

Catalog

Catalog.....	1
1) Documented source code and user manual.....	2
1.1 User manual	2
2) Introduction to the Distance Vector algorithm	6
3) Explanation of each class and each function's functionality	6
3.1 routerTable.class	6
3.2 Main.class.....	7
3.3 loadfile.class	9
3.4 finalTableForSelectedRouter.class	10
3.5 finalTable.class	11
3.6 LinkFailure.class.....	11
3.7 shortestPath.class	12
4) Details of finding a shortest path	14
5) Problems and reasons	16
6) Test User Cases	17
6.1 User case 1(Undirected graph).....	17
6.2 User case 2(Undirected graph).....	19
6.3 User case 3(Undirected graph).....	21
6.4 User case 4(Undirected graph).....	23
6.5 User case 5(Undirected graph).....	25
6.6 User case 6(Undirected graph).....	27
6.7 User case 7(Directed graph)	29
6.8 User case 8(Directed graph)	31
6.9 User case 9(Directed graph)	33
6.10 User case 10(Error processing).....	35

1) Documented source code and user manual

The total source code contains seven parts:

main.java

routerTable.java

loadfile.java

finalTableForSelectedRouter.java

finalTable.java

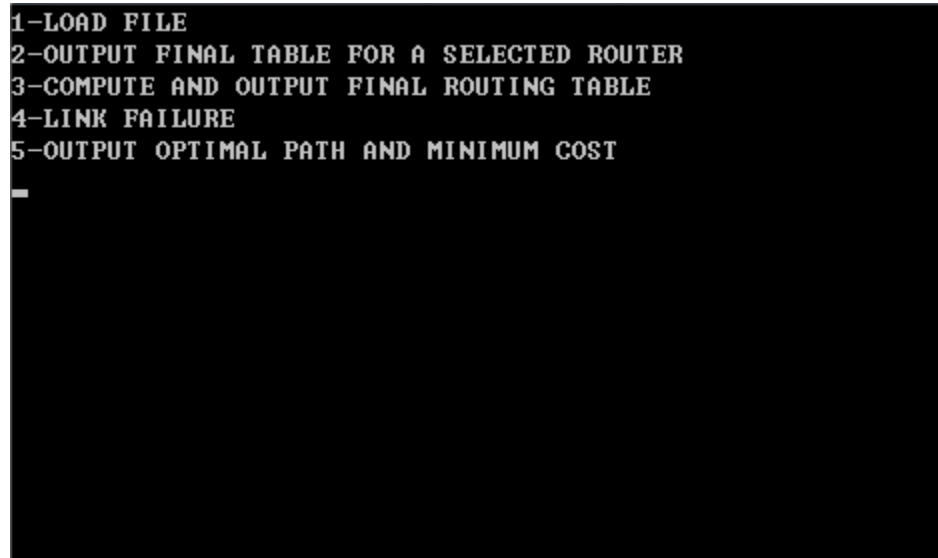
linkFailure.java

shortestPath.java

And all the source code has been attached with this report.

1.1 User manual

When the program runs, there is a menu. As shown below:



```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
_
```

You can enter 1, 2, 3, 4, 5 to choose different function, firstly you should enter 1 to load a file.

Input: 1(function of loading file)

In this function, you will be asked to input the name of txt file, if the file is in the same directory as executable file, you can just enter the file name, such as 'testfile.txt', else if the txt file is in the other directory, you should enter the absolute path of the file, such as 'C:\testfile.txt'.

The program will output the original routing table in the file. As shown below:

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
1
Please load original routing table data file:
testfile.txt
Original routing table is as follows:
  0  2  5  1 -1
  2  0  8  7  9
  5  8  0 -1  4
  1  7 -1  0  2
 -1  9  4  2  0
```

Input: 2(function of outputting a final routing table for a selected router)

In this function, you will be asked to enter a router's number, such as '2', the router is named as 1, 2, 3, ..., the largest number should be the total number of routers, and you cannot enter a number larger than that. Then the program will output the final routing table of a selected router. As shown below:

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
2
Please input a router number
2
The final table of this selected router is:
[2, 0, 7, 3, 5]
```

Input: 3(function of outputting a final routing table for all the routers)

In this function, you needn't to enter anything, and the program will output a final routing table automatically.

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
3
Final routing table computed by Distance Vector algorithm is:
  0  2  5  1  3
  2  0  7  3  5
  5  7  0  6  4
  1  3  6  0  2
  3  5  4  2  0

```

Input: 4(function of disable any router)

In this function, you need to enter the router number which you want to fail it, such as '2', which means this router #4 will break down, and all corresponding router cannot connect to it. Also, you cannot enter a number larger than the total number of routers.

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
4
Please input the number of failed router:
2
Router failed.

```

Input: 5(function of outputting a shortest path between two routers)

In this function, you will be asked to enter the router numbers of source, destination and bypass(not necessary), and the numbers should be separate by blank, such as '1 5 3', or you can just enter '1 5' without the bypass router number.

The program will output the shortest path from source to destination, and if there is a bypass, the path would also via bypass. And the program will also output the total cost from source to destination.

source: 1; destination: 5; bypass: 3

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 5 3
The shortest path from 1 to 5 (bypass 3)is 1-->3-->5, the total cost is 9
```

source:1; destination: 5

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 5
The shortest path from 1 to 5 is 1-->4-->5, the total cost is 3
```

2) Introduction to the Distance Vector algorithm

Let $D_x(y)$ represent the distance from router x to router y , and $c(x, y)$ represent the cost of edge $c(x, y)$, if x and y is not directly connected, $c(x, y)$ is infinity.

To initialize all the value of $D_x(y)$, the algorithm can do the operations below:

if router w is a neighbor of x , $D_x(w)$ is $c(x, w)$, else if router v is not a neighbor of x , $D_x(v)$ is infinity; and $D_x(x)$ is always 0.

And to get the best route from x to y , which means $D_x(y)$ is smallest, we will do the update for $O(VE)$ times (V represents the number of vertices and E represents the number of edges):

$$D_x(y) = \min\{c(x, v) + D_v(y)\}$$

The formula formally do these operations, from time to time, each node sends a copy of its distance vector to each of its neighbors. When a node x receives a new distance vector from any of its neighbor v , it will update its own distance vector by adding $D_v(y)$ and $c(x, v)$, if the route from x to y via v is less cost, $D_x(y)$ will be updated by $D_v(y) + c(x, v)$. Iteratively, we can get the least-cost path from x to y , this is the best route from x to y .

