5
Please enter data for Edge 1:
Source: 1
Destination: 2
Cost: 1
Please enter data for Edge 2:
Source: 1
Destination: 3
Cost: 5
Please enter data for Edge 3:
Source: 2
Destination: 3
Cost: 3
Please enter data for Edge 4:
Source: 2
Destination: 4
Cost: 2
Please enter data for Edge 5:
Source: 3
Destination: 4
Cost: 3

The initial Routing Tables are:
Dest cost nextHop   Dest cost nextHop    Dest cost nextHop    Dest cost nextHop
1 0 0   1 1 1   1 4 1   1 3 1
2 1 0   2 0 1   2 3 2   2 2 3
3 4 1   3 3 1   3 0 2   3 3 3
4 3 1   4 2 1   4 3 2   4 0 3

Enter the source node:
1
Enter the destination node:
3
Cost:4 nextHop:1

D:\CN>_
```

PROGRAM-2:

AIM:

To implement Distance Vector Routing Algorithm in TCL.

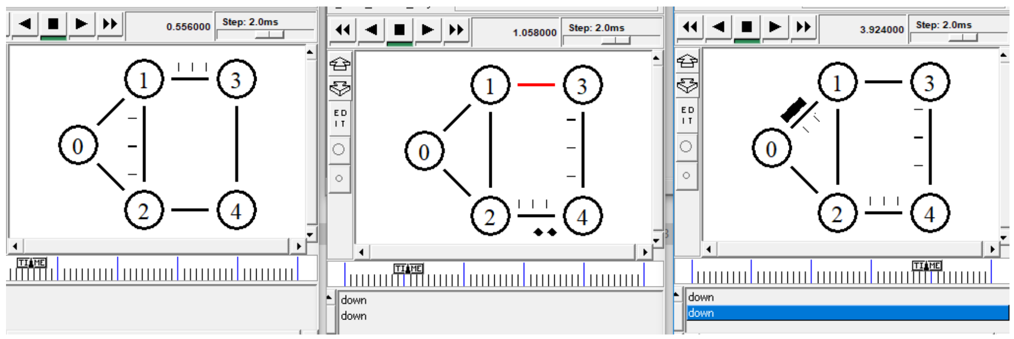
Program:

```
set ns [new Simulator]
set nf [open ns.nam w]
$ns namtrace-all $nf
set tr [open ns.tr w]
$ns trace-all $tr
proc finish {} {
      global nf ns tr
```

```
        $ns flush-trace
        close $tr
        exec nam out.nam &
        exit 0
        }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
$ns duplex-link $n1 $n2 10Mbps 10ms DropTail
$ns duplex-link-op $n1 $n2 orient down
$ns duplex-link $n3 $n4 10Mbps 10ms DropTail
$ns duplex-link-op $n3 $n4 orient down
$ns duplex-link $n2 $n4 10Mbps 10ms DropTail
$ns duplex-link-op $n2 $n4 orient right
$ns duplex-link $n1 $n3 10Mbps 10ms DropTail
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link $n0 $n1 10Mbps 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-up
$ns duplex-link $n0 $n2 10Mbps 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

```
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $tcp $sink
$ns connect $udp $null
$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3
$ns rtproto DV
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"
$ns run
```

Output:


0.556000
Step: 2.0ms
1.058000
Step: 2.0ms
3.924000
Step: 2.0ms
0
1
2
3
4
TIME
down
down
down
down