Link State Routing Simulator

Kharage Suyog Vijaykumar

CWID – A20402686

skharage@hawk.iit.edu

Seat # 27

Link State Routing –

Link state routing protocols maintain complete road map of the network in each router running a link state routing protocol. Each router running a link state routing protocol originates information about the router, its directly connected links, and the state of those links. This information is sent to all the routers in the network as multicast messages. Link-state routing always try to maintain full networks topology by updating itself incrementally whenever a change happens in network.

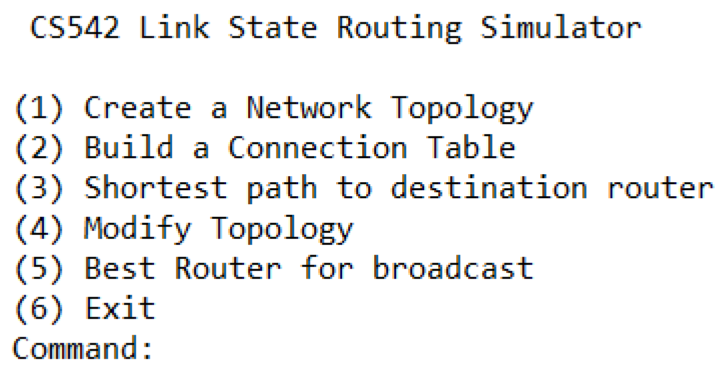
Link state protocols are based on Shortest Path First (SPF) algorithm to find the best path to a destination. Shortest Path First (SPF) algorithm is also known as Dijkstra algorithm, since it is conceptualized by Dijkstra. In Shortest Path First (SPF) algorithm, whenever a link's state changes, a routing update called a Link-State Advertisement (LSA) is exchanged between routers. When a router receives an LSA routing update, the link-state algorithm is used to recalculate the shortest path to affected destinations. Each router constructs a map of the complete network.

Dijkstra's algorithm -

Dijkstra's algorithm solves the problem of finding the shortest path from a point in a graph (the *source*) to a destination. It turns out that one can find the shortest paths from a given source to all points in a graph in the same time, hence this problem is sometimes called the single-source shortest paths problem.

Link state routing algorithm uses Dijkstra's algorithm to calculate shortest path between two routers in network topology.

After executing, this program displays the menu to perform operations on topology matrix. These are as below:



```
  CS542 Link State Routing Simulator

 (1) Create a Network Topology
 (2) Build a Connection Table
 (3) Shortest path to destination router
 (4) Modify Topology
 (5) Best Router for broadcast
 (6) Exit
 Command:
```

User is prompted to enter choice for various operations mentioned above.

To create network topology - `readInputMatrix(String filename)`

In this function, user is prompted to enter input file name of topology matrix. Then number of rows and columns are calculated and using them, costs are transferred to matrix of particular size (row, column) and it displayed to user.

To calculate shortest path to destination router - `shortestPath(matrix)`

In this function, source and destination routers are checked whether any one of them is down or not. If both routers are up, then only `Dijkstra()` function is called to calculate shortest path.
Algorithm of *Dijkstra()* -

Step 1 - Take source and destination router from user

Pass the original input topology matrix, source, destination and choice to function Dijkstra.

In function Dijkstra,

Declare and Initialize ‘distance’ array with size of matrix distance.

Declare and Initialize ‘visited’ array with size of matrix distance.

Declare and Initialize ‘predecessor’ array with size of matrix distance.

Declare variable nextRouter.

```
For every source router,
    For size of matrix,
        -Copy router's weights into distance array
    For size of predecessor,
        -fill the predecessor array with source number
    Assign source number to nextRouter.
    Mark the position of nextRouter in visited array as true.
    For size of matrix,
        - Calculate minimum cost of distance array
        - Assign value of index of minimum cost to nextRouter
        - If value of nextRouter is same as destination,
            o Do nothing
        - Else mark the position of nextRouter in visited array as true.
        - For size of distance array
            o If element of distance array is -1,
                ▪ update element of distance array with addition of element of next
                  router’s index in distance array and corresponding element of input
                  matrix if element of distance array is -1
                ▪ assign value of nextRouter to corresponding index of predecessor array.
    For size of distance array,
            - Do the addition of cost of every element and store in variable sum and store this
              sum to source index of global array tempDistance.
```

Step 2 - Calculate shortest path from source to destination router using back tracing of predecessor array's element.