3) Explanation of each class and each function's functionality

3.1 *routerTable.class*

This is the data structure of the program, I use a ArrayList to store the input txt file, since the router table can be any size, I cannot store it in a two-dimensional array without knowing the size of it.

```
//store the routing table in a arraylist
ArrayList<Integer> Table = new ArrayList<Integer>();

routerTable(ArrayList<Integer> table) {
    super();
    Table = table;
}
```

Also, I define a update function to update the ArrayList, though calling this method, the data can be stored in the ArrayList.

```
//update the value of routing table in the arraylist
public void update(routerTable t, ArrayList<Integer> a){
    t.Table = a;
}
```

3.2 Main.class

In main.java, the class main contains entrance of whole program, it will display a menu via the code below:

```
System.out.println("1-LOAD FILE");
System.out.println("2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER");
System.out.println("3-COMPUTE AND OUTPUT FINAL ROUTING TABLE");
System.out.println("4-LINK FAILURE");
System.out.println("5-OUTPUT OPTIMAL PATH AND MINIMUM COST");
```

By entering a number, the main function will enter a switch statement, each number represents corresponding function in the menu, and in each case, the program will use different class to realize the functionality.

In case 1, the main function will use loadfile.class and call the method of update, use loadFile.class and call the method of LoadFile to load a file and store it in the data structure of program, if the inputted original routing table is valid, then use for loop to output the routing table in certain format.

```
case 1:
    Router.update(Router, null);
    lf.LoadFile(Router.Table);
    Router.update(Router, lf.temp);
    if(Router.Table!= null&&Router.Table.size() != 0){
        //system prompt
        System.out.println("Original routing table is as follows:");
        for(int i = 0; i < lf.routerNumber; i++){
            for(int j = 0; j < lf.routerNumber; j++)
                System.out.printf("%4s",Router.Table.get(lf.routerNumber*i+j));
```

```

        System.out.println();
    }
}
else
    System.out.println("The original routing table has not been inputted.");
break;

```

In case 2, if the original routing table has been inputted, the main function will use finalTableForSelectedRouter.class, and call the method Seclected to compute the routing table, and print it.

```

case 2:
    if(Router.Table!= null&&Router.Table.size() != 0){
        fs.Seclected(Router.Table, lf.routerNumber);
        System.out.println("The final table of this selected router is:");
        System.out.println(fs.selected);
    }
    else
        System.out.println("The original routing table has not been inputted.");
    break;

```

In case 3, if the original routing table has been inputted, the main function will use finalTable.class and call the method of RoutingTable to compute the routing table for all the routers, then output it via for loop.

```

case 3:
    if(Router.Table!= null&&Router.Table.size() != 0){
        ft.RoutingTable(Router.Table, lf.routerNumber);
        System.out.println("Final routing table computed by Distance Vector
algorithm is:");
        for(int i = 0; i < lf.routerNumber; i++){
            for(int j = 0; j < lf.routerNumber; j++){
                System.out.printf("%4s", ft.RoutingTable.get(lf.routerNumber*i+j));
                System.out.println();
            }
        }
    }
    else
        System.out.println("The original routing table has not been inputted.");
    break;

```

In case 4, if the original routing table has been inputted, the main function will use linkFailure.class and call the method of FailLink to make certain router to be unconnected to any other routers, and update the new routing table to the ArrayList for later use.


```

case 4:
    if(Router.Table!= null&&Router.Table.size() != 0){
        lfail.Faillink(Router.Table, lf.routerNumber);
        Router.update(Router, lfail.failedRouterTable);
    }
    else
        System.out.println("The original routing table has not been inputted.");
    break;

```

In case 5, if the original routing table has been inputted, the main function will use shortestPath.class and call the function of Path to compute the shortest path between two routers.

```

case 5:
    if(Router.Table!= null&&Router.Table.size() != 0)
        sp.Path(Router.Table, lf.routerNumber);
    else
        System.out.println("The original routing table has not been inputted.");
    break;

```

3.3 loadfile.class

This class is used to load an original file of routing table, and get necessary information to continue the following computation.

The program is going to read a txt file, and output it, thus I use the function FileReader in java to get the whole files. And I want to store the values in this txt file to an ArrayList, each time when I read a line, I strip the blank and add this line to list.

What's more, I gather a count for the number of routers, since I need this number to identify each number in the ArrayList, to validate input router number and so on .

```

//FileReader file = new FileReader("testfile.txt");
FileReader file = new FileReader(scan.nextLine());
in = new BufferedReader(file);
String line;
//the line is not empty
while((line=in.readLine())!=null){
    //every element in the txt file is separate by blank
    for(int i = 0; i < line.split(" ").length; i++){

```

```

        temp.add(Integer.parseInt(line.split(" ")[i]));
    }
    //count the number of routers
    routerNumber++;
}

```

3.4 finalTableForSelectedRouter.class

This is the first class that I use Bellman-ford algorithm to compute the routing table for a selected router.

To get the valid select number, I verify it with two steps, first of all, it should be a digit, then it should be greater than 1 and smaller than the number of routers.

```

while(selectNumber == Integer.MIN_VALUE){
    try{
        selectNumber = scan.nextInt();
    } catch (InputMismatchException ex){
        System.out.println("invalid value, input again:");
        scan.nextLine();
    }
}
//input a vaild value
while(selectNumber > size || selectNumber < 1){
    System.out.println("Invalid value, input again:");
    selectNumber = scan.nextInt();
}

```

The next step is to compute the routing table, I use a array list to store the original value of routing table, and define -1(indirectly connected) to be 10000, which is bigger enough to consider it as infinity. And to get the Distance Vector we just use the formula $D_x(y) = \min\{c(x, v) + D_v(y)\}$.

```

//compute the shortest distance
for(int n = 1; n < size; n++)
    for(int v = 0; v < size; v++)
        for(int i = 0; i < size; i++){
            for(int j = 0; j < size; j++){
                if(temp.get(v*size + i) > (a.get(v*size +
j)<0?infinity:a.get(v*size + j)) + temp.get(j*size + i)){
                    temp.set(v*size + i, a.get(v*size + j) +
temp.get(j*size + i));
                }
            }
        }
}

```

To get the selected router's routing table, we just need to get the selected line in the routing table and output it.

```
//define the selected router's routing table
for(int i = 0; i < size; i++)
    selected.add(temp.get((selectNumber - 1)*size + i));
```

3.5 finalTable.class

This class is used to compute the routing table for all the routers, and the function is almost the same as finalTableForSelectedRouter.class, and it don't need to input a router number to get the certain router's routing table.

