I. Source code

In this source code part, there are total 4 classes included:

1. Application class;
2. Router class;
3. RouterTable class;
4. FindPath class;

Application.java:

```
import java.io.*;
import java.util.ArrayList;
import java.util.Scanner;
public class Application {
   public static void main(String [] args){

       String fileName;
       String li,resp;
       int [] ar,ar2,ar3,ar4,ar5,ar6;
       int rotNum=0, i=0,j=0, source,dest,source1,dest1,fRouter;
       ArrayList<int []> tableArray=new ArrayList<int []>();
       Router soc,det,soc1,det1,downRouter;
       try{
          //input file
          Scanner sc1=new Scanner(System.in);
          System.out.println("Welcome! ");
          System.out.println("Enter the input file name: ");
          fileName=sc1.nextLine();

          //upload the file into matrix
          Scanner sc2=new Scanner(new BufferedReader(new
FileReader(fileName)));
          while(sc2.hasNextLine()){
             sc2.nextLine();
             rotNum++;
```

```
            }
            int[][] matx= new int[rotNum][rotNum];
            Scanner sc3=new Scanner(new BufferedReader(new
FileReader(fileName)));
            while(sc3.hasNext()){
                  li=sc3.next();
                  matx[i][j]=Integer.parseInt(li);
                  if(j<rotNum-1){
                     j++;
                  }
                  else{
                      j=0;
                      i++;
                  }
               }

            //output the matrix

      System.out.println("-----------------------------------");
            System.out.println("The initial matrix is as follows:");
            for(int m=0;m<rotNum;m++){
               System.out.println();
               for(int n=0;n<rotNum;n++){
               System.out.print(matx[m][n]+"\t ");
               }
             }

            //initial the routers

            for(int m=0;m<rotNum;m++){
               ar=new int [rotNum];
               for(int n=0;n<rotNum;n++){
                  ar[n]=matx[m][n];
                  }
               tableArray.add(ar);
            }
            //Create a content of routers
            RouterTable rTable=new RouterTable();
            for(int m=0;m<rotNum;m++){
               Router re=new
Router(m,tableArray.get(m));//initialize each router
```

```
            rTable.add(re);
         }
         for(int m=0;m<rotNum;m++){
            rTable.pop(m).findNeigb(rTable);//find neighbor of
each router
         }
         for(int m=0;m<rotNum;m++){
            rTable.pop(m).send(rTable);//start update the table of
each router
         }

         //output the number of iterations
         System.out.println();

   System.out.println("-----------------------------------");
         System.out.print("The required number of iterations n = ");
         System.out.print(rTable.pop(0).getIte());

         //output the final matrix
         System.out.println();

   System.out.println("-----------------------------------");
         System.out.println("The final matrix computed by the DV
algorithm is as follows: ");
         for(int m=0;m<rotNum;m++){
            for(int k=0;k<rTable.pop(m).getTable().size();k++){
               if(rTable.pop(m).getTable().get(k)<999){

   System.out.print(rTable.pop(m).getTable().get(k)+"\t ");
               }else
               {
                  System.out.print("NA"+"\t ");
               }
            }
            System.out.println();
         }

         //output the shortest path repeatedly
         do{
            //make the user input the source and destination
            System.out.println();
```

```
   System.out.println("----------------------------------");
            System.out.println("Enter the source and destination
nodes: ");
            Scanner sc4=new Scanner(System.in);
            source=sc4.nextInt();
            dest=sc4.nextInt();
            soc=rTable.pop(source-1);
            det=rTable.pop(dest-1);
            FindPath fp=new FindPath();
            fp.sPath(soc, det);

            //output the length of path from source to destination

   System.out.println("----------------------------------");
            System.out.print("The length of this path is :");
            System.out.println("\t"+fp.pLength(soc,det));

            //provide an other chance to find the path
            System.out.println("Another source-destination
pair ?(yes/no)");
            Scanner sc5=new Scanner (System.in);
            resp=sc5.next();
         }while(resp.equals("yes"));

         //make the user enter the router which they want to set down

   System.out.println("----------------------------------");
         System.out.println("Enter the number of the router whose
failure is to be simulated:");
         Scanner sc6=new Scanner(System.in);
         fRouter=sc6.nextInt();
         downRouter=rTable.pop(fRouter-1);
         downRouter.setDown(rTable);

         //output the number of iterations
         System.out.println();

   System.out.println("----------------------------------");
         System.out.print("The required number of iterations n = ");
         System.out.print(rTable.pop(0).getIte());
```

```
          //output the final matrix
          System.out.println();

   System.out.println("-----------------------------------");
          System.out.println("The final matrix computed by the DV
algorithm is as follows: ");
          for(int m=0;m<rotNum;m++){
             for(int k=0;k<rTable.pop(m).getTable().size();k++){
                if(rTable.pop(m).getTable().get(k)<999){

   System.out.print(rTable.pop(m).getTable().get(k)+" \t");
                }else
                {
                System.out.print("NA"+" \t");//set the distance of
unreachable router to NA
                }
             }
             System.out.println();
          }

          //output the shortest path repeatedly
          do{
             System.out.println();

   System.out.println("-----------------------------------");
             System.out.println("Enter the source and destination
nodes: ");
             Scanner sc4=new Scanner(System.in);
             source1=sc4.nextInt();
             dest1=sc4.nextInt();
             soc1=rTable.pop(source1-1);
             det1=rTable.pop(dest1-1);
             FindPath fp1=new FindPath();
             fp1.sPath(soc1, det1);

             //output the length of path from source to destination

   System.out.println("-----------------------------------");
             System.out.print("The length of this path is :");
             System.out.println("\t"+fp1.pLength(soc1, det1));
```

```
            //provide an other chance to find the path
            System.out.println("Another source-destination
pair ?(yes/no)");
            Scanner sc5=new Scanner (System.in);
            resp=sc5.next();
         }while(resp.equals("yes"));
         System.out.println();

   System.out.println("-----------------------------------");
         System.out.println("Thanks for using");


      }catch(IOException io){
         System.out.println(io);
      }
      }
}
```