To calculate connection table from router entered by router - *`buildInterfaceTable`*`(matrix,router)`

First, router number entered by user is checked whether it is down or not by checking cost of path from that router to all other routers is -1 or not. For this, separate function performs all the necessary steps. If all costs are -1, then message is displayed as 'Router is down'.

If that router is up, then only performed 1st step of above algorithm and based on path stored in predecessor array, displayed the next router on that path from source router.

To modify topology i.e. to delete the router – *`modifyTopology`*`(matrix)`

To delete the router from topology, cost of all the incoming and outgoing path from that router are set to -1 and displayed the updated topology matrix.

If cost of any path is calculated before deleting router, then cost of that route is displayed by calculating it again. This cost can be updated or as same as old value. Otherwise, it asks for the cost calculation of new path by asking source and destination router to user.

To calculate best router for broadcast- *`bestRouter`*`(matrix)`

To calculate best router for broadcast, for 'total number of router' times, Dijkstra's algorithm is called (performed only step 1 of above algorithm) and calculated the minimum of elements of tempDistance array and displayed index of that minimum element as best router for broadcast. Further, shortest path from that router to every other router is calculated using Dijkstra's algorithm.

To exit from program-

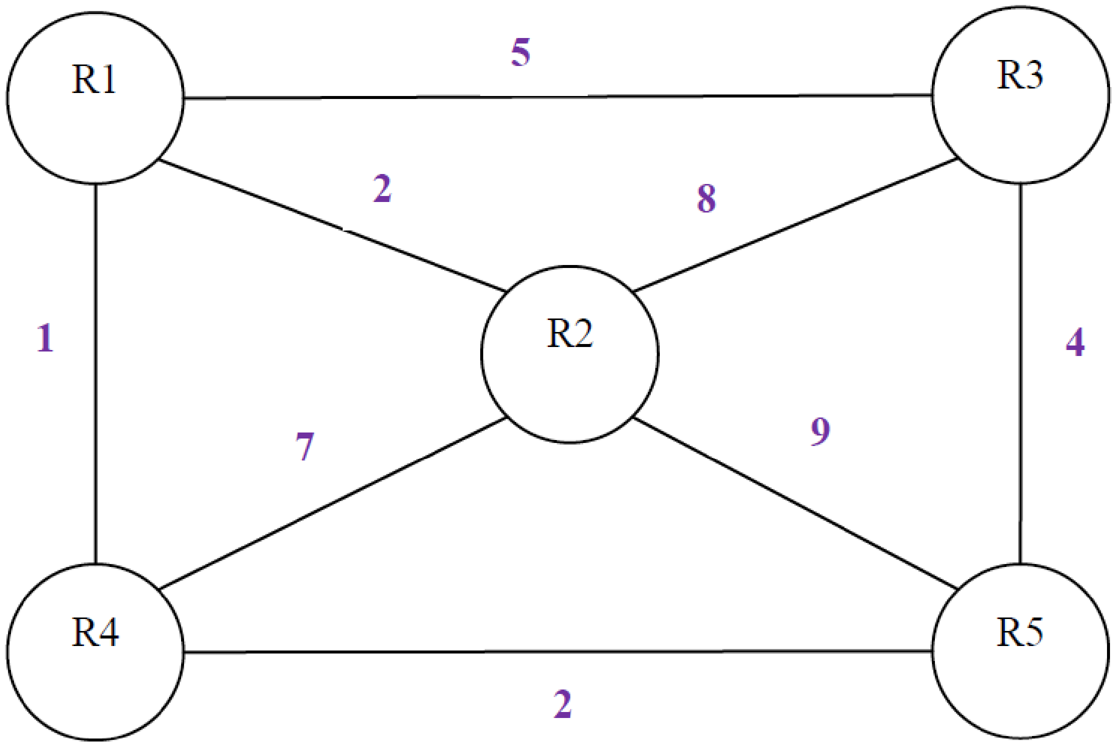
After entering command for exit, program exits and displays '`Exit CS542-04 2017 Fall project. Good Bye!`'

