CS 542 Project Report
Link State Routing Simulator

Name: Vishwanath Bidari

CWID: A20352720

Section 04

Contents

1. INTRODUCTION

A. PURPOSE

This project implements Link state routing simulator.

B. SCOPE

The main scope of this this project are as below.

1. Create network topology in the form of matrix, which only indicates the costs of links between all directly connected routers.
2. Display connection table for any router any network topology as a source to all other routers in network topology.
3. Display the shortest path and shortest distance between source and destination.
4. Remove router from above network topology and update connection table. Recalculate shortest path and shortest distance between source and destination.
5. Display best router in network topology which connects to all other routers in topology with minimum cost.

C. OVERVIEW

Routing is the process of determining the path from source to destination in network. Routing process is performed in many kind of network applications such as telephone circuit switching network, electronic data network, transportation networks. Routing process implements effective path between source and destination with minimum cost. The nodes in network topology typically includes routers, bridges, switches, firewalls. Routing table is maintained at every node in network using routing process. Forwarding is the process of interchange between two nodes in network. Forwarding is performed by looking connection table for each destination. Connection table provides interface through which forwarding is done to reach destination with minimum cost in network.

D. ACRONYMS

LSP	Link State
JVM	Java virtual machine
LSA	Link state algorithm
CN	Computer Networks

2. Link State Routing Protocol.

There are two routing protocols used in packet switching networks for computer communications. Routing protocols are as below.

1. Link State Routing protocol
2. Distance-vector routing protocol.

Examples of link state routing protocol are Open short path first (OSPF) and Intermediate system to intermediate system (IS-IS).

- Each router is responsible for finding its neighbor router and detecting their name.
- Each router implements link state algorithm to construct a connection table, which consists all its neighbors and cost of each neighbor.
- Each router transmits LSP to all other routers in table. Each router stores most recent LSP form each router.
- Link state flooding is sequencing and aging procedures.
- Each router stores the identical link state database.
- Each router uses complete LSA information in network to compute shortest path between source and destination.
- Router uses Dijkstra's algorithm to calculate the shortest path.

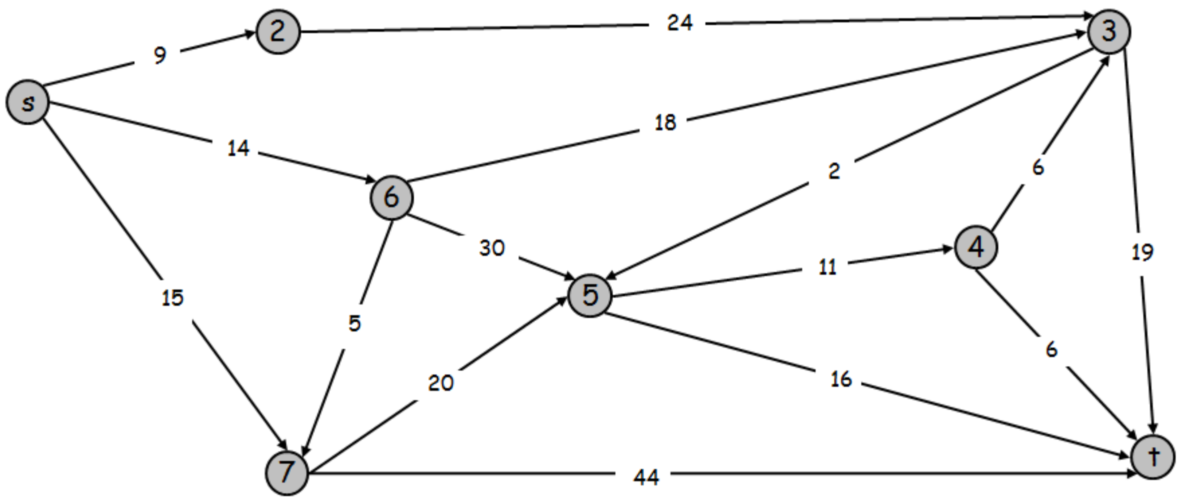
3. Dijkstra’s Shortest Path Algorithm.

Dutch computer scientist ‘Edsger Dijkstra’ derived Dijkstra algorithm in 1956 and published in 1959. This algorithm is mainly used in graph network topology problems to find shortest path from source to destination. Graph should have non-negative edges. This algorithm is used in implementing many routing protocols.

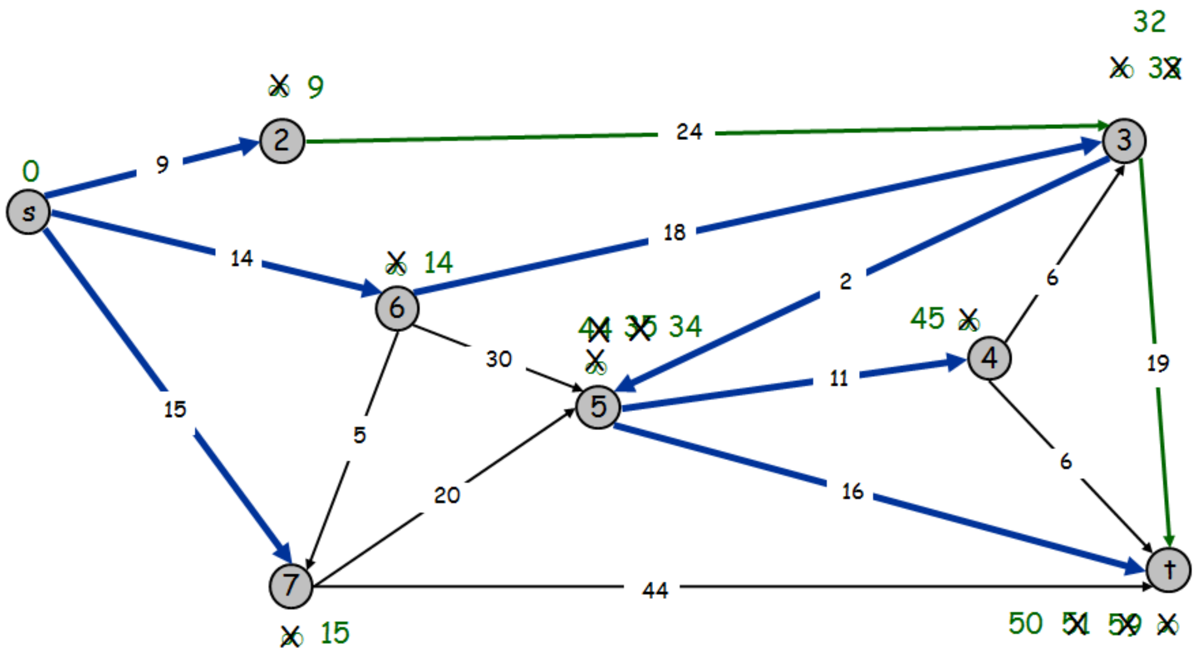
This algorithm is used in finding minimum cost of shortest paths from single source vertex to single destination vertex in network. For example, if the vertices of the graph represent cities and cost of edge between vertices represent distance between cities connected by road. Dijkstra algorithm is used to find shortest path between source city to all other cities in network. So this algorithm is mostly used network routing protocols. Some of the most common protocols uses Dijkstra algorithm are OSPF (Open short path first) and IS-IS.

Dijkstra algorithm which does not use min-priority queue runs in $O(|V|^2)$, where V is number of vertices in graph. Dijkstra algorithm can also implemented by min-priority queue using Fibonacci heap. This algorithm have time complexity of $O(|E|+|V|\log|V|)$ where $|E|$ is the number of edges. This is asymptotically fastest known single-source shortest path algorithm for arbitrary directed graph with non-negative edges.