It is also use BF algorithm to compute the Distance Vector.

```
//use BF algorithm to compute the routing table
for(int n = 1; n < size; n++)
    for(int v = 0; v < size; v++)
        for(int i = 0; i < size; i++)
            for(int j = 0; j < size; j++){
                if(RoutingTable.get(v*size + i) > (a.get(v*size +
j)<0?infinity:a.get(v*size + j)) + RoutingTable.get(j*size + i)){
                    RoutingTable.set(v*size+i, a.get(v*size + j) +
RoutingTable.get(j*size + i));
                }
            }
        }
```

3.6 LinkFailure.class

This class is used to realize the function of changing certain router's state to failure. To realize this function, we just need to update the value of original routing table. All the value's related to this router must be set to -1, which means this router is not reachable.

```
//set routing table's values which is related to failed router to -1
for(int i = 0; i < size; i++){
    a.set((failedRouter - 1)*size + i, -1);
    a.set(i*size + failedRouter - 1, -1);
}
```

Of course, I verify the input value, it should be a digit and is larger than 0, and smaller than the number of routers.

```
//input the failed router number
int failedRouter = Integer.MIN_VALUE;
```

```

while(failedRouter == Integer.MIN_VALUE){
    try{
        failedRouter = scan.nextInt();
    } catch (InputMismatchException ex){
        System.out.println("invalid value, input again:");
        scan.nextLine();
    }
}
//input a valid value
while(failedRouter > size || failedRouter < 1){
    System.out.println("Invalid value, input again:");
    failedRouter = scan.nextInt();
}

```

3.7 shortestPath.class

This class is used to find the shortest path from one router to another router. It is also using BF algorithm, however, it keeps an array to store the path and the concept of finding this path is introduced below in the 4) section.

Since we need to input three values, source, destination and bypass, we have to verify them.

```

//validate value of source, destination and bypass
while(bypass==-1?source>size||source<1||destination>size||destination<1:
source>size||source<1||destination>size||destination<1||bypass>size||bypass<1){
    System.out.println("Invalid value, input again:");
    br = new BufferedReader(new InputStreamReader(System.in));
    input = br.readLine();
    routerNumber = new String[3];
    //the source, destination and bypass is separate by blank
    routerNumber = input.split(" ");
    //initialize source, destination and bypass
    source = Integer.parseInt(routerNumber[0]);
    destination = Integer.parseInt(routerNumber[1]);
    bypass = -1;
    if(routerNumber.length>2)
        bypass = Integer.parseInt(routerNumber[2]);
}

```

And the next part of this class is initialization and use BF algorithm to update Distance Vector. After updating the Distance Vector, we add another statement, which is to define the next router in the path.

```

//use BF algorithm to compute the path
for(int n = 1; n < size; n++)
    for(int v = 0; v < size; v++)

```

```

        for(int i = 0; i < size; i++){
            for(int j = 0; j < size; j++){
                if(temp.get(v*size + i) > (a.get(v*size +
j)<0?infinity:a.get(v*size + j)) + temp.get(j*size + i)){
                    temp.set(v*size + i, a.get(v*size + j) +
temp.get(j*size + i));
                    next[v*size+i] = j+1;
                }
            }
        }
    }

```

The last part of this class is outputting the path, since there is two situation, we use a if-else statement to realize it differently.

One is that there is no bypass, we just need to output the path from source to destination. If the next router in the path has not reached the destination, we will store this router in an ArrayList iteratively.

```

int start = source;
int end = destination;
while(end != next[(start-1)*size+end-1]){
    path.add(start);
    if(start == next[(start-1)*size+end-1])
        break;
    start = next[(start-1)*size+end-1];
}
System.out.print("The shortest path from " + source + " to " + destination + " is ");
for(int i = 0; i <= path.size()-1; i++)
    System.out.print(path.get(i) + "-->");
System.out.print(destination + ", the total cost is " + temp.get((source-
1)*size+destination-1));
System.out.println();

```

Another is that there is a bypass, and we will compute the path in two parts, from source to bypass, and from bypass to end.

```

int start = source;
int middle = bypass;
int end = destination;
//from source to bypass
while(middle != next[(start-1)*size+middle-1]){
    Bpath.add(start);
    if(start == next[(start-1)*size+middle-1])
        break;
    start = next[(start-1)*size+middle-1];
}

//from bypass to destination
while(end != next[(middle-1)*size+end-1]){
    Bpath.add(middle);
    if(middle == next[(middle-1)*size+end-1])
        break;
    middle = next[(middle-1)*size+end-1];
}

```

```

}
System.out.print("The shortest path from " + source + " to " + destination + " (bypass "
"+bypass+)is ");
for(int i = 0; i <= Bpath.size()-1; i++)
    System.out.print(Bpath.get(i) + "-->");
System.out.print(destination + ", the total cost is " + (temp.get((source-
1)*size+bypass-1)+temp.get((bypass-1)*size+destination-1)));
System.out.println();

```

4) Details of finding a shortest path

To get a path, we need to know each router's next router, which means we have to reach this subsequent router before reaching the target router. I use a array to store each router's subsequent router. By searching the source router's subsequent router iteratively, we can get the path.

For example, we need to find a shortest path from router #1 to router #3, router #1's subsequent router is router #2, and router #2's subsequent router is router #3, since we have reached the destination router, the path is 1-2-3.

The pseudocode of outputting a path:

```

Initialize router = source
While router's subsequent router hasn't reach detination router
    output this subsequent router
    router = subsequent router
Endwhile

```

I still use Bellman-Ford algorithm to compute the routing table, however, after update a router's distance in the routing table, I will also update this router's subsequent router.

For the value of distance, we use this formula $D_x(y) = \min\{c(x, v) + D_v(y)\}$, and once $D_x(y)$ is update by $c(x, v) + D_v(y)$, the subsequent router of x is also changed to v.

The pseudocode of update an ancestor:

```

If there is a change of distance vector
For each router
    For each edge
         $D_x(y) = \min\{c(x, v) + D_v(y)\}$ 
        subsequent router of x equals v
    Endfor
Endfor

```

Particularly, since we are not sure whether there is a bypass, I define a `BufferedReader` to accept the input string, and if the length of string is larger than 2, there is a bypass, thus we can assign the bypass, else the assignment of bypass is -1.