User Manual –

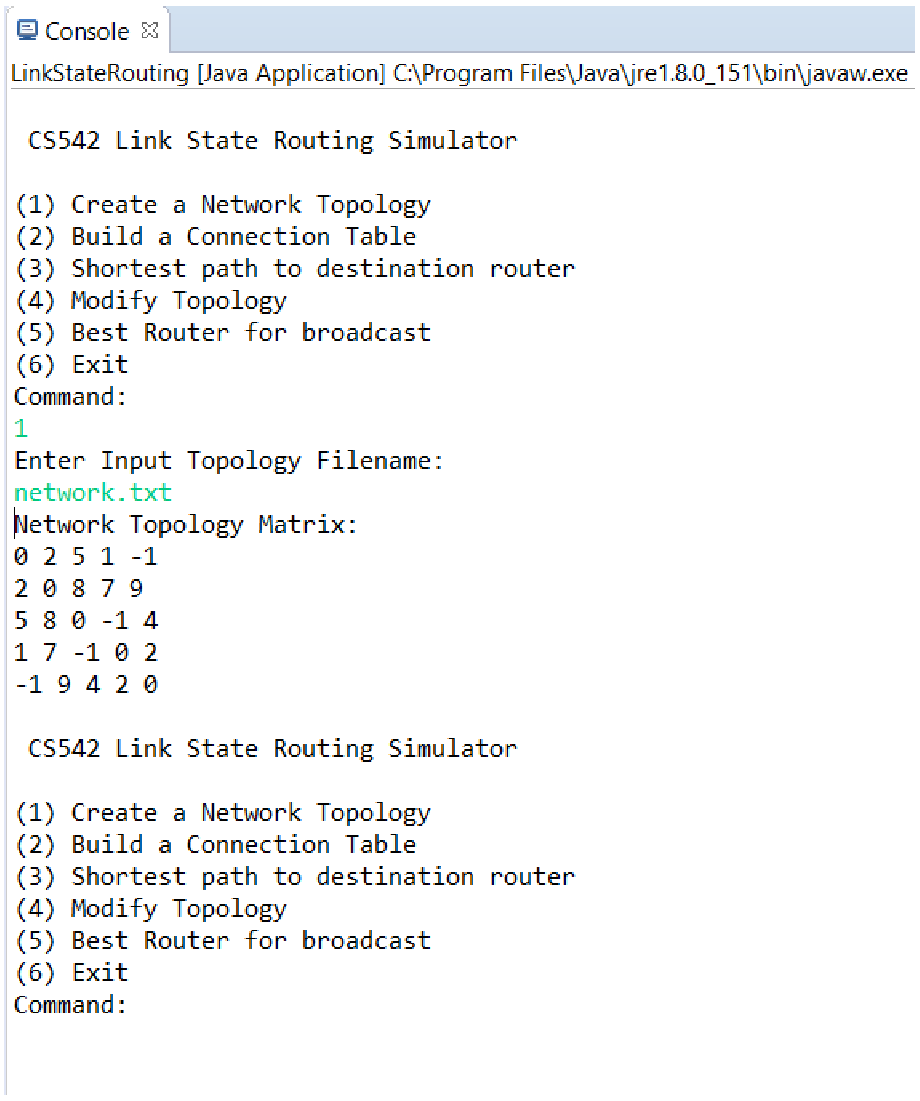
1. Import ‘LinkStateRouting.jar’ file in Eclipse IDE
2. Place ‘LinkStateRouting.jar’ file and ‘network.txt’ file in a single source folder
3. Run LinkStateRouting.java in the Eclipse
4. A menu with various operations on topology will be populated
5. User can choose any option from 6 options provided and the output