Router class.java:

```
import java.util.ArrayList;

public class Router {
   ArrayList <Integer> table=new ArrayList<Integer>();
   ArrayList <Integer> copTb;
   ArrayList <Router> neigb=new ArrayList<Router>();

   Router [] nextHop;
   public static int iter=0;

   int routName;
   boolean flag;

   //constructor of Router class
   public Router(int rtn, int[] ary){
      routName=rtn;
      nextHop=new Router[ary.length];
      for(int i=0;i<ary.length;i++)
```

```
        {
            table.add(ary[i]);//add the initial record of router
distance
        }

    };

    //find the neighbor of router
    public void findNeigb(RouterTable rt){
        for(int i=0;i<table.size();i++)
            {
                //Check the initial distance to find the neighbor
                if(table.get(i)!= 999 && table.get(i)!=0)
                {
                    neigb.add(rt.pop(i));//record the neighbor router
                    nextHop[i]=rt.pop(i);//update the nexthop
                }
            }

    }

    //send the record to the neighbor router
    public void send(RouterTable rt){
        for(int i=0;i<neigb.size();i++){
            neigb.get(i).updt(routName,table,rt);
        }
    }

    //update the routing table by using the received record
    public void updt(int rName, ArrayList<Integer> tb, RouterTable
rt){
        copTb=new ArrayList<Integer>();//copy the received record
        flag=false;
        for(int i=0;i<tb.size();i++)
        {
            copTb.add(tb.get(i));
        }
        for (int i=0;i<tb.size();i++)
        {
            copTb.set(i,copTb.get(i)+table.get(rName));
            //check if update is needed
```

```
            if(copTb.get(i)<table.get(i)){
                table.set(i,copTb.get(i));
                nextHop[i]=rt.pop(rName);
                flag=true;//if the record is updated, set flag to true
            }else if(copTb.get(i)>table.get(i) &&
nextHop[i]==rt.pop(rName)){
                table.set(i, copTb.get(i));
                flag=true;//if the record is updated, set flag to true
            }

        }
        //if the table is updated, then call the send method
        if(flag==true){
            iter++;
            this.send(rt);
        }
    }

    //return the record distance of the router
    public ArrayList<Integer> getTable()
    {
        for(int i=0;i<table.size();i++)
        {
            if(table.get(i)>999)
            {
                table.set(i,999);
            }
        }
        return table;
    }

    //return the iteration times
    public int getIte(){
        return iter;
    }

    //return the nextHop of destination router
    public Router getNextHop(int i){
        return nextHop[i];
    }
```

```
    //set this router down
    public void setDown(RouterTable rt){
        iter=0;
        for(int i=0;i<table.size();i++)
        {
            table.set(i, 999);
        }
        this.send(rt);
    }
}
```

RouterTable.java:

```
import java.util.ArrayList;


public class RouterTable {
    ArrayList <Router> router=new ArrayList<Router>();

    //the constructor of RouterTable
    public RouterTable(){

    }

    //add a router into the content
    public void add(Router rot)
    {
        router.add(rot);
    }

    //return the i-th router
    public  Router pop(int i)
    {
        return router.get(i);
    }
}
```

FindPath.java:

```
import java.util.ArrayList;


public class FindPath {
   Router sorc, dest;
   RouterTable rTable;
   int rNum1, rNum2;
   int count=0;
   ArrayList<Integer> path=new ArrayList<Integer>();

   //the constructor of FindPath
   public FindPath(){
   }

   //print the path from the source to the destination
   public void sPath(Router source, Router destination){
      count++;
      rNum1=source.routName;
      rNum2=destination.routName;
      System.out.print(rNum1+1+"->");
      if(source.getNextHop(rNum2).equals(destination)){
         System.out.println(rNum2+1);
      }
      else{
         sPath(source.getNextHop(rNum2),destination);
      }

   }

   //return the path length between source and destination
   public int pLength(Router source, Router destination){
      return source.table.get(destination.routName);
   }
}
```