Example: Find shortest path from source s to destination t in below graph using Dijkstra algorithm,



Final graph with shortest path from source s to t is as below.



Pseudo code for Dijkstra algorithm:

```
Initialization:
N' = {u}
for all nodes v
  if v adjacent to u
    then $D(v) = c(u,v)$
  else $D(v) = \infty$

Loop
  find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      $D(v) = \min( D(v), D(w) + c(w,v) )$
    /* new cost to v is either old cost to v or known
       shortest path cost to w plus cost from w to v */
until all nodes in N'
```

4. Application Design Description

The objective of this application is as below.

1. Create network topology in the form of matrix, which only indicates the costs of links between all directly connected routers.
2. Display connection table for any router any network topology as a source to all other routers in network topology.
3. Display the shortest path and shortest distance between source and destination.
4. Remove router from above network topology and update connection table. Recalculate shortest path and shortest distance between source and destination.
5. Display best router in network topology which connects to all other routers in topology with minimum cost.

This application is implemented in Java.
Application provides the following options to user.
1. Create a Network Topology
2. Build a Connection Table
3. Shortest Path to Destination Router
4. Modify a topology
5. Exit

Implementation of application functions as below.

create_topology() method is used to create topology matrix from input file, which only indicates the costs of links between all directly connected routers.

build_connection_tab() method is used build connection table for source router. Connection table contain interface value for each router in network topology from source router. Interface is router through which source can connect to destination router.

Dikjstra() method is used to implement Dijkstra algorithm. Dijkstra algorithm calculates shortest distance to all routers in topology from source router. This method also calculates predecessor for each router in topology from source router.

shortest_path_destination() method is used to fine shortest distance and shortest path from source to destination router in topology.

remove_router() is used remove router from topology matrix and calls build_connection_tab and shortest_path_destination methods to recalculate connection table and shortest path from source to destination router respectively.

best_router() method is used to find router which is connected to all or maximum number of routers in topology with minimum cost.

Implementation of Dijkstra algorithm

- Source router number is passed to Dijkstra method to calculate shortest distance from source router all other routers in network topology matrix. It also calculates predecessor for each router in topology from source router.

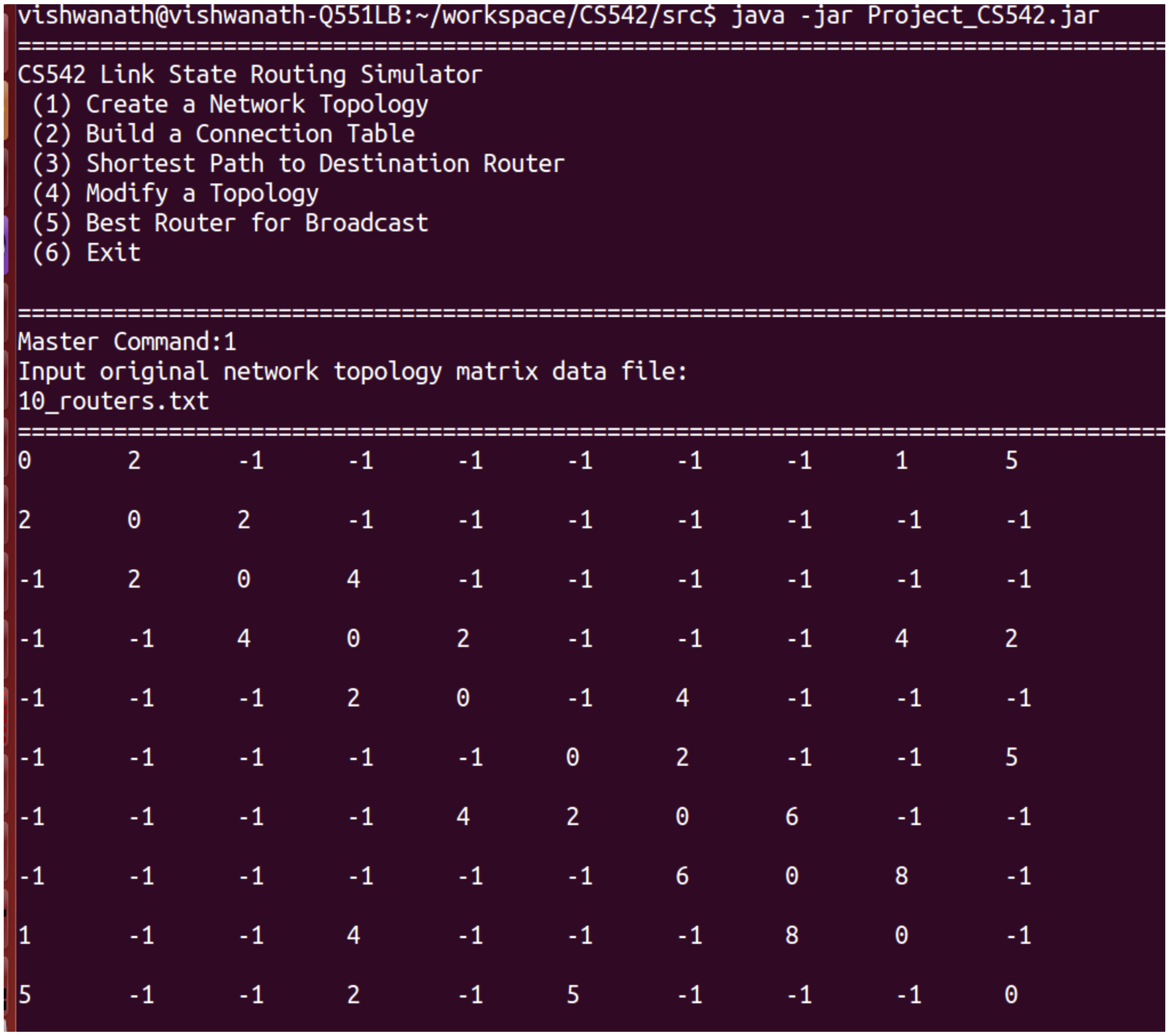
1. Initialize arrays such as shortDist to Integer max value, preDsr to -1 and visited to false. It means shortest distance to all routers from source is infinite.
2. Visited array false means any router is not visited in algorithm.
3. Initialize distance of source from itself to zero.
4. Find router which is not visited and minimum distance form source and set visited of that router value to true in visited array.
5. Find adjacent routers to selected minimum distance router and calculate minimum distance to adjacent routers from selected minimum distance router from source.
6. Calculate predecessor of each router while calculating shortest distance from router.
7. Repeat from step 4 till all routers are visited from source.