Test Report

Input network topology -



Option 1 – Create network topology

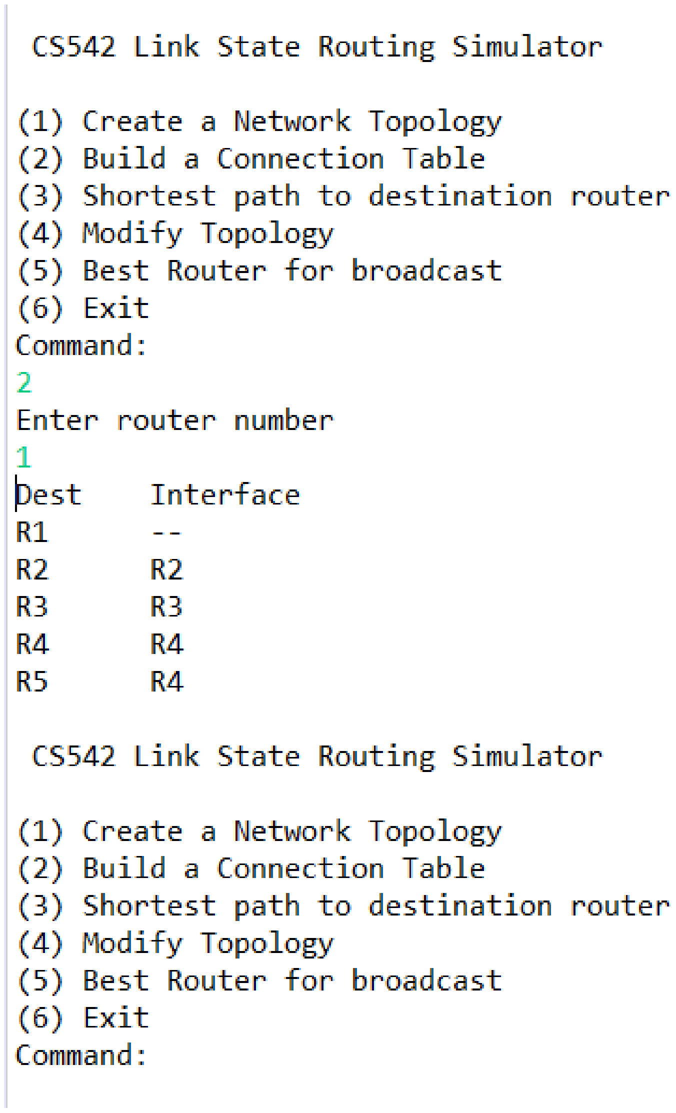


Console

LinkStateRouting [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe

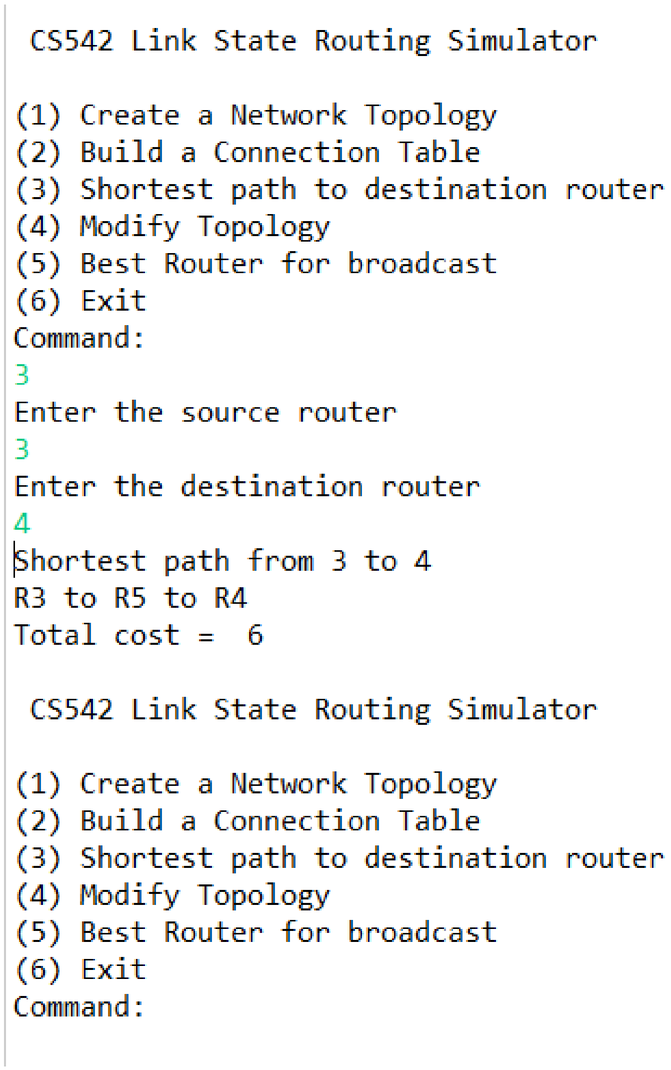
```
 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
1
Enter Input Topology Filename:
network.txt
Network Topology Matrix:
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

 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
```

Option 2 – Build a connection table



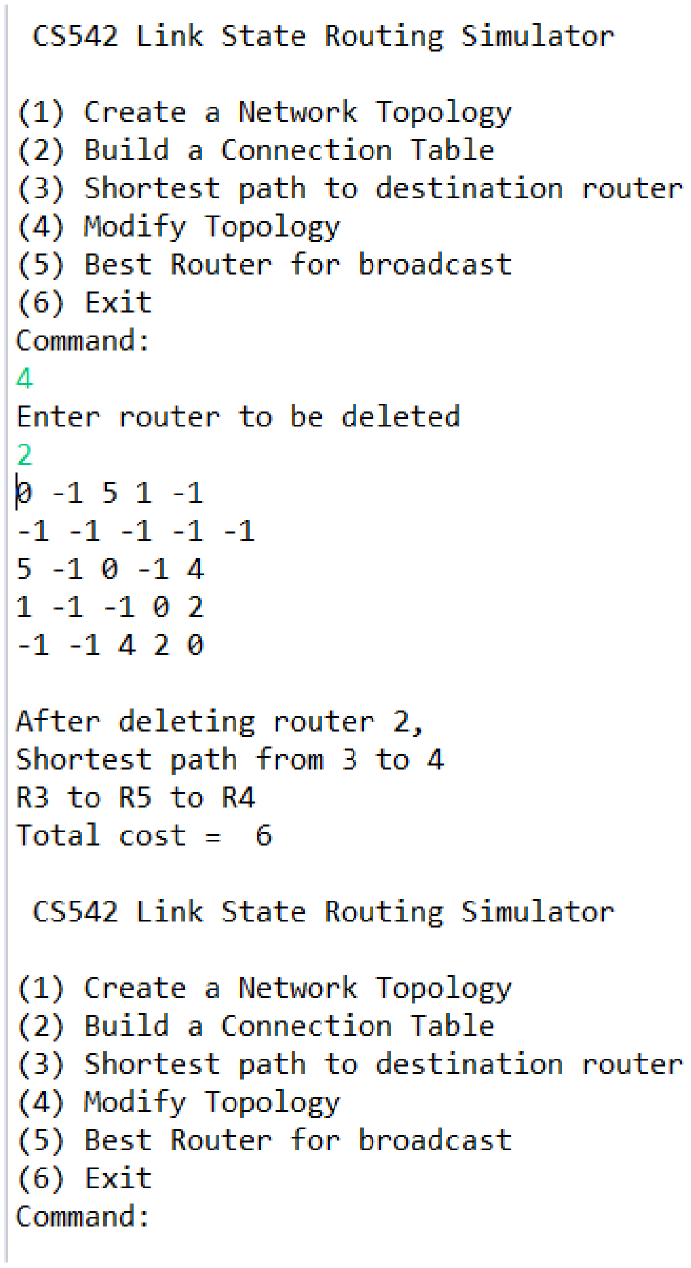
```
 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
2
Enter router number
1
Dest    Interface
R1      --
R2      R2
R3      R3
R4      R4
R5      R4

 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
```