Another problem is that we have to compute the shortest path in two different situations, one is that there is no bypass, and we only need to find a shortest path from source to destination, the other is that there exists a bypass and we have to compute the shortest path from source to bypass, and from bypass to destination.

5) Problems and reasons

1. I've set the infinity(-1) to 10000 in computing the shortest path and routing table, so if the cost of the path is larger than 10000, the program may come up with a wrong routing table or shortest path.
2. If the cost of a path is larger than three-digit number, the distance vectors in the routing table can't be aligned, since I set 4 blanks between two numbers.

6) Test User Cases

6.1 User case 1(Undirected graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0

Description of topology: This network is like a chain, each distance between two routers in 1. For any i and j , the shortest distance from R_i to R_j is $|j - i|$.

The final routing table produced by the program is:

Final routing table computed by Distance Vector algorithm is:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11
5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10
6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9
7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8
8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7
9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6
10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5
11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4
12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3
13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Obviously, the result is correct.

For some selected router, let it be 10:

```
Please input a router number:
10
The final table of this selected router is:
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6]
```

The routing table is just the same as 10th line in the final routing table.

As for the shortest path, let the source be 1 and destination be 16:

```
Please input the source and destination router number(bypass router number):
1 16
The shortest path from 1 to 16 is 1-->2-->3-->4-->5-->6-->7-->8-->9-->10-->11-->
12-->13-->14-->15-->16, the total cost is 15
```

The result is correct.

To test the link failure function, let the failed router be 4, and we are going to test it with the shortest path function, if router #4 is failed, there should be no path from 1 to 16.

```
Please input the number of failed router:
4
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 16
There is no path between 1 and 16.
```

It's the expected result.

6.2 User case 2(Undirected graph)

The original routing table is:

Original routing table is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Description of topology: Router #1 is the center of the network, each router can reach router #1 in one step, and the distance for router #i to router #1 is i-1. Thus, the distance between router #i and router #j is i+j-2.

The final routing table:

Final routing table computed by Distance Vector algorithm is:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	3	0	5	6	7	8	9	10	11	12	13	14	15	16	17
3	4	5	0	7	8	9	10	11	12	13	14	15	16	17	18
4	5	6	7	0	9	10	11	12	13	14	15	16	17	18	19
5	6	7	8	9	0	11	12	13	14	15	16	17	18	19	20
6	7	8	9	10	11	0	13	14	15	16	17	18	19	20	21
7	8	9	10	11	12	13	0	15	16	17	18	19	20	21	22
8	9	10	11	12	13	14	15	0	17	18	19	20	21	22	23
9	10	11	12	13	14	15	16	17	0	19	20	21	22	23	24
10	11	12	13	14	15	16	17	18	19	0	21	22	23	24	25
11	12	13	14	15	16	17	18	19	20	21	0	23	24	25	26
12	13	14	15	16	17	18	19	20	21	22	23	0	25	26	27
13	14	15	16	17	18	19	20	21	22	23	24	25	0	27	28
14	15	16	17	18	19	20	21	22	23	24	25	26	27	0	29
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	0

The distance vector is correct.

For some certain router, let it be router #16, the routing table is:

```
Please input a router number:
16
The final table of this selected router is:
[15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 0]
```

As for the shortest path, let the source be 2 and the destination be 16 and the bypass is 3:

```
Please input the source and destination router number(bypass router number):
2 16 3
The shortest path from 2 to 16 (bypass 3) is 2-->1-->3-->1-->16, the total cost is 20
```

The correct path should be 2-1-3-1-16, and the cost is $1+2+2+15=20$, the result produced by program is correct.

To test the link failure function, let router 1 be failed, then all the network will be disconnected:

```
4
Please input the number of failed router:
1
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 2
There is no path between 1 and 2.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 16
There is no path between 1 and 16.
```

The result is as expected.

6.3 User case 3(Undirected graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0

Description of topology: This network is like a cycle, and each edge's distance is 1.

The final table for this network is:

Final routing table computed by Distance Vector algorithm is:

0	1	2	3	4	5	6	7	8	7	6	5	4	3	2	1
1	0	1	2	3	4	5	6	7	8	7	6	5	4	3	2
2	1	0	1	2	3	4	5	6	7	8	7	6	5	4	3
3	2	1	0	1	2	3	4	5	6	7	8	7	6	5	4
4	3	2	1	0	1	2	3	4	5	6	7	8	7	6	5
5	4	3	2	1	0	1	2	3	4	5	6	7	8	7	6
6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	7
7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8
8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7
7	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6
6	7	8	7	6	5	4	3	2	1	0	1	2	3	4	5
5	6	7	8	7	6	5	4	3	2	1	0	1	2	3	4
4	5	6	7	8	7	6	5	4	3	2	1	0	1	2	3
3	4	5	6	7	8	7	6	5	4	3	2	1	0	1	2
2	3	4	5	6	7	8	7	6	5	4	3	2	1	0	1
1	2	3	4	5	6	7	8	7	6	5	4	3	2	1	0

And since it is a cycle, the farthest router from router #1 is router #9, and the distance is 9, which is just the same as the distance vector computed by program.

As for the selected router, let it be 9:

```

Please input a router number:
9
The final table of this selected router is:
[8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7]

```

To verify the shortest distance function, we test it in two cases, one is from 1 to 4, and the other is from 1 to 14, the first should be 1-2-3-4, and the other should be 1-16-15-14, each distance is 3.

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 4
The shortest path from 1 to 4 is 1-->2-->3-->4, the total cost is 3
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 14
The shortest path from 1 to 14 is 1-->16-->15-->14, the total cost is 3

```

If we let the router #3 fail, if router #1 want to go to router #4, it will go though router #16, router #15,... router #5, to router #4.