Source code for Dijkstra algorithm is as below.

```
/**
   * This dikjstra method is used to implement Dijkstra algorithm.
   * Dijkstra algorithm calculates shortest distance to all routers in topology from source router.
   * This method also calculates predecessor for each router in topology from source router.
   *
   * @param int src: Source router number.
   * @return Nothing.
   * @exception IOException On input error.
   * @see IOException
   */
public static void dikjstra(int src)
{
 //Number of vertices in graph
 int V=graph.length;

 //Index of shorted distance router in topology matrix from adjacent router.
 //Initialize with -1
 int min_index=-1;

 //Value of shortest distance router in topology matrix from adjacent router
 //Initialize with max value
 int min_value=Integer.MAX_VALUE;

 //Array to store shortest distance to all routers in topology from source router
 shortDist=new int[V];

 //Array to store predecessor to all routers in topology from source router.
 preDsr=new int[V];

 //Array to store visited routes in topology from source router
 Boolean visited[]=new Boolean[V];

 //Initialize arrays to default values
 for(int i=0;i<V;i++)
 {
       //Initialize with max distance to all routers from source router
       shortDist[i]=Integer.MAX_VALUE;

       //Initialize all routers with not visited state.
           visited[i]=false;

           // Initialize all router predecessor as not reachable from source router
           preDsr[i]=-1;
     }
     //Initialize distance to source router from itself as zero.
     shortDist[src]=0;

     //Initialize predecessor of source router as itself
     preDsr[src]=src;

    //Calculate shortest distance and predecessor to all routers in topology from source router.
     for(int v=0; v<V-1;v++)
     {
           min_index=-1;
           min_value=Integer.MAX_VALUE;
           // Find router in topology matrix which is at shortest distance to adjacent source router.
           //Router should be not visited
           for(int j=0;j<V;j++)
           {
                 if(visited[j]==false && shortDist[j] <=min_value)
                 {
                       min_value=shortDist[j];
                       min_index=j;
                 }
           }

           //Mark shortest router as visited
           visited[min_index]=true;

           //Find adjacent routers to router which shortest from source router to find shortest to
           //all routers in topology from source router.
           // Also calculates predecessor for each router in topology from source router.
           for(int k=0;k< V;k++)
           {
                 if(!visited[k] &&  graph[min_index][k]!= 0 &&  graph[min_index][k]!= -1 && shortDist[min_index]!=Integer.MAX_VALUE &&
shortDist[min_index]+(graph[min_index][k]==-1?0:graph[min_index][k]) < shortDist[k])
                 {
                       shortDist[k]=shortDist[min_index]+(graph[min_index][k]==-1?0:graph[min_index][k]);

                       preDsr[k]=min_index;

                 }
           }
     }
```

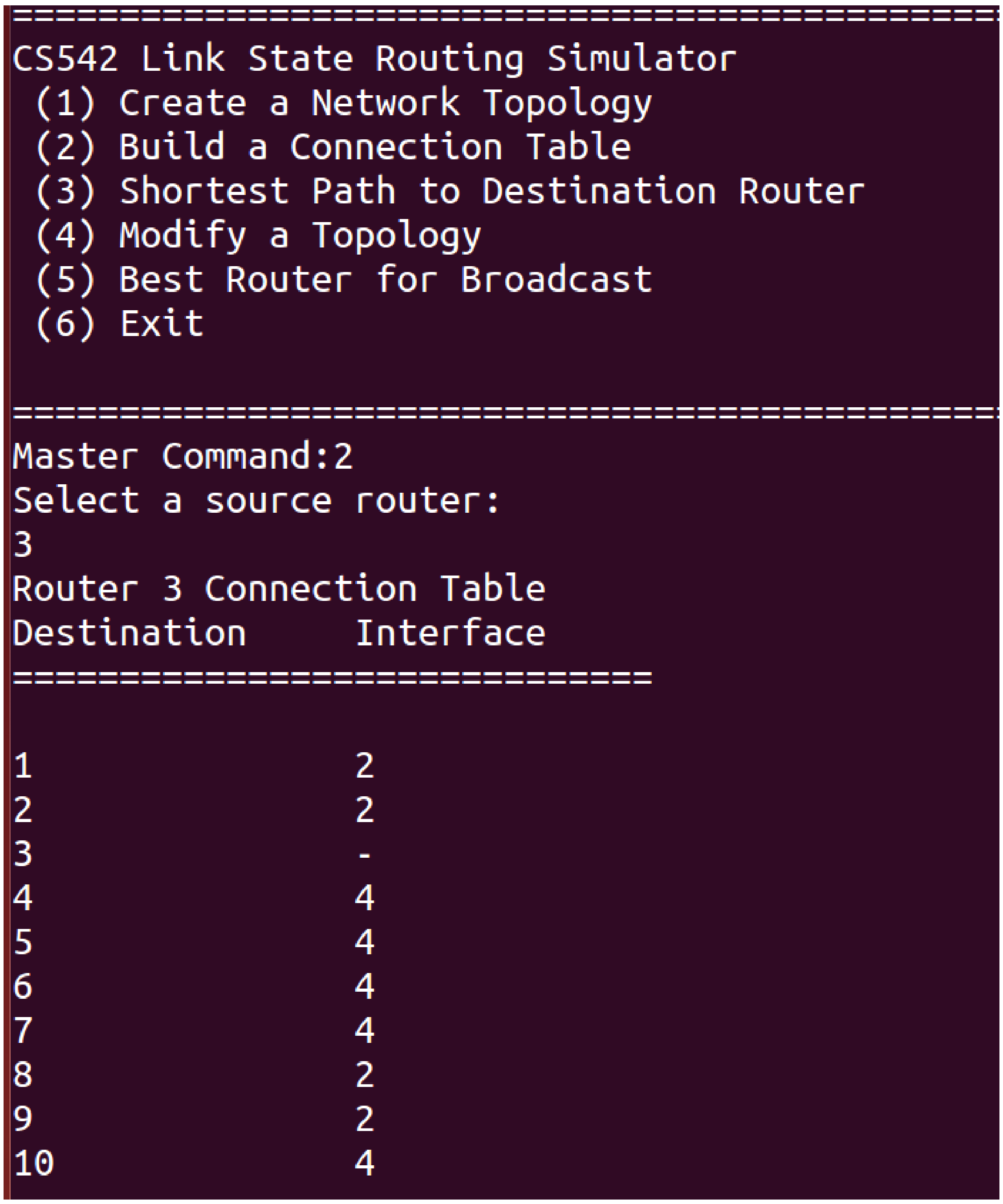
5. Test Cases.