Option 3 – Shortest path to destination router



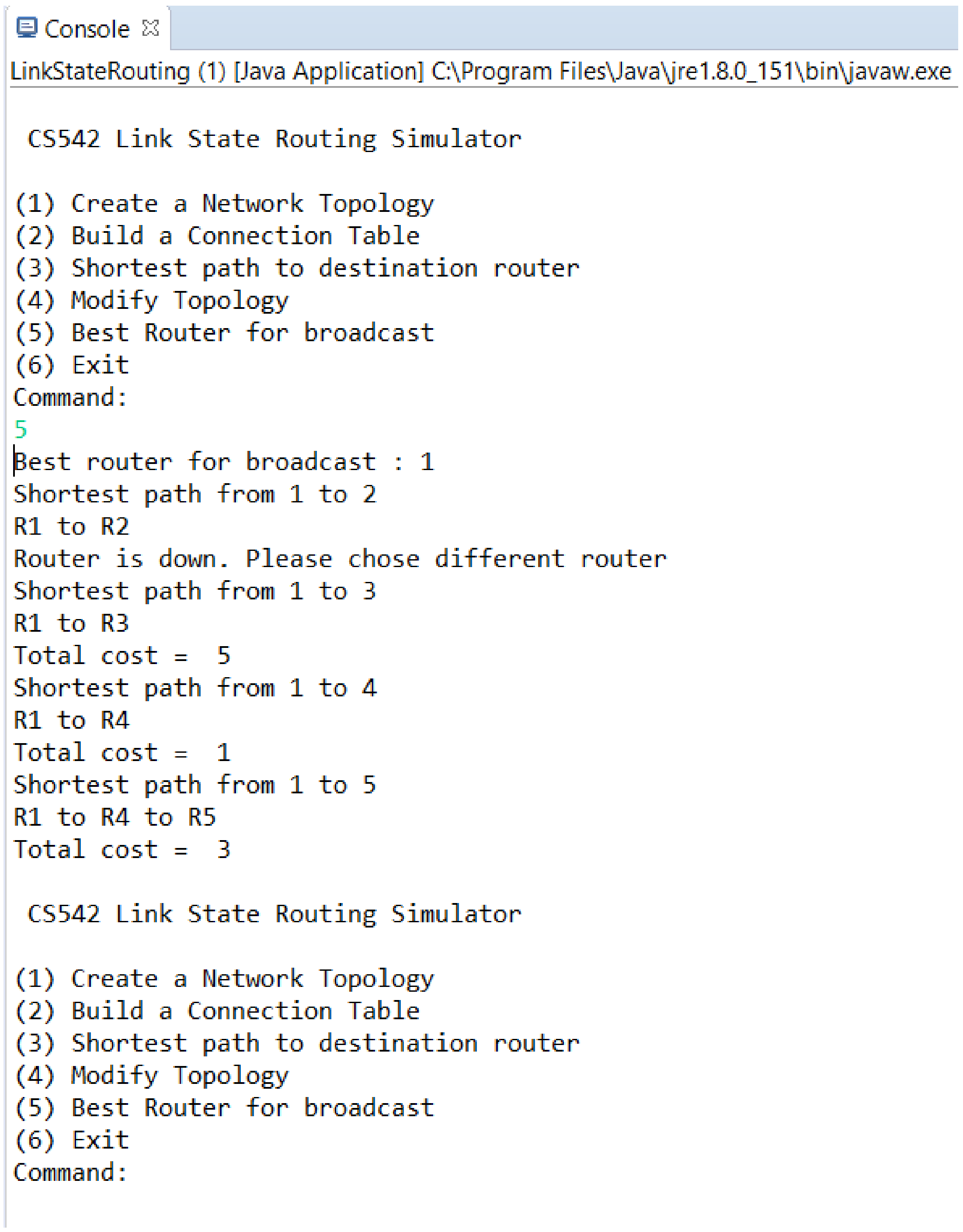
```
 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
3
Enter the source router
3
Enter the destination router
4
Shortest path from 3 to 4
R3 to R5 to R4
Total cost =  6

 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
```