```

Please input the number of failed router:
3
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 4
The shortest path from 1 to 4 is 1-->16-->15-->14-->13-->12-->11-->10-->9-->8-->
7-->6-->5-->4, the total cost is 13

```

That's the correct answer.

6.4 User case 4(Undirected graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	0	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1
-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0

Description of topology: the network is divided into two parts, 1, 2, 3, 4, 5, 6, 7, 8 and 9, 10, 11, 12, 13, 14, 15, 16, and router #4 and router #12 is connected. Let the source be i , destination be j , if i and j are in the same part, the distance is $|j-i|$, and to reach the other part, the distance is $|4-i| + |12-j| + 1$.

The final router table is correct.

Final routing table computed by Distance Vector algorithm is:

0	1	2	3	4	5	6	7	7	6	5	4	5	6	7	8
1	0	1	2	3	4	5	6	6	5	4	3	4	5	6	7
2	1	0	1	2	3	4	5	5	4	3	2	3	4	5	6
3	2	1	0	1	2	3	4	4	3	2	1	2	3	4	5
4	3	2	1	0	1	2	3	5	4	3	2	3	4	5	6
5	4	3	2	1	0	1	2	6	5	4	3	4	5	6	7
6	5	4	3	2	1	0	1	7	6	5	4	5	6	7	8
7	6	5	4	3	2	1	0	8	7	6	5	6	7	8	9
7	6	5	4	5	6	7	8	0	1	2	3	4	5	6	7
6	5	4	3	4	5	6	7	1	0	1	2	3	4	5	6
5	4	3	2	3	4	5	6	2	1	0	1	2	3	4	5
4	3	2	1	2	3	4	5	3	2	1	0	1	2	3	4
5	4	3	2	3	4	5	6	4	3	2	1	0	1	2	3
6	5	4	3	4	5	6	7	5	4	3	2	1	0	1	2
7	6	5	4	5	6	7	8	6	5	4	3	2	1	0	1
8	7	6	5	6	7	8	9	7	6	5	4	3	2	1	0

As for some certain router, let it be 4, the routing table for it is:

```
Please input a router number:
4
The final table of this selected router is:
[3, 2, 1, 0, 1, 2, 3, 4, 4, 3, 2, 1, 2, 3, 4, 5]
```

To test the shortest path function, let the source be 2 and destination be 15, and there is a bypass 11. The path should be 2-3-4-12-11-12-13-14-15.

```
Please input the source and destination router number(bypass router number):
2 15 11
The shortest path from 2 to 15 (bypass 11)is 2-->3-->4-->12-->11-->12-->13-->14-->15, the total cost is 8
```

It is correct.

Let the router #12 failed, the first part should be able to reach any router in part one, but they cannot reach any router in part two. Let the source be 1 and destination be 7, the

```
Please input the number of failed router:
12
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 7
The shortest path from 1 to 7 is 1-->2-->3-->4-->5-->6-->7, the total cost is 6
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 11
There is no path between 1 and 11.
```


6.5 User case 5(Undirected graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	0	1	-1	1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	1	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	0	1	-1	1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1	0	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	0	1	-1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1	0

Description of topology: the router is divided into 4 parts, 1, 2, 3, 4 is a cycle and 5, 6, 7, 8 is a cycle, and 9, 10, 11, 12 is a cycle, and 13, 14, 15, 16 is a cycle. Edge(1, 5), (5, 9), (9, 13), (13, 1) link these four parts together, and each edge's distance is 1.

The final routing table:

Final routing table computed by Distance Vector algorithm is:

0	1	2	1	1	2	3	2	2	3	4	3	1	2	3	2
1	0	1	2	2	3	4	3	3	4	5	4	2	3	4	3
2	1	0	1	3	4	5	4	4	5	6	5	3	4	5	4
1	2	1	0	2	3	4	3	3	4	5	4	2	3	4	3
1	2	3	2	0	1	2	1	1	2	3	2	2	3	4	3
2	3	4	3	1	0	1	2	2	3	4	3	3	4	5	4
3	4	5	4	2	1	0	1	3	4	5	4	4	5	6	5
2	3	4	3	1	2	1	0	2	3	4	3	3	4	5	4
2	3	4	3	1	2	3	2	0	1	2	1	1	2	3	2
3	4	5	4	2	3	4	3	1	0	1	2	2	3	4	3
4	5	6	5	3	4	5	4	2	1	0	1	3	4	5	4
3	4	5	4	2	3	4	3	1	2	1	0	2	3	4	3
1	2	3	2	2	3	4	3	1	2	3	2	0	1	2	1
2	3	4	3	3	4	5	4	2	3	4	3	1	0	1	2
3	4	5	4	4	5	6	5	3	4	5	4	2	1	0	1
2	3	4	3	3	4	5	4	2	3	4	3	1	2	1	0

Let the selected router be 10:

```
Please input a router number:
10
The final table of this selected router is:
[3, 4, 5, 4, 2, 3, 4, 3, 1, 0, 1, 2, 2, 3, 4, 3]
```

And if we let the router 13 failed, we'll test it by outputting the final routing table:

```
Please input the number of failed router:
13
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
3
Final routing table computed by Distance Vector algorithm is:
  0  1  2  1  1  2  3  2  2  3  4  3 -1 -1 -1 -1
  1  0  1  2  2  3  4  3  3  4  5  4 -1 -1 -1 -1
  2  1  0  1  3  4  5  4  4  5  6  5 -1 -1 -1 -1
  1  2  1  0  2  3  4  3  3  4  5  4 -1 -1 -1 -1
  1  2  3  2  0  1  2  1  1  2  3  2 -1 -1 -1 -1
  2  3  4  3  1  0  1  2  2  3  4  3 -1 -1 -1 -1
  3  4  5  4  2  1  0  1  3  4  5  4 -1 -1 -1 -1
  2  3  4  3  1  2  1  0  2  3  4  3 -1 -1 -1 -1
  2  3  4  3  1  2  3  2  0  1  2  1 -1 -1 -1 -1
  3  4  5  4  2  3  4  3  1  0  1  2 -1 -1 -1 -1
  4  5  6  5  3  4  5  4  2  1  0  1 -1 -1 -1 -1
  3  4  5  4  2  3  4  3  1  2  1  0 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  1  2
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  0  1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  2  1  0
```

And then if we want to reach 16 from 1, it is impossible:

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 16
There is no path between 1 and 16.
```

6.6 User case 6(Undirected graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	0	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	-1	2	2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	0	-1	-1	2	2	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	2	-1	0	-1	-1	-1	3	3	-1	-1	-1	-1	-1	-1
-1	-1	2	-1	-1	0	-1	-1	-1	-1	3	3	-1	-1	-1	-1
-1	-1	-1	2	-1	-1	0	-1	-1	-1	-1	-1	3	3	-1	-1
-1	-1	-1	2	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	3	3
-1	-1	-1	-1	3	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	3	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	3	-1	-1	-1	-1	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	3	-1	-1	-1	-1	-1	0	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	3	-1	-1	-1	-1	-1	0	-1	-1	-1
-1	-1	-1	-1	-1	-1	3	-1	-1	-1	-1	-1	-1	0	-1	-1
-1	-1	-1	-1	-1	-1	-1	3	-1	-1	-1	-1	-1	-1	0	-1
-1	-1	-1	-1	-1	-1	-1	3	-1	-1	-1	-1	-1	-1	-1	0

Description of topology: Except router #1, the other routers form a binary tree, router 2 is the root and router i is father of router $2*i-1$ and router $2*i$, router 9, 10, 11, 12, 13, 14, 15, 16 are leaves. The edges in layer 1 cost 1, edges in layer 2 cost 2, and edges in layer 3 cost 3. Router 1 is connected just with router 2.