Create Network Topology



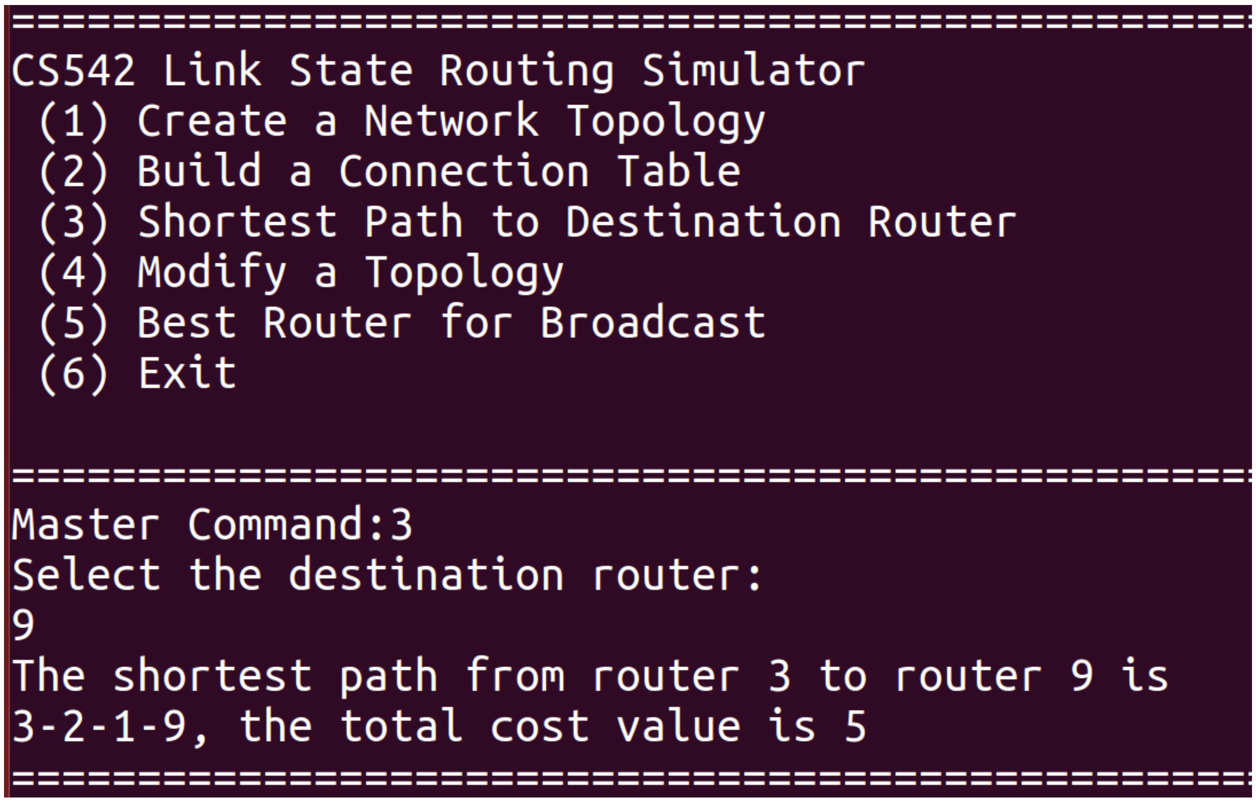
```
vishwanath@vishwanath-Q551LB:~/workspace/CS542/src$ java -jar Project_CS542.jar
==============================================================================
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

==============================================================================
Master Command:1
Input original network topology matrix data file:
10_routers.txt
==============================================================================
0       2       -1      -1      -1      -1      -1      -1      1       5

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

-1      2       0       4       -1      -1      -1      -1      -1      -1

-1      -1      4       0       2       -1      -1      -1      4       2

-1      -1      -1      2       0       -1      4       -1      -1      -1

-1      -1      -1      -1      -1      0       2       -1      -1      5

-1      -1      -1      -1      4       2       0       6       -1      -1

-1      -1      -1      -1      -1      -1      6       0       8       -1

1       -1      -1      4       -1      -1      -1      8       0       -1

5       -1      -1      2       -1      5       -1      -1      -1      0
```

Build Connection table for a Router.



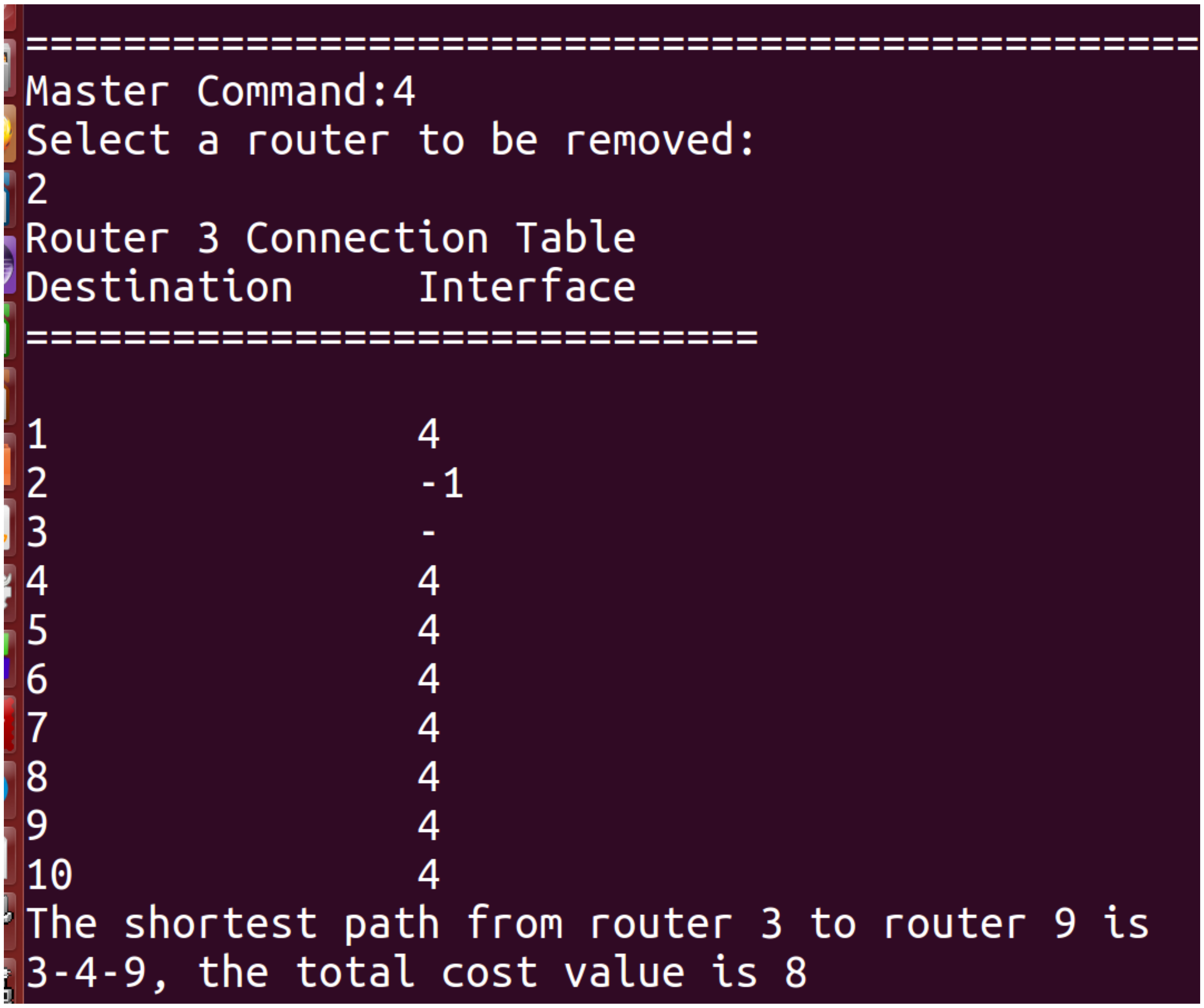
```
===============================================
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit


===============================================
Master Command:2
Select a source router:
3
Router 3 Connection Table
Destination     Interface
==============================


1               2
2               2
3               -
4               4
5               4
6               4
7               4
8               2
9               2
10              4
```

Find shortest path from above source to destination



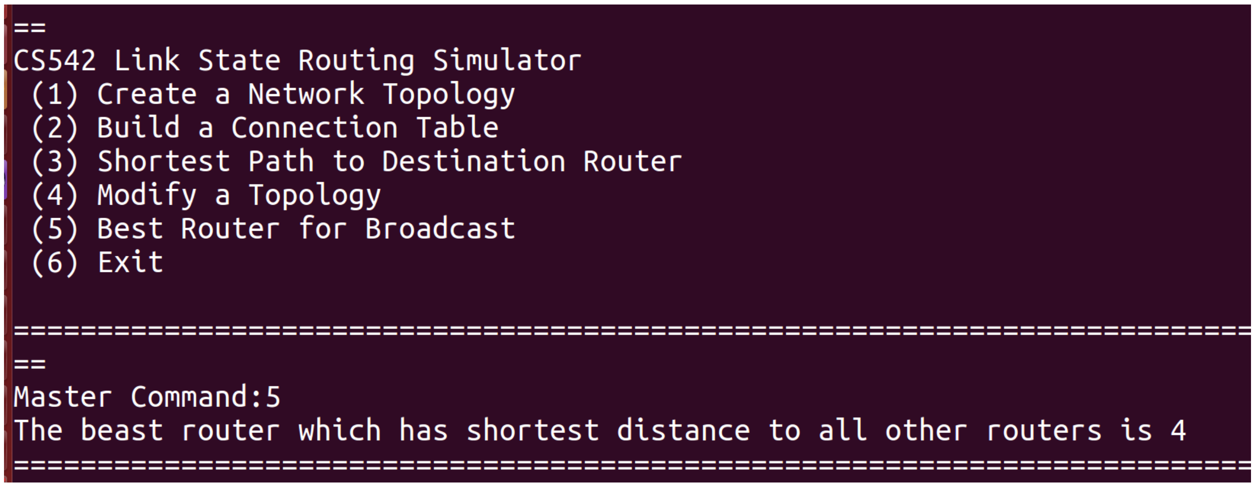
```
==================================================
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

==================================================
Master Command:3
Select the destination router:
9
The shortest path from router 3 to router 9 is
3-2-1-9, the total cost value is 5
==================================================
```