Option 4 – Modify Topology



```
 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
4
Enter router to be deleted
2
0 -1 5 1 -1
-1 -1 -1 -1 -1
5 -1 0 -1 4
1 -1 -1 0 2
-1 -1 4 2 0

After deleting router 2,
Shortest path from 3 to 4
R3 to R5 to R4
Total cost =  6

 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
```

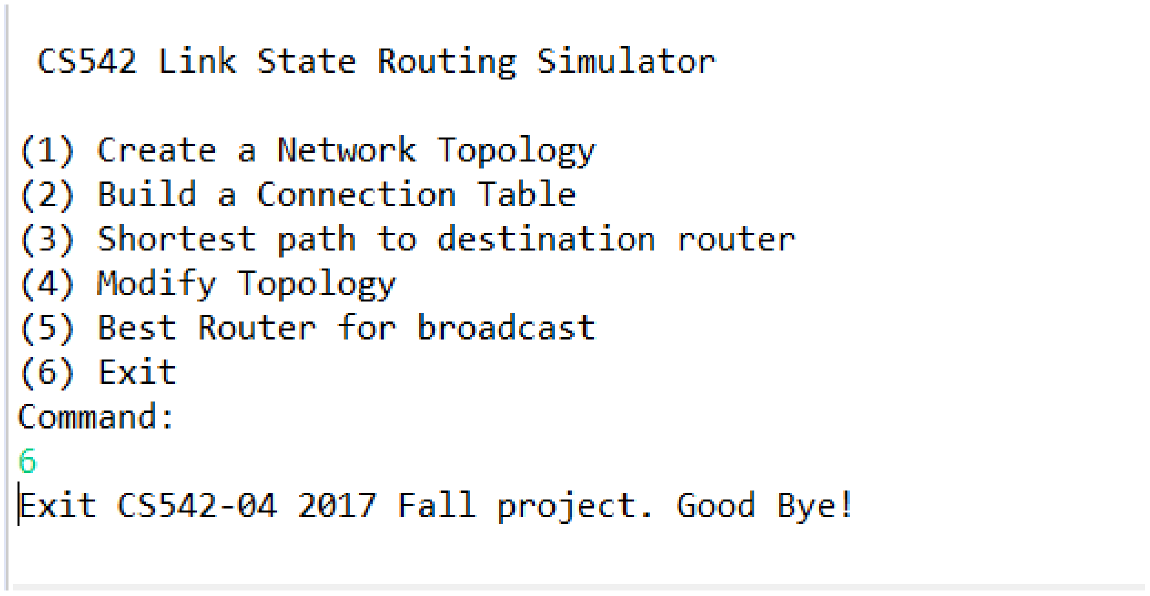
Option 5 – Best Router for Broadcast


Console

LinkStateRouting (1) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe

```
 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
5
Best router for broadcast : 1
Shortest path from 1 to 2
R1 to R2
Router is down. Please chose different router
Shortest path from 1 to 3
R1 to R3
Total cost =  5
Shortest path from 1 to 4
R1 to R4
Total cost =  1
Shortest path from 1 to 5
R1 to R4 to R5
Total cost =  3

 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
```

Option 6 - Exit



```
 CS542 Link State Routing Simulator

(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest path to destination router
(4) Modify Topology
(5) Best Router for broadcast
(6) Exit
Command:
6
Exit CS542-04 2017 Fall project. Good Bye!
```