The final routing table:

Final routing table computed by Distance Vector algorithm is:

0	1	2	2	4	4	4	4	7	7	7	7	7	7	7	7
1	0	1	1	3	3	3	3	6	6	6	6	6	6	6	6
2	1	0	2	2	2	4	4	5	5	5	5	7	7	7	7
2	1	2	0	4	4	2	2	7	7	7	7	5	5	5	5
4	3	2	4	0	4	6	6	3	3	7	7	9	9	9	9
4	3	2	4	4	0	6	6	7	7	3	3	9	9	9	9
4	3	4	2	6	6	0	4	9	9	9	9	3	3	7	7
4	3	4	2	6	6	4	0	9	9	9	9	7	7	3	3
7	6	5	7	3	7	9	9	0	6	10	10	12	12	12	12
7	6	5	7	3	7	9	9	6	0	10	10	12	12	12	12
7	6	5	7	7	3	9	9	10	10	0	6	12	12	12	12
7	6	5	7	7	3	9	9	10	10	6	0	12	12	12	12
7	6	7	5	9	9	3	7	12	12	12	12	0	6	10	10
7	6	7	5	9	9	3	7	12	12	12	12	6	0	10	10
7	6	7	5	9	9	7	3	12	12	12	12	10	10	0	6
7	6	7	5	9	9	7	3	12	12	12	12	10	10	6	0

For some selected router, let it be router 9:

```
Please input a router number:
9
The final table of this selected router is:
[7, 6, 5, 7, 3, 7, 9, 9, 0, 6, 10, 10, 12, 12, 12, 12]
```

If we want to transfer from router 9 to router 16, it will trace back to the root 2, and go down to the leaf 16:

```
Please input the source and destination router number(bypass router number):
9 16
The shortest path from 9 to 16 is 9-->5-->3-->2-->4-->8-->16, the total cost is
12
```

Let the router 2 failed, we'll test it by outputting the final routing table, since router 2 is failed, the left part of the tree cannot reach the right part of the tree, and router 1 is also disconnected.

```
Please input the number of failed router:
2
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
3
Final routing table computed by Distance Vector algorithm is:
  0  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
-1  -1  0  -1  2  2  -1  -1  5  5  5  5  -1  -1  -1  -1
-1  -1  -1  0  -1  -1  2  2  -1  -1  -1  -1  5  5  5  5
-1  -1  2  -1  0  4  -1  -1  3  3  7  7  -1  -1  -1  -1
-1  -1  2  -1  4  0  -1  -1  7  7  3  3  -1  -1  -1  -1
-1  -1  -1  2  -1  -1  0  4  -1  -1  -1  -1  3  3  7  7
-1  -1  -1  2  -1  -1  4  0  -1  -1  -1  -1  7  7  3  3
-1  -1  5  -1  3  7  -1  -1  0  6  10  10  -1  -1  -1  -1
-1  -1  5  -1  3  7  -1  -1  6  0  10  10  -1  -1  -1  -1
-1  -1  5  -1  7  3  -1  -1  10  10  0  6  -1  -1  -1  -1
-1  -1  5  -1  7  3  -1  -1  10  10  6  0  -1  -1  -1  -1
-1  -1  -1  5  -1  -1  3  7  -1  -1  -1  -1  0  6  10  10
-1  -1  -1  5  -1  -1  3  7  -1  -1  -1  -1  6  0  10  10
-1  -1  -1  5  -1  -1  7  3  -1  -1  -1  -1  10  10  0  6
-1  -1  -1  5  -1  -1  7  3  -1  -1  -1  -1  10  10  6  0
```

6.7 User case 7(Directed graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	0	2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	0	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	0	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	0	5	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	0	6	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	0	7	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	0	8	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	0	9	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	0	10	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	11	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	12	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	13	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	14	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	15
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Description of topology: The network is like a cycle, but it is directed, 1->2->3->4->5->6->7->8->9->10->11->12->13->14->15->16->1. And the distance between router #i and router #i+1 is i, the distance between router #16 and router #1 is 16. To reach another router, one can only follow the direction and cannot reverse spread.

The final routing table:

Final routing table computed by Distance Vector algorithm is:

0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120
135	0	2	5	9	14	20	27	35	44	54	65	77	90	104	119
133	134	0	3	7	12	18	25	33	42	52	63	75	88	102	117
130	131	133	0	4	9	15	22	30	39	49	60	72	85	99	114
126	127	129	132	0	5	11	18	26	35	45	56	68	81	95	110
121	122	124	127	131	0	6	13	21	30	40	51	63	76	90	105
115	116	118	121	125	130	0	7	15	24	34	45	57	70	84	99
108	109	111	114	118	123	129	0	8	17	27	38	50	63	77	92
100	101	103	106	110	115	121	128	0	9	19	30	42	55	69	84
91	92	94	97	101	106	112	119	127	0	10	21	33	46	60	75
81	82	84	87	91	96	102	109	117	126	0	11	23	36	50	65
70	71	73	76	80	85	91	98	106	115	125	0	12	25	39	54
58	59	61	64	68	73	79	86	94	103	113	124	0	13	27	42
45	46	48	51	55	60	66	73	81	90	100	111	123	0	14	29
31	32	34	37	41	46	52	59	67	76	86	97	109	122	0	15
16	17	19	22	26	31	37	44	52	61	71	82	94	107	121	0