Modify network topology



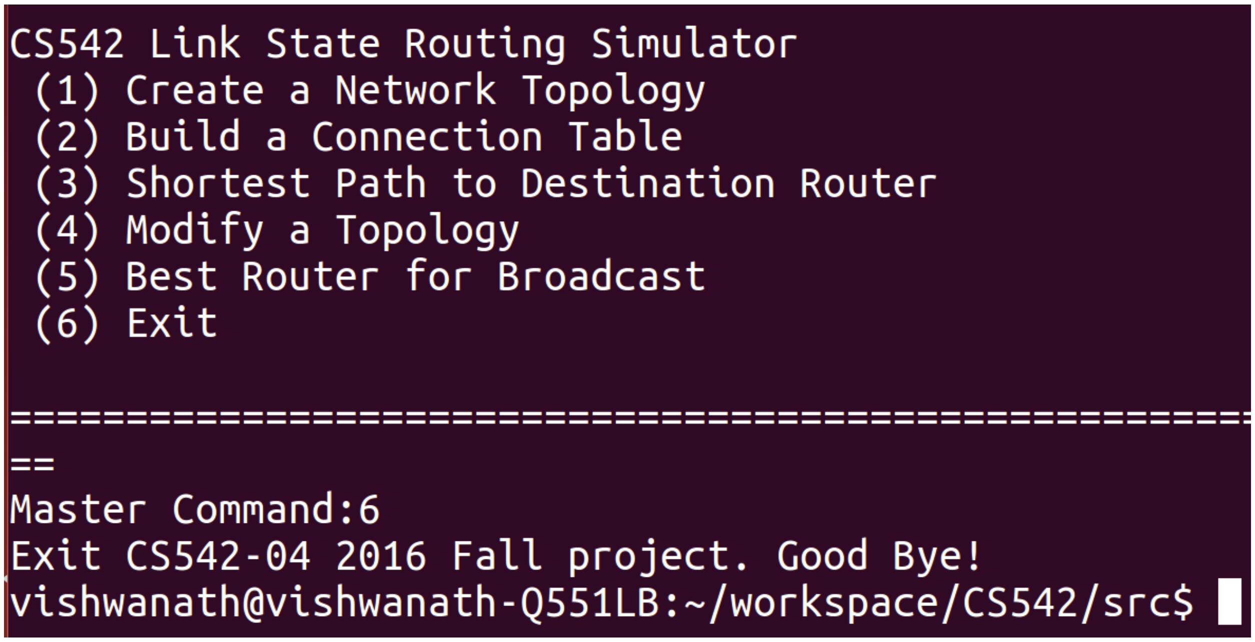
```
==================================================
Master Command:4
Select a router to be removed:
2
Router 3 Connection Table
Destination     Interface
==============================


1               4
2               -1
3               -
4               4
5               4
6               4
7               4
8               4
9               4
10              4
The shortest path from router 3 to router 9 is
3-4-9, the total cost value is 8
```

Best router in network topology



```
==
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

=====================================================================
==
Master Command:5
The beast router which has shortest distance to all other routers is 4
=====================================================================
```

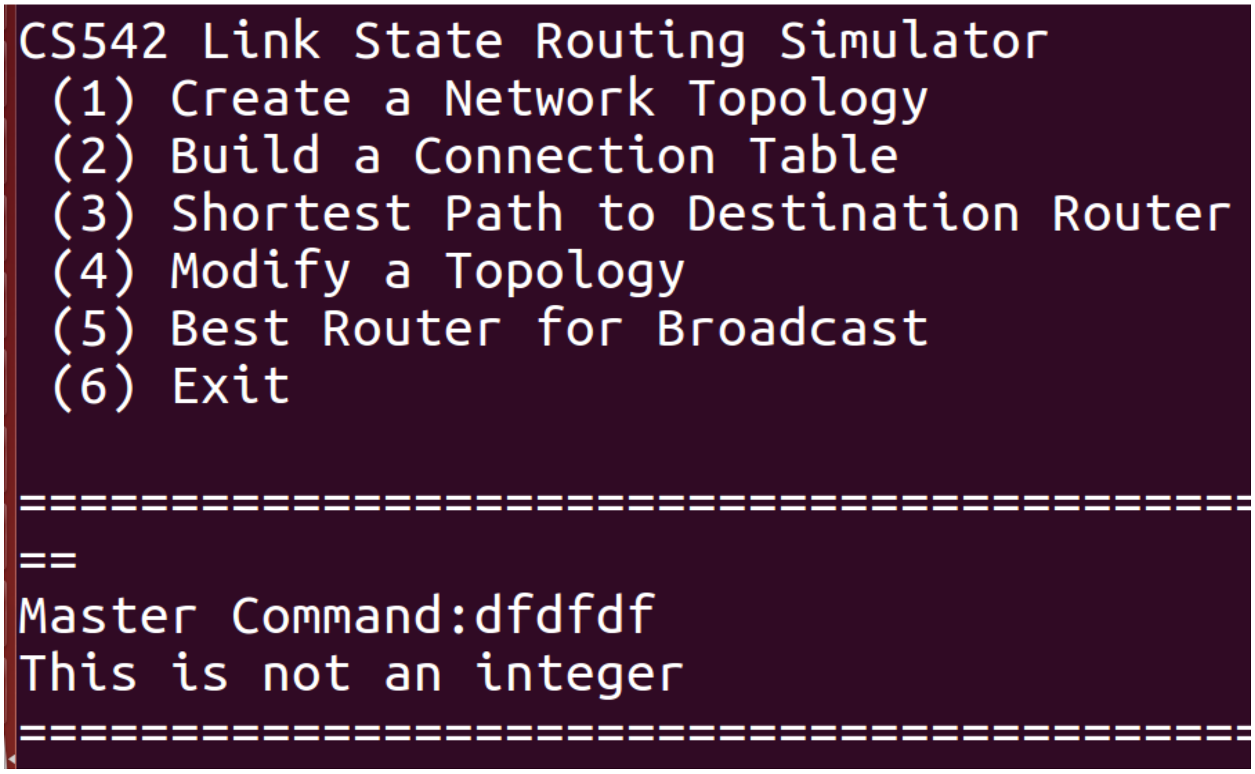
Exit from Project



```
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

==========================================================
==
Master Command:6
Exit CS542-04 2016 Fall project. Good Bye!
vishwanath@vishwanath-Q551LB:~/workspace/CS542/src$
```

