PART B

1.BELLMANFORD ALGORITHM

CN LAB INTERNALS -01

```
package Bellman_ford;
import java.util.*;
public class Main{
        static int n,dest;
        static double[] prevDistanceVector,distanceVector;
        static double[][] adjacencyMatrix;
        public static void main(String[] args) {
        // TODO code application logic here
                Scanner scanner = new Scanner(System.in);
                System.out.println("Enter number of nodes");
                n = scanner.nextInt();
                adjacencyMatrix = new double[n][n];
                System.out.println("Enter Adjacency Matrix (Use 'Infinity' for
No Link)");
                for (int i = 0; i< n; i++)
                {
                        for (int j = 0; j < n; j++)
                                adjacencyMatrix[i][j] = scanner.nextDouble();
                }
                System.out.println("Enter destinationvertex");
                dest = scanner.nextInt();
                distanceVector = new double[n];
                for (int i = 0; i< n; i++)
                        distanceVector[i] = 99;
                distanceVector[dest - 1] = 0;
                bellmanFordAlgorithm();
                System.out.println("Distance Vector");
                for (int i = 0; i< n; i++) {
                        if (i == dest - 1)
                                continue;
                        else
                                System.out.println("Distance from " + (i + 1) +
" is " + distanceVector[i]);
                }
                System.out.println();
                System.out.println("enter the vertices between which link is
broken (u and v)");
                int u = scanner.nextInt();
                int v = scanner.nextInt();
                adjacencyMatrix[u-1][v-1] = 99;
                for (int i = 0; i< n; i++)
                        distanceVector[i] = 99;
                distanceVector[dest - 1] = 0;
                bellmanFordAlgorithm();
                System.out.println("Distance Vector");
                for (int i = 0; i< n; i++) {
                        if (i == dest - 1)
                                continue;
                        else
                                System.out.println("Distance from " + (i + 1) +
```

```
" is " + distanceVector[i]);
                }
        }

        static void bellmanFordAlgorithm()
        {
                for (int i = 0; i< n ; i++)
                {
                        for (int j = 0; j < n; j++)
                        {
                                double min = distanceVector[j];
                                for (int k = 0; k < n; k++)
                                {
                                        if (min >adjacencyMatrix[j][k] +
distanceVector[k] && adjacencyMatrix[j][k]!=99)
                                                min = adjacencyMatrix[j][k] +
distanceVector[k];
                                }
                                distanceVector[j] = min;
                        }
                }
        }
}
```

```
OUTPUT:
Enter number of nodes
6
Enter Adjacency Matrix (Use 'Infinity' for No Link)
0 3 2 5 99 99
3 0 99 1 4 99
2 99 0 2 99 1
5 1 2 0 3 99
99 4 99 3 0 2
99 99 1 99 2 0
Enter destinationvertex
6
Distance Vector
Distance from 1 is 3.0
Distance from 2 is 4.0
Distance from 3 is 1.0
Distance from 4 is 3.0
Distance from 5 is 2.0

enter the vertices between which link is broken (u and v)
1
3
Distance Vector
Distance from 1 is 7.0
Distance from 2 is 4.0
Distance from 3 is 1.0
Distance from 4 is 3.0
Distance from 5 is 2.0
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

PART A
1.THREE NODES

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc, argv);

  NodeContainer nodes;
  nodes.Create(3);

  PointToPointHelper point;
  point.SetDeviceAttribute("DataRate",StringValue("1Mbps"));
  point.SetChannelAttribute("Delay",StringValue("2ms"));


  NetDeviceContainer devices;
  devices=point.Install(nodes.Get(0),nodes.Get(1));
  NetDeviceContainer devices1;
  devices1=point.Install(nodes.Get(1),nodes.Get(2));

  InternetStackHelper stack;
  stack.Install(nodes);

  Ipv4AddressHelper add;
  add.SetBase("10.1.1.0","255.255.255.0");

  Ipv4InterfaceContainer interface=add.Assign(devices);
  Ipv4InterfaceContainer interface1=add.Assign(devices1);

  UdpEchoServerHelper echo(9);
  ApplicationContainer serverApps=echo.Install(nodes.Get(2));
  serverApps.Start(Seconds(1.0));
  serverApps.Stop(Seconds(10.0));

  UdpEchoClientHelper echoClient(interface.GetAddress(1),9);
  echoClient.SetAttribute("MaxPackets",UintegerValue(1));
  echoClient.SetAttribute("Interval",TimeValue(Seconds(1.0)));
  echoClient.SetAttribute("PacketSize",UintegerValue(1024));

  ApplicationContainer clientApps=echoClient.Install(nodes.Get(0));
  clientApps.Start(Seconds(2.0));
  clientApps.Stop(Seconds(10.0));
```

```
  AsciiTraceHelper ascii;
  csma.EnableAsciiAll(ascii.CreateFileStream ("first.tr"));

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

2.PING

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/internet-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("CsmaPingExample");

static void PingRtt (std::string context, Time rtt)
{
  std::cout << context << " " << rtt << std::endl;
}

int main (int argc, char *argv[])
{

  CommandLine cmd;
  cmd.Parse (argc, argv);

  NS_LOG_INFO ("Create nodes.");
  NodeContainer c;
  c.Create (6);

  NS_LOG_INFO ("Build Topology.");
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (10000)));
  csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (0.2)));
  NetDeviceContainer devs = csma.Install (c);

  NS_LOG_INFO ("Add ip stack.");
  InternetStackHelper ipStack;
  ipStack.Install (c);

  NS_LOG_INFO ("Assign ip addresses.");
  Ipv4AddressHelper ip;
  ip.SetBase ("192.168.1.0", "255.255.255.0");
  Ipv4InterfaceContainer addresses = ip.Assign (devs);
```

```
  NS_LOG_INFO ("Create Sink.");

  NS_LOG_INFO ("Create Applications.");
  uint16_t port = 9;
  ApplicationContainer app;
  PacketSinkHelper sink ("ns3::UdpSocketFactory",Address (InetSocketAddress
(Ipv4Address::GetAny (), port)));
  app = sink.Install (c.Get (3));
  app.Start (Seconds (0.0));

  NS_LOG_INFO ("Create pinger");

  V4PingHelper ping = V4PingHelper (addresses.GetAddress (3));
  NodeContainer pingers;
  pingers.Add (c.Get (1));
  pingers.Add (c.Get (2));

  ApplicationContainer apps;
  apps = ping.Install (pingers);
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (5.0));

  Config::Connect ("/NodeList//ApplicationList//$ns3::V4Ping/Rtt",MakeCallback
(&PingRtt));

  NS_LOG_INFO ("Run Simulation.");

  AsciiTraceHelper ascii;
  csma.EnableAsciiAll (ascii.CreateFileStream ("ping1.tr"));

  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---------------------------
```