For some selected router, let it be router #2:

```
Please input a router number:
2
The final table of this selected router is:
[135, 0, 2, 5, 9, 14, 20, 27, 35, 44, 54, 65, 77, 90, 104, 119]
```

To test a path, let the source be 1 and destination be 4, but there is a bypass 13, thus we need to go through the whole cycle to reach router #4, the distance should be $1+2+\dots+16+1+2+3=142$.

```
Please input the source and destination router number(bypass router number):
1 4 13
The shortest path from 1 to 4 (bypass 13) is 1-->2-->3-->4-->5-->6-->7-->8-->9-->10-->11-->12-->13-->14-->15-->16-->1-->2-->3-->4, the total cost is 142
```

And once a router is failed, the network is broken into two parts, and the routers still need to transfer in the right direction. Let the failed router be 8, and we will retest it by finding a shortest path from 1 to 6, 9 to 13 and 1 to 13.

```
Please input the number of failed router:
8
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 6
The shortest path from 1 to 6 is 1-->2-->3-->4-->5-->6, the total cost is 15
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
9 13
The shortest path from 9 to 13 is 9-->10-->11-->12-->13, the total cost is 42
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 13
There is no path between 1 and 13.
```

6.8 User case 8(Directed graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2
2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0

Description of topology: It is still a cycle, however, the cycle has two directions, the forward direction is ascending order, the reverse direction is descending order. Each edge in forward direction costs 1, and Each edge in reverse direction costs 2. For router #1, if it want to reach router #12, 13, 14, 15, 16, it can transfer in the reverse direction, and if it want to reach router #2, 3, ..., 11, it should transfer in the forward direction. The largest cost in final routing table should be 10.

Final routing table computed by Distance Vector algorithm is:

0	1	2	3	4	5	6	7	8	9	10	10	8	6	4	2
2	0	1	2	3	4	5	6	7	8	9	10	10	8	6	4
4	2	0	1	2	3	4	5	6	7	8	9	10	10	8	6
6	4	2	0	1	2	3	4	5	6	7	8	9	10	10	8
8	6	4	2	0	1	2	3	4	5	6	7	8	9	10	10
10	8	6	4	2	0	1	2	3	4	5	6	7	8	9	10
10	10	8	6	4	2	0	1	2	3	4	5	6	7	8	9
9	10	10	8	6	4	2	0	1	2	3	4	5	6	7	8
8	9	10	10	8	6	4	2	0	1	2	3	4	5	6	7
7	8	9	10	10	8	6	4	2	0	1	2	3	4	5	6
6	7	8	9	10	10	8	6	4	2	0	1	2	3	4	5
5	6	7	8	9	10	10	8	6	4	2	0	1	2	3	4
4	5	6	7	8	9	10	10	8	6	4	2	0	1	2	3
3	4	5	6	7	8	9	10	10	8	6	4	2	0	1	2
2	3	4	5	6	7	8	9	10	10	8	6	4	2	0	1
1	2	3	4	5	6	7	8	9	10	10	8	6	4	2	0

To test the function of finding the shortest path, let the source router be 1 and the destination router be 11, and there is a bypass 16, if there is no bypass, the path should be 1-2-3-...-11, however, since there is a bypass, we need to reach router 16 first, and 1-16-1-2-...-11 is longer than 1-16-15-...-11, the path should be the latter.

```
Please input the source and destination router number(bypass router number):
1 11 16
The shortest path from 1 to 11 (bypass 16) is 1-->16-->15-->14-->13-->12-->11, the total cost is 12
```

Let the router #8 fail, we will test it by finding the shortest path from 1 to 6, 9 to 13, 1 to 13, since the network is dual direction, unlike user case 7, we can still reach 13 from 1 .

```
Please input the number of failed router:
8
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 6
The shortest path from 1 to 6 is 1-->2-->3-->4-->5-->6, the total cost is 5
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
9 13
The shortest path from 9 to 13 is 9-->10-->11-->12-->13, the total cost is 4
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
1 13
The shortest path from 1 to 13 is 1-->16-->15-->14-->13, the total cost is 8
```


6.9 User case 9(Directed graph)

The original routing table is:

Original routing table is as follows:

0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0	1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	0

Description of topology: The network is also a cycle, unlike the cycle introduced in user case 8, router #1 cannot reach #16 in reverse direction, only router #16 can reach router #1 in forward direction.

Final routing table computed by Distance Vector algorithm is:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13
6	4	2	0	1	2	3	4	5	6	7	8	9	10	11	12
8	6	4	2	0	1	2	3	4	5	6	7	8	9	10	11
10	8	6	4	2	0	1	2	3	4	5	6	7	8	9	10
10	10	8	6	4	2	0	1	2	3	4	5	6	7	8	9
9	10	10	8	6	4	2	0	1	2	3	4	5	6	7	8
8	9	10	10	8	6	4	2	0	1	2	3	4	5	6	7
7	8	9	10	10	8	6	4	2	0	1	2	3	4	5	6
6	7	8	9	10	10	8	6	4	2	0	1	2	3	4	5
5	6	7	8	9	10	10	8	6	4	2	0	1	2	3	4
4	5	6	7	8	9	10	10	8	6	4	2	0	1	2	3
3	4	5	6	7	8	9	10	10	8	6	4	2	0	1	2
2	3	4	5	6	7	8	9	10	10	8	6	4	2	0	1
1	2	3	4	5	6	7	8	9	10	10	8	6	4	2	0

For some selected router, let it be router #7, it should reach router #1 in forward direction, and reach router #2 in reverse direction.

```

Please input a router number:
7
The final table of this selected router is:
[10, 10, 8, 6, 4, 2, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```

To test the function of finding a shortest path, let the source be 7 and destination be 1 and 2, as introduced above, router #7 should reach router #1 in forward direction, and reach router #2 in reverse direction.

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
7 1
The shortest path from 7 to 1 is 7-->8-->9-->10-->11-->12-->13-->14-->15-->16-->
1, the total cost is 10
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
7 2
The shortest path from 7 to 2 is 7-->6-->5-->4-->3-->2, the total cost is 10

```

And if router #8 fails this time, router #7 cannot reach router #13, router #13 can reach router #7.

```

Please input the number of failed router:
8
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
7 13
There is no path between 7 and 13.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
13 7
The shortest path from 13 to 7 is 13-->14-->15-->16-->1-->2-->3-->4-->5-->6-->7,
the total cost is 10

```

6.10 User case 10(Error processing)

If the txt file name is not correct or there is no such file, the program should inform that the input is invalid.