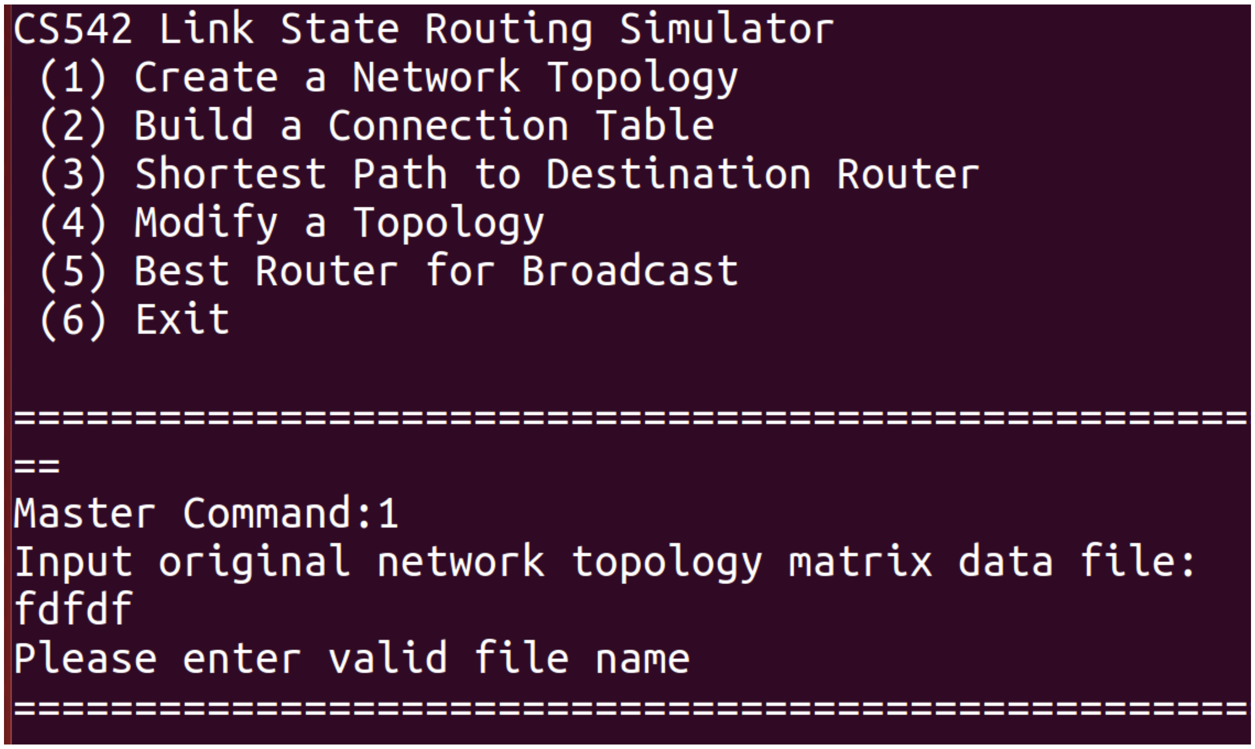
6. Error Handling

Invalid input for option selection



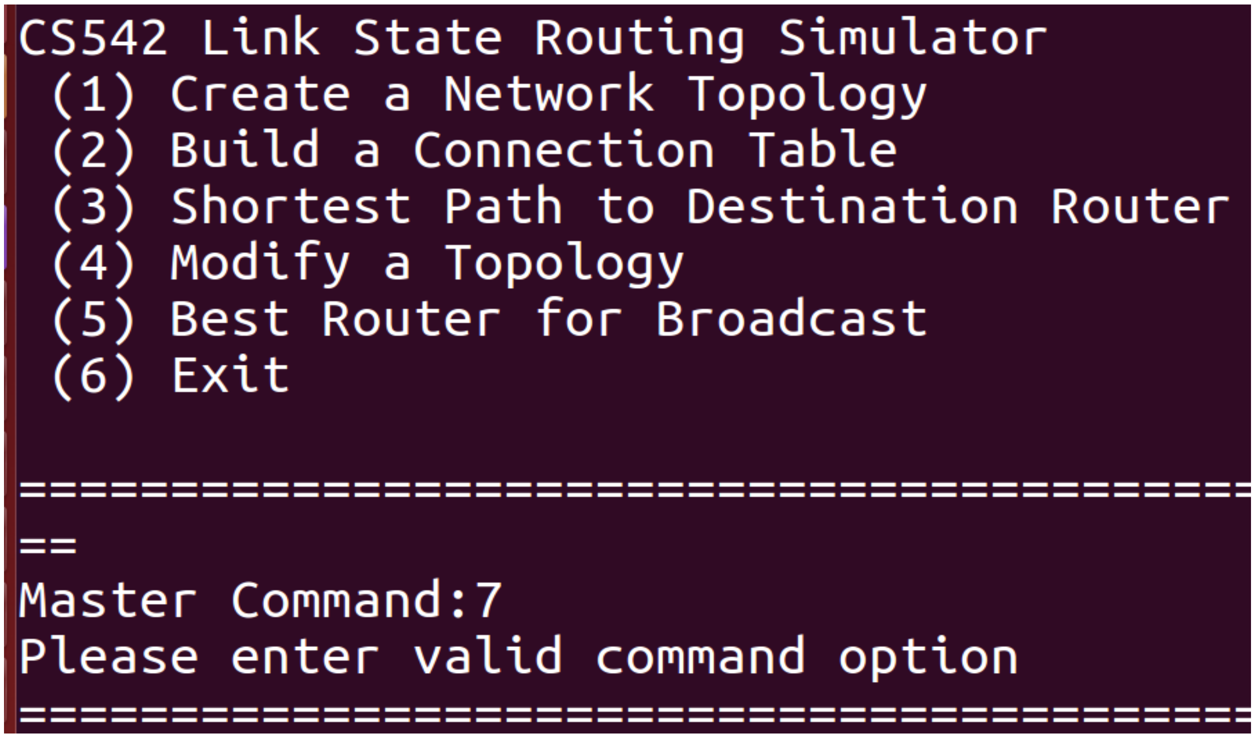
```
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

======================================
==
Master Command:dfdfdf
This is not an integer
======================================
```

Invalid file name input



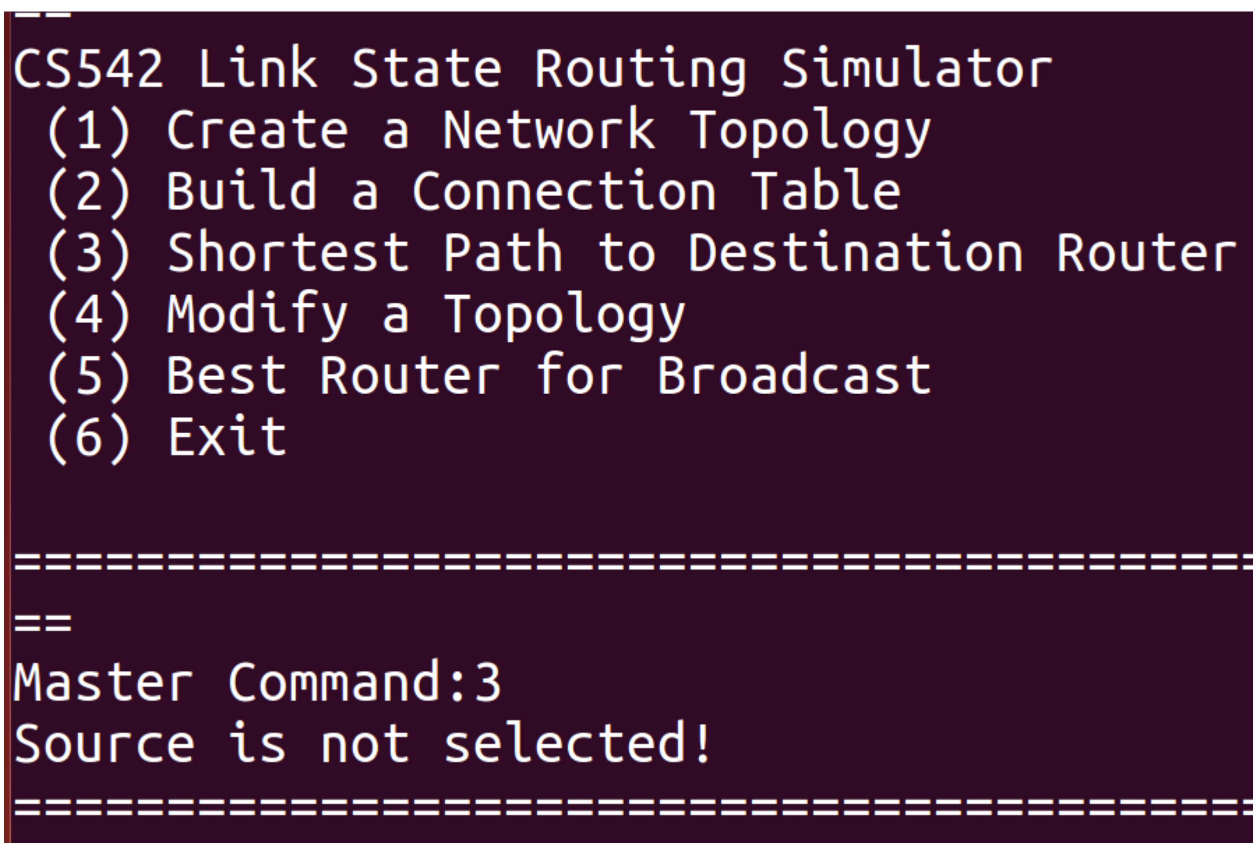
```
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

==============================================
==
Master Command:1
Input original network topology matrix data file:
fdfdf
Please enter valid file name
==============================================
```

Invalid option



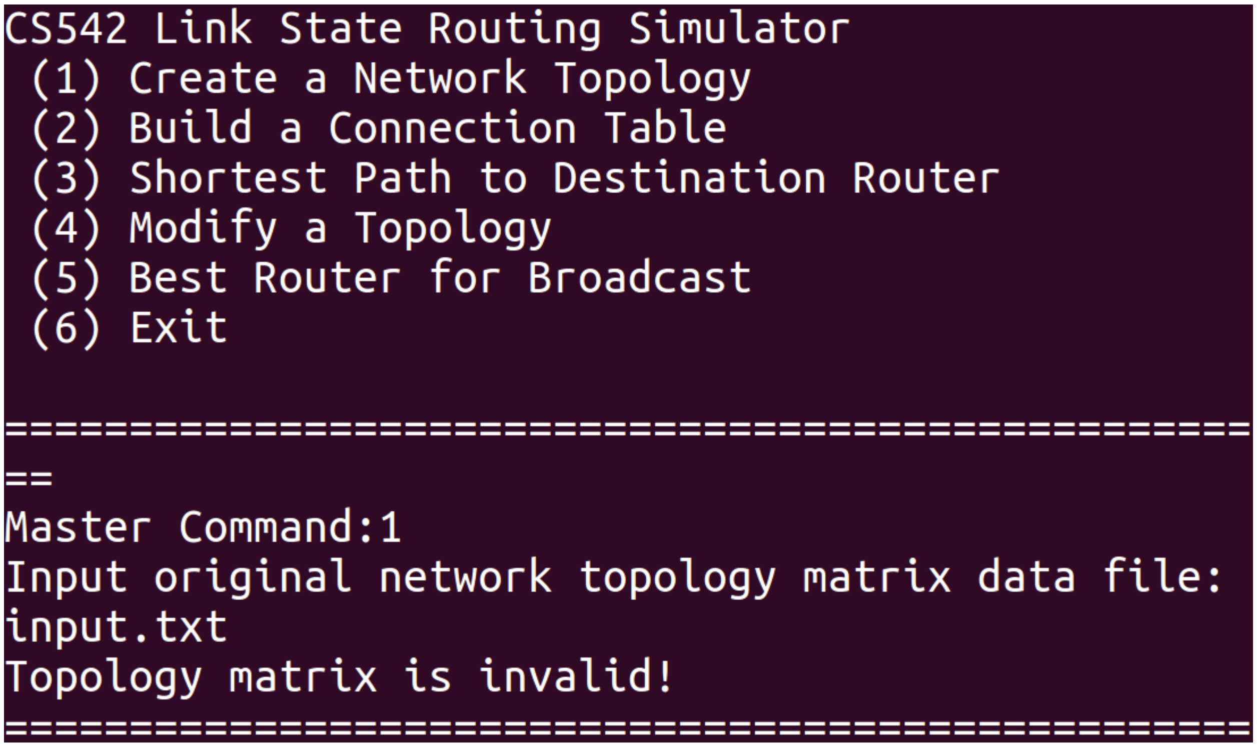
```
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

========================================
==
Master Command:7
Please enter valid command option
========================================
```

Source is not selected



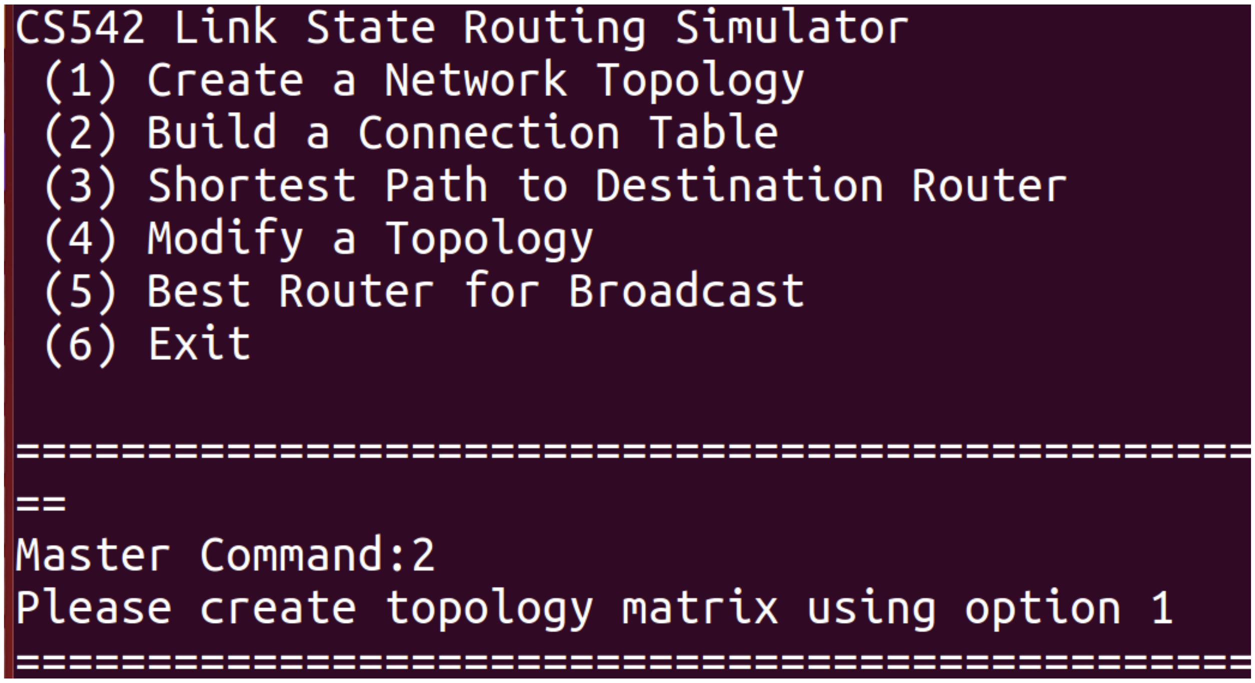
```
==
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

========================================
==
Master Command:3
Source is not selected!
========================================
```