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
1
Please load original routing table data file:
test.txt
There is no such file.
The original routing table has not been inputted.
```

And if the original table is not inputted, function 2, 3, 4, 5 is also not available.

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
2
The original routing table has not been inputted.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
3
The original routing table has not been inputted.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
4
The original routing table has not been inputted.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
The original routing table has not been inputted.
```

Let's input the valid original routing table, it is:

```
Original routing table is as follows:
0  8  3  4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
8  0 -1 -1 1 11 -1 -1 -1 -1 -1 -1 -1 -1 -1
3 -1 0  2 -1 7  7 -1 -1 -1 -1 -1 -1 -1 -1
4 -1 2  0 -1 -1 6 15 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 0 -1 -1 -1 5 -1 2 -1 -1 -1 -1
-1 11 7 -1 -1 0 -1 -1 4 -1 -1 6 -1 -1 -1
-1 -1 7  6 -1 -1 0  2 -1 2 -1 -1 -1 -1 -1
-1 -1 -1 15 -1 -1 2  0 -1 -1 -1 1 -1 -1 -1
-1 -1 -1 -1 5  4 -1 -1 0  5 -1 -1 2 -1 -1
-1 -1 -1 -1 -1 2 -1 -1 -1 5  0 -1 -1 7 15 -1
-1 -1 -1 -1 2 -1 -1 -1 -1 -1 0 -1 -1 1 -1
-1 -1 -1 -1 -1 6 -1 1 -1 -1 -1 0 -1 -1 3 -1
-1 -1 -1 -1 -1 -1 -1 -1 2 -1 -1 -1 0 1 -1 1
-1 -1 -1 -1 -1 -1 -1 -1 -1 7 -1 -1 1 0 -1 2
-1 -1 -1 -1 -1 -1 -1 -1 -1 15 1 3 -1 -1 0 1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 2 1 0
```

For function 2, if the input router number is not correct, the program should inform that the value is invalid and ask user to input again:

```
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
2
Please input a router number:
0
Invalid value, input again:
```

Let the selected router be 1, we will output the final routing table for router #1:

```
Please input a router number:
1
The final table of this selected router is:
[0, 8, 3, 4, 9, 10, 10, 12, 14, 12, 11, 13, 14, 15, 12, 13]
```

And select the function 3, the program should output the whole final routing table:

Final routing table computed by Distance Vector algorithm is:

0	8	3	4	9	10	10	12	14	12	11	13	14	15	12	13
8	0	11	12	1	10	10	8	6	11	3	7	6	7	4	5
3	11	0	2	12	7	7	9	11	9	14	10	13	14	13	14
4	12	2	0	13	9	6	8	13	8	13	9	14	15	12	13
9	1	12	13	0	9	9	7	5	10	2	6	5	6	3	4
10	10	7	9	9	0	9	7	4	9	9	6	6	7	8	7
10	10	7	6	9	9	0	2	7	2	7	3	8	9	6	7
12	8	9	8	7	7	2	0	8	4	5	1	6	7	4	5
14	6	11	13	5	4	7	8	0	5	5	7	2	3	4	3
12	11	9	8	10	9	2	4	5	0	9	5	7	7	8	8
11	3	14	13	2	9	7	5	5	9	0	4	3	4	1	2
13	7	10	9	6	6	3	1	7	5	4	0	5	6	3	4
14	6	13	14	5	6	8	6	2	7	3	5	0	1	2	1
15	7	14	15	6	7	9	7	3	7	4	6	1	0	3	2
12	4	13	12	3	8	6	4	4	8	1	3	2	3	0	1
13	5	14	13	4	7	7	5	3	8	2	4	1	2	1	0

For function 4, the program should also validate the input router number, it should be bigger than 0 and smaller than the number of routers:

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
4
Please input the number of failed router:
0
Invalid value, input again:
17
Invalid value, input again:

```

Let's input a valid number, and let router #1 failed, the final routing table should be updated, since router #1 is failed.

```

Please input the number of failed router:
1
Router failed.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
3
Final routing table computed by Distance Vector algorithm is:
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 0 17 16 1 10 10 8 6 11 3 7 6 7 4 5
-1 17 0 2 16 7 7 9 11 9 14 10 13 14 13 14
-1 16 2 0 15 9 6 8 13 8 13 9 14 15 12 13
-1 1 16 15 0 9 9 7 5 10 2 6 5 6 3 4
-1 10 7 9 9 0 9 7 4 9 9 6 6 7 8 7
-1 10 7 6 9 9 0 2 7 2 7 3 8 9 6 7
-1 8 9 8 7 7 2 0 8 4 5 1 6 7 4 5
-1 6 11 13 5 4 7 8 0 5 5 7 2 3 4 3
-1 11 9 8 10 9 2 4 5 0 9 5 7 7 8 8
-1 3 14 13 2 9 7 5 5 9 0 4 3 4 1 2
-1 7 10 9 6 6 3 1 7 5 4 0 5 6 3 4
-1 6 13 14 5 6 8 6 2 7 3 5 0 1 2 1
-1 7 14 15 6 7 9 7 3 7 4 6 1 0 3 2
-1 4 13 12 3 8 6 4 4 8 1 3 2 3 0 1
-1 5 14 13 4 7 7 5 3 8 2 4 1 2 1 0

```

For the function 5, we should also validate the value of input numbers, the source, destination and bypass should be bigger than 0 and smaller than 16.

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
0 16
Invalid value, input again:
1 17
Invalid value, input again:
1 16 0
Invalid value, input again:

```

Let's input the valid value, and since router #1 is failed, there shouldn't be any path involved with router #1.

```

Please input the source and destination router number(bypass router number):
1 16
There is no path between 1 and 16.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
16 1
There is no path between 16 and 1.
1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
2 16 1
There is no path between 2 and 16.

```

We input a set of valid value to test the function 5:

```

1-LOAD FILE
2-OUTPUT FINAL TABLE FOR A SELECTED ROUTER
3-COMPUTE AND OUTPUT FINAL ROUTING TABLE
4-LINK FAILURE
5-OUTPUT OPTIMAL PATH AND MINIMUM COST
5
Please input the source and destination router number(bypass router number):
2 16
The shortest path from 2 to 16 is 2-->5-->11-->15-->16, the total cost is 5

```

Compare it with final routing table, and observe the topology graph, the path is correct.