Invalid topology matrix



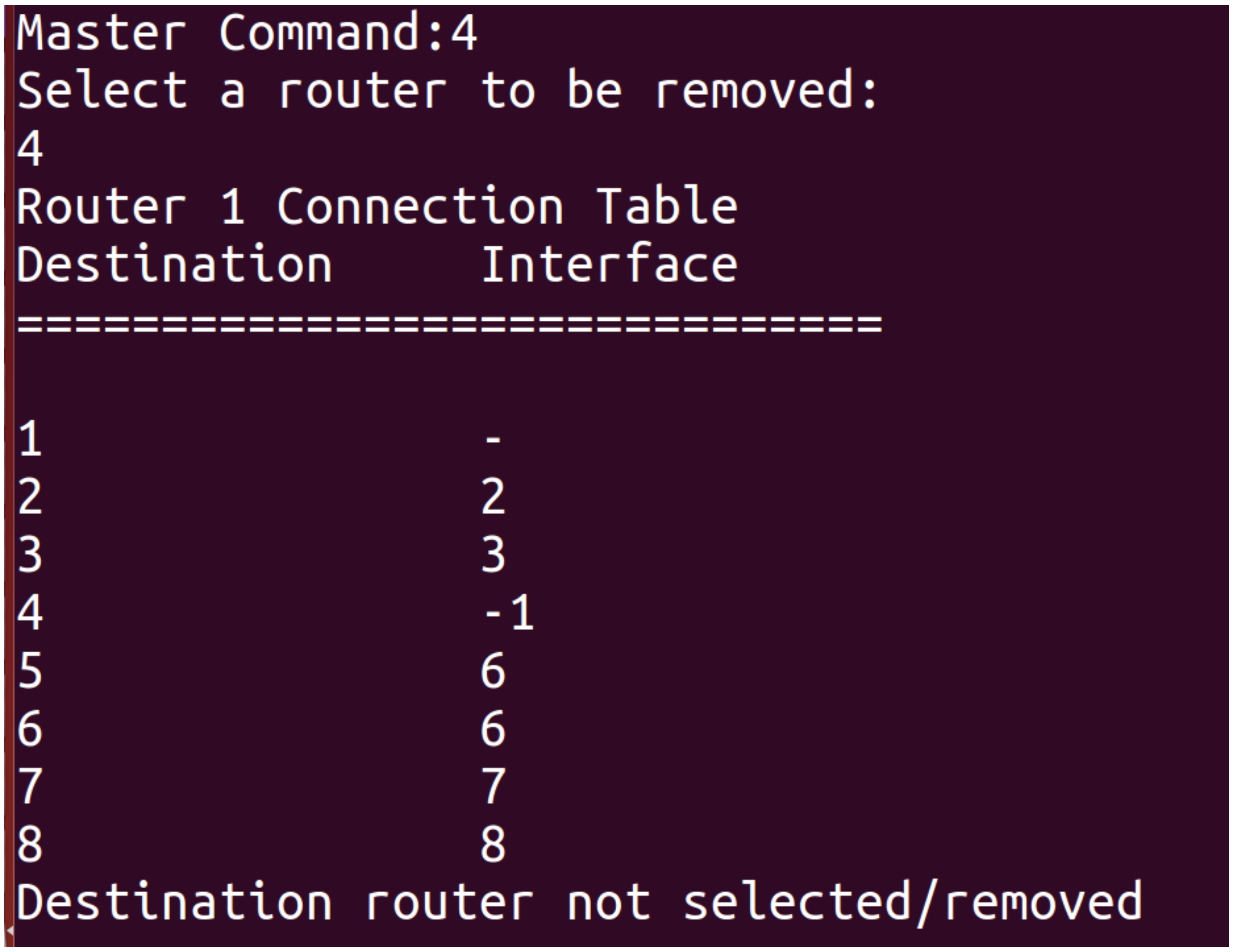
```
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

================================================
==
Master Command:1
Input original network topology matrix data file:
input.txt
Topology matrix is invalid!
================================================
```

Improper sequence of operation



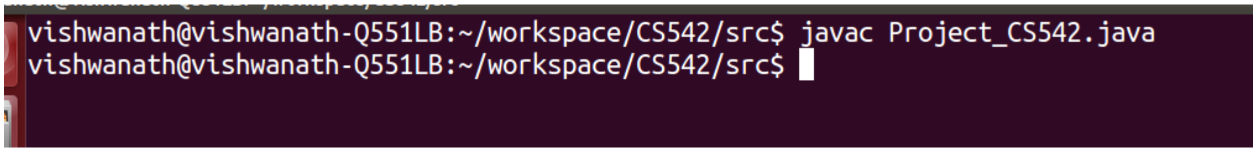
```
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

==============================================
==
Master Command:2
Please create topology matrix using option 1
==============================================
```

Destination router not selected.

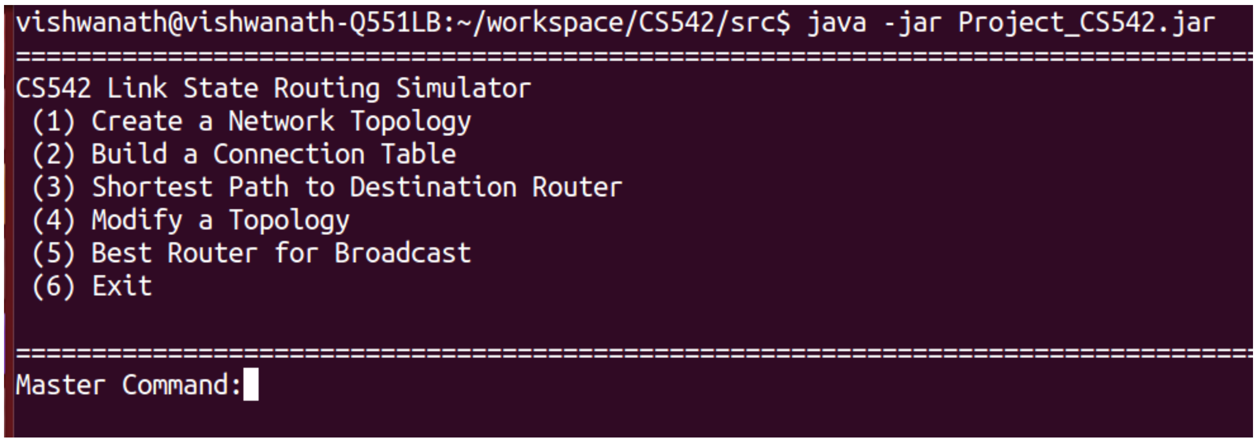


```
Master Command:4
Select a router to be removed:
4
Router 1 Connection Table
Destination     Interface
==============================


1               -
2               2
3               3
4               -1
5               6
6               6
7               7
8               8
Destination router not selected/removed
```

How to compile and run application.



```
vishwanath@vishwanath-Q551LB:~/workspace/CS542/src$ javac Project_CS542.java
vishwanath@vishwanath-Q551LB:~/workspace/CS542/src$ 
```



```
vishwanath@vishwanath-Q551LB:~/workspace/CS542/src$ java -jar Project_CS542.jar
================================================================================
CS542 Link State Routing Simulator
 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest Path to Destination Router
 (4) Modify a Topology
 (5) Best Router for Broadcast
 (6) Exit

================================================================================
Master Command:
```

References.

https://en.wikipedia.org/wiki/Link-state_routing_protocol

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm