DISTANCE VECTOR ROUTING PROTOCOL

MOHAMMAD SHOAIB ANSARI
BT21CSE063
COMPUTER NETWORKS

INTRODUCTION

Distance Vector Routing (DVR) Algorithm:

This assignment implements the Distance Vector Routing (DVR) algorithm, which is a decentralized routing protocol used in computer networks. The algorithm allows routers to communicate with their neighbors to share information about their routing tables, and then each router independently calculates the best path to reach every other destination in the network.

USAGE

1. Create a text file (e.g., input.txt) containing the network topology in the following format:

```
<number of routers>
<router names separated by spaces>
<router1> <router2> <weight>
<router1> <router3> <weight>
...
EOF
```

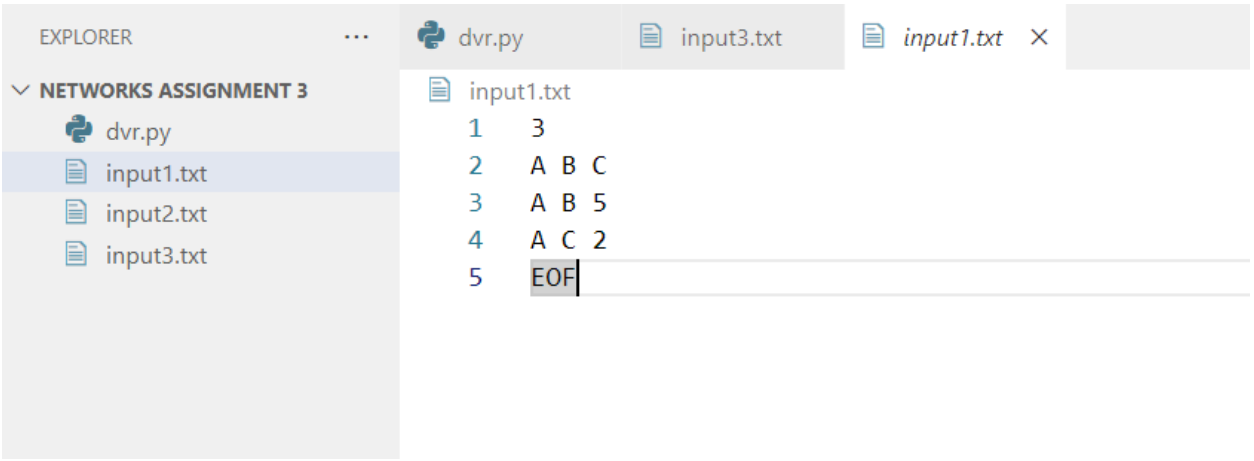
2. Run the script with the input file as an argument:

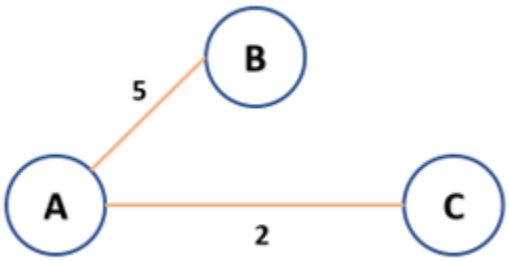
```
python dvr.py <input_file>
```

3. The script will output the initial setup information, followed by the updated routing tables after each iteration of the DVR algorithm.

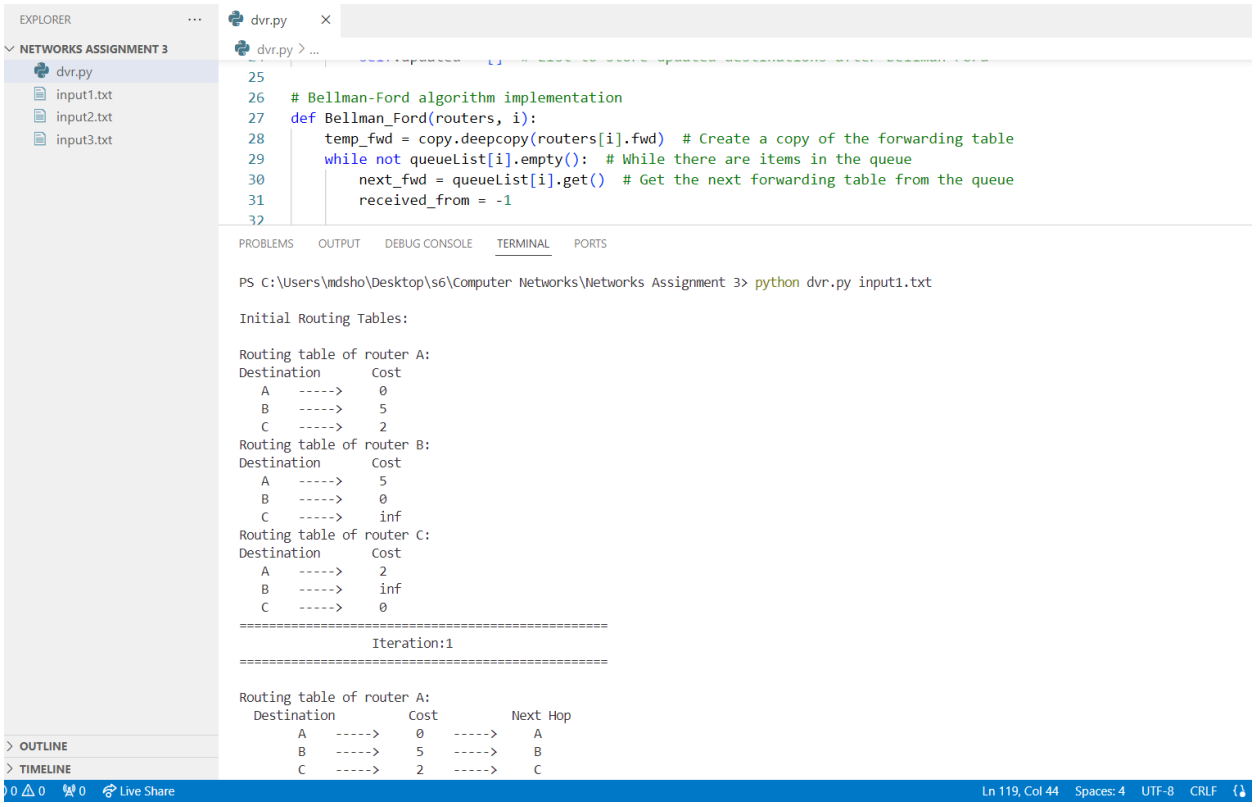
TEST CASES

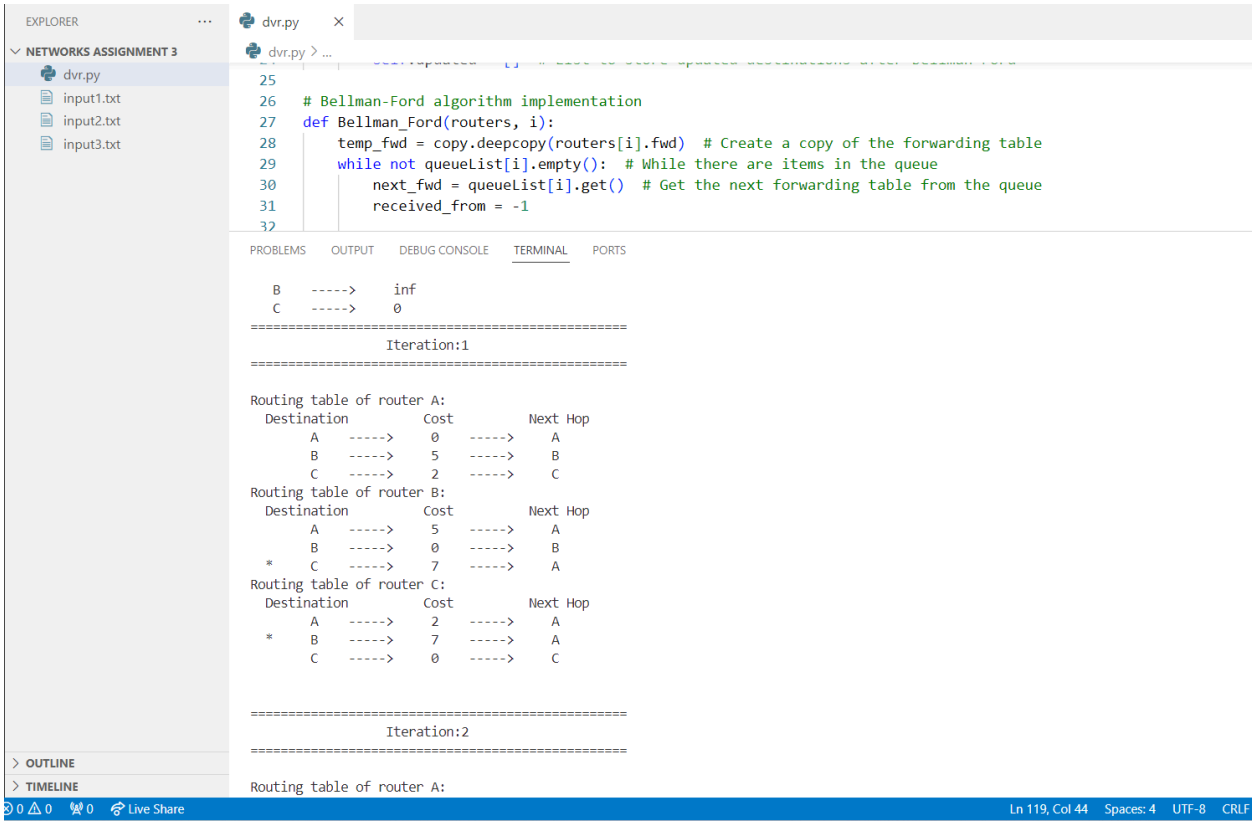
Test case 1:
Input:

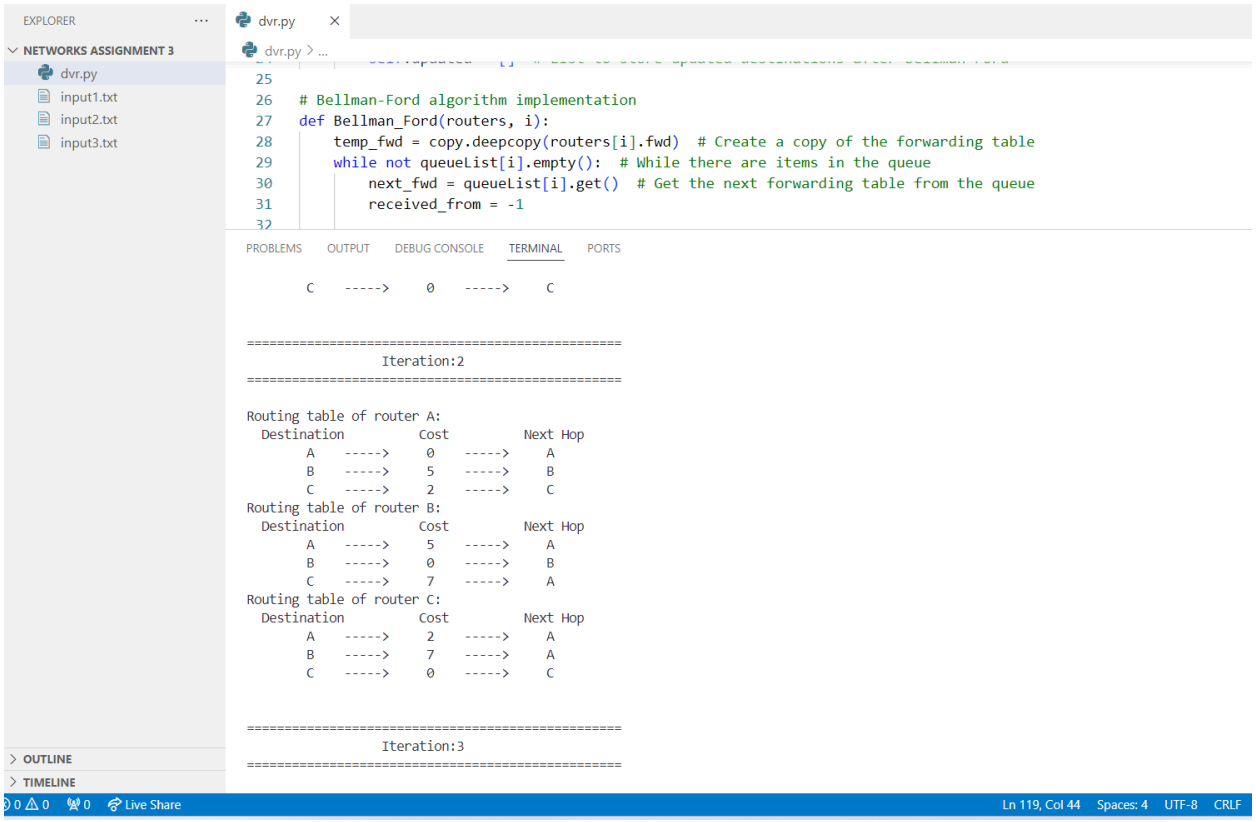


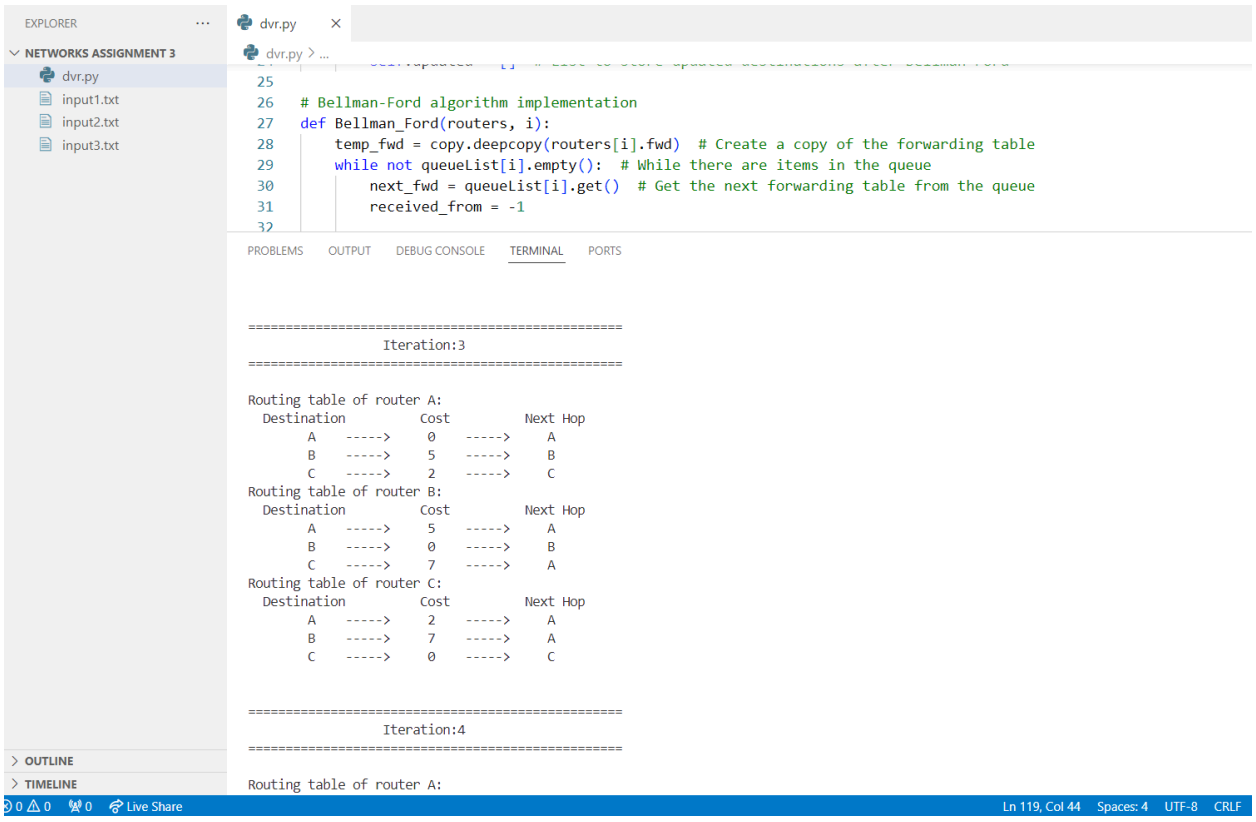


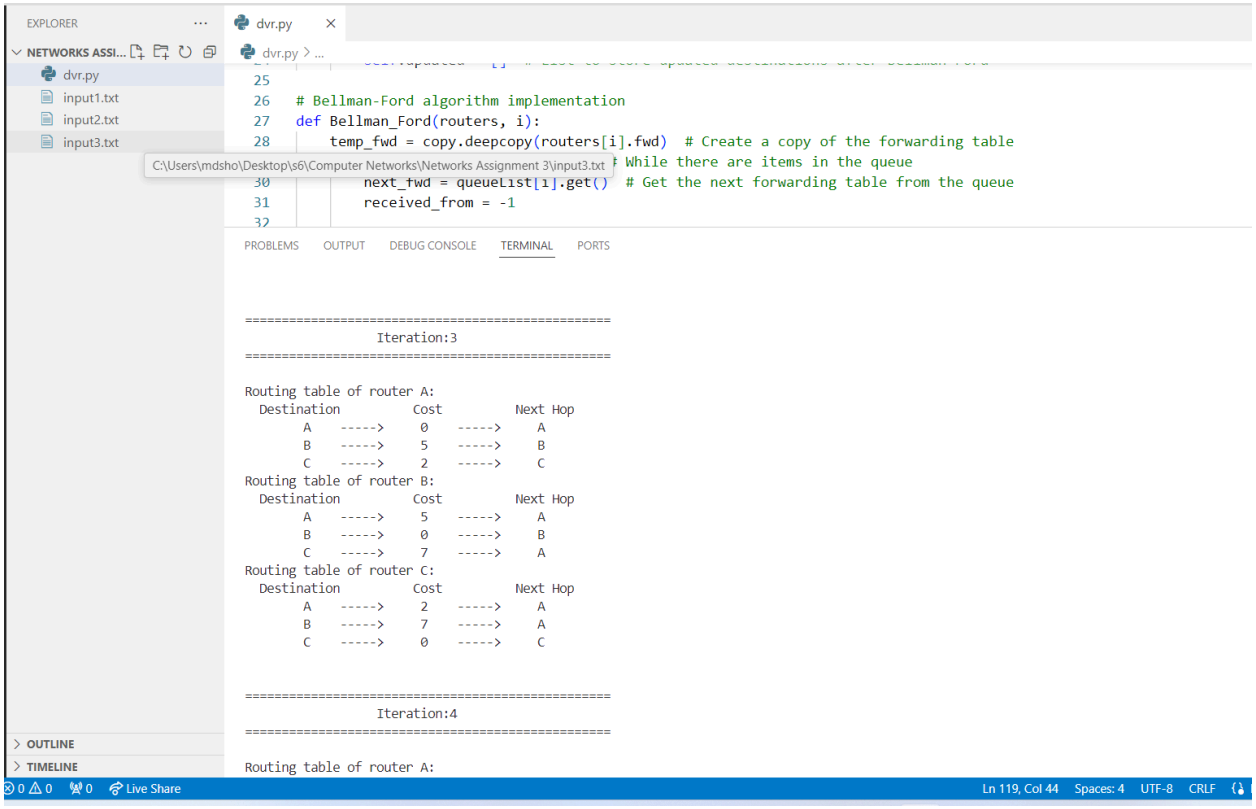
Output:


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
31 received_from = -1
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\mdsho\Desktop\s6\Computer Networks\Networks Assignment 3> python dvr.py input1.txt
Initial Routing Tables:
Routing table of router A:
Destination Cost
A -----> 0
B -----> 5
C -----> 2
Routing table of router B:
Destination Cost
A -----> 5
B -----> 0
C -----> inf
Routing table of router C:
Destination Cost
A -----> 2
B -----> inf
C -----> 0
==
Iteration:1
==
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 5 -----> B
C -----> 2 -----> C
OUTLINE
TIMELINE
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
31 received_from = -1
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
B -----> inf
C -----> 0
==
Iteration:1
==
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 5 -----> B
C -----> 2 -----> C
Routing table of router B:
Destination Cost Next Hop
A -----> 5 -----> A
B -----> 0 -----> B
* C -----> 7 -----> A
Routing table of router C:
Destination Cost Next Hop
A -----> 2 -----> A
* B -----> 7 -----> A
C -----> 0 -----> C
==
Iteration:2
==
Routing table of router A:
OUTLINE
TIMELINE
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF

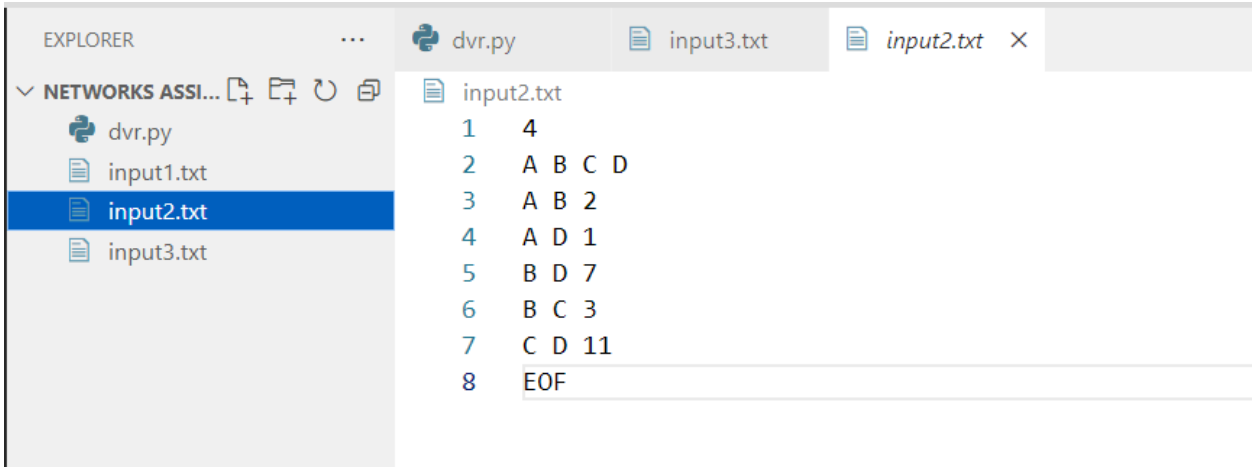

EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
31 received_from = -1
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
C -----> 0 -----> C
Iteration:2
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 5 -----> B
C -----> 2 -----> C
Routing table of router B:
Destination Cost Next Hop
A -----> 5 -----> A
B -----> 0 -----> B
C -----> 7 -----> A
Routing table of router C:
Destination Cost Next Hop
A -----> 2 -----> A
B -----> 7 -----> A
C -----> 0 -----> C
Iteration:3
OUTLINE
TIMELINE
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF

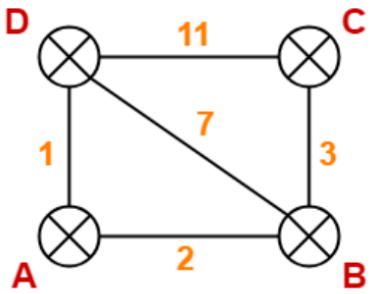

EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
31 received_from = -1
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Iteration:3
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 5 -----> B
C -----> 2 -----> C
Routing table of router B:
Destination Cost Next Hop
A -----> 5 -----> A
B -----> 0 -----> B
C -----> 7 -----> A
Routing table of router C:
Destination Cost Next Hop
A -----> 2 -----> A
B -----> 7 -----> A
C -----> 0 -----> C
Iteration:4
Routing table of router A:
OUTLINE
TIMELINE
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF



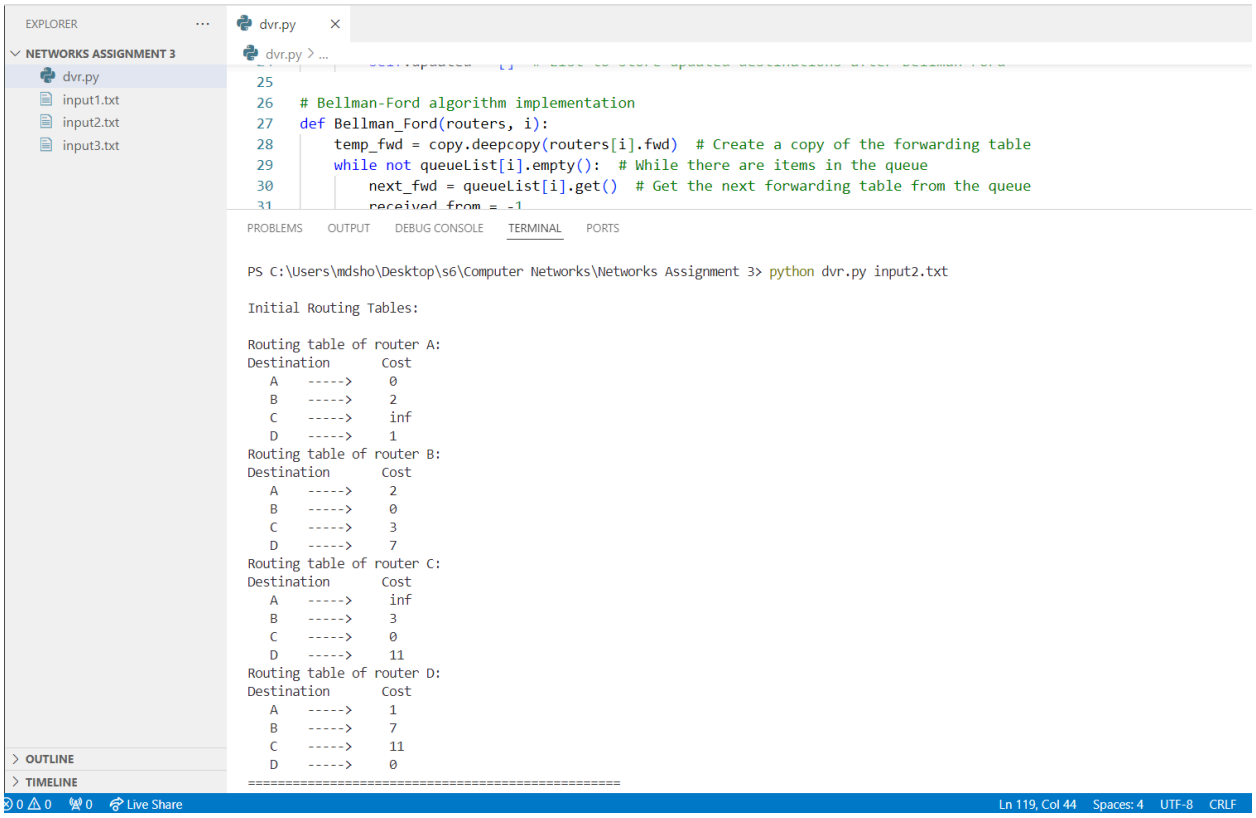
Test case 2:

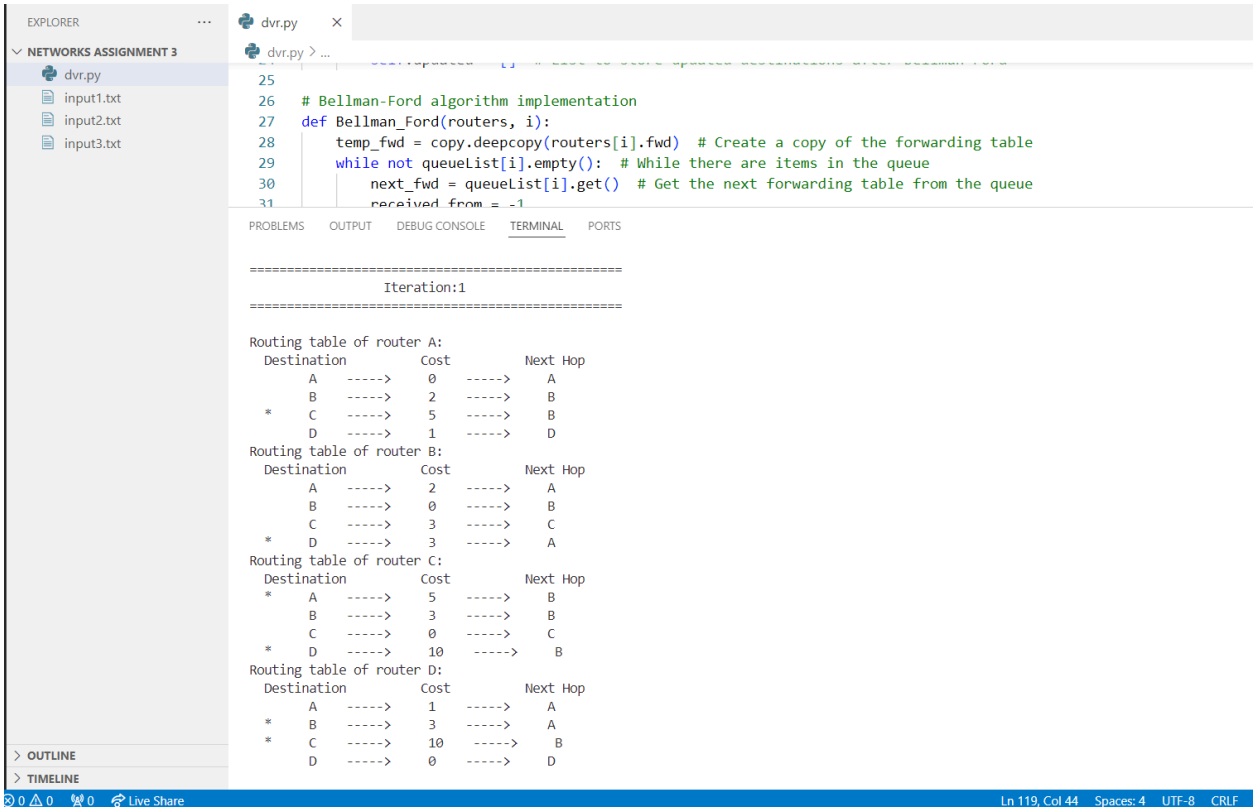
Input:

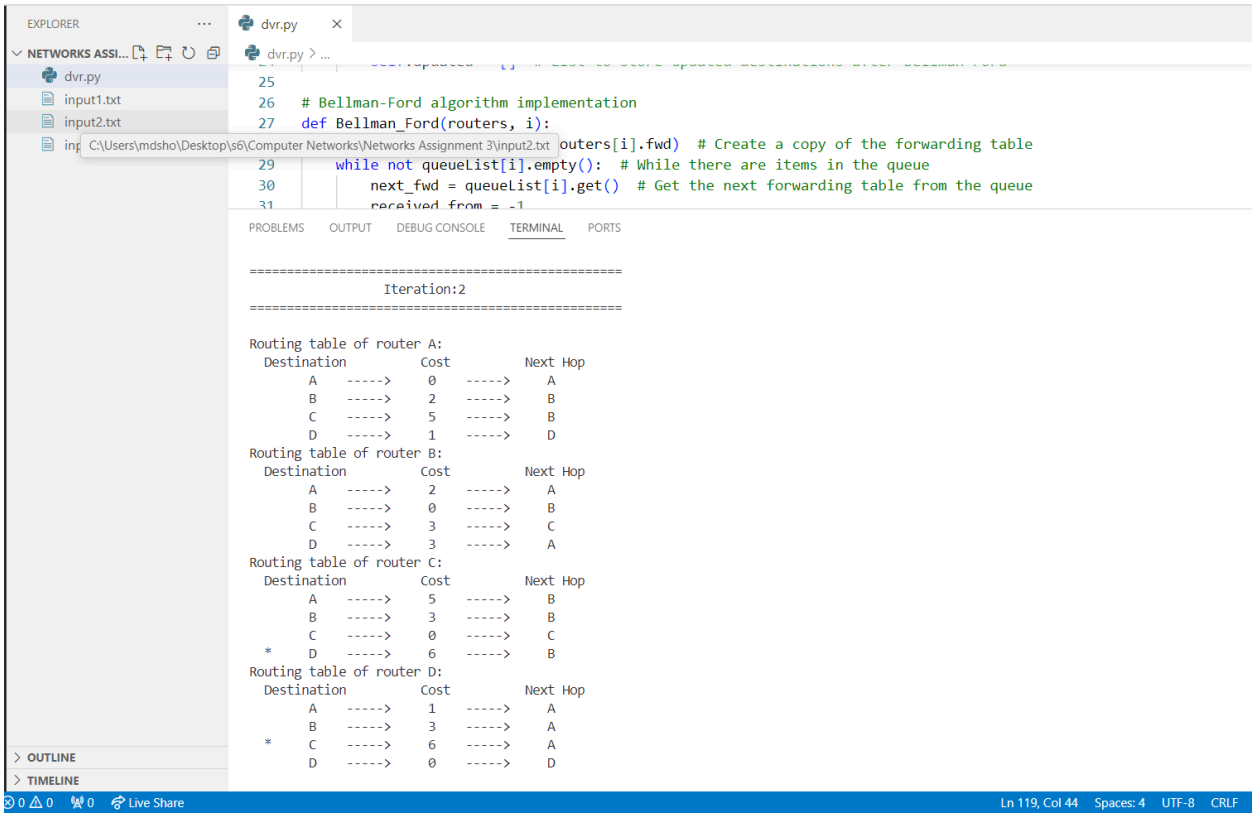


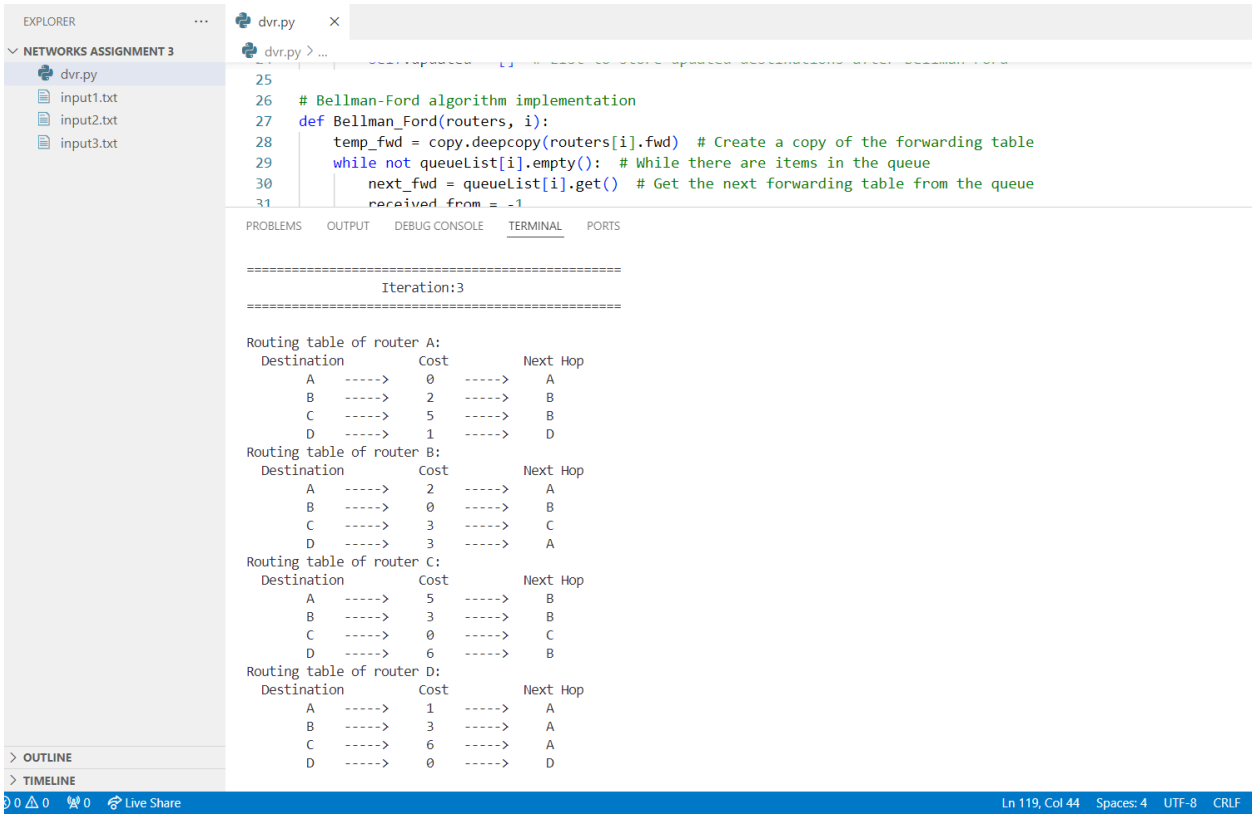

D
11
C
7
1
3
2
A
B

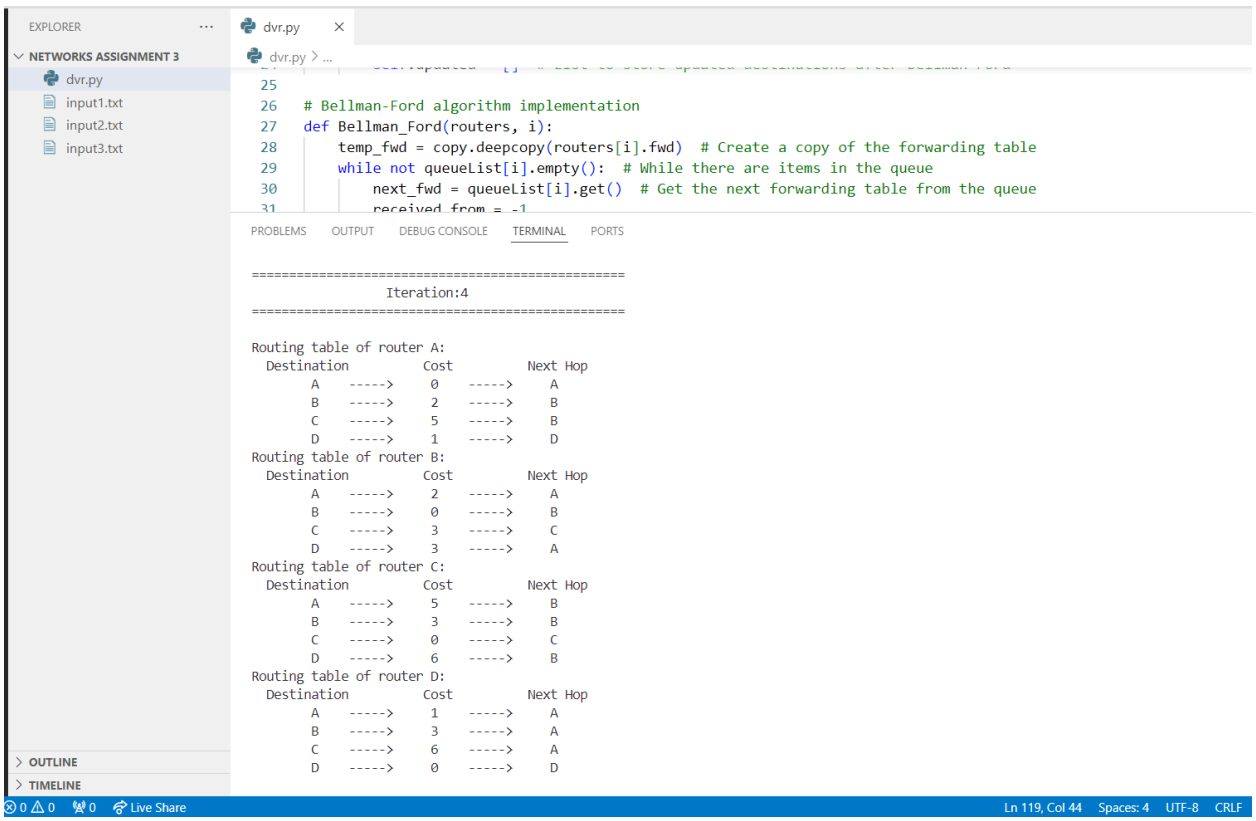
Output:


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\mdsho\Desktop\s6\Computer Networks\Networks Assignment 3> python dvr.py input2.txt
Initial Routing Tables:
Routing table of router A:
Destination Cost
A -----> 0
B -----> 2
C -----> inf
D -----> 1
Routing table of router B:
Destination Cost
A -----> 2
B -----> 0
C -----> 3
D -----> 7
Routing table of router C:
Destination Cost
A -----> inf
B -----> 3
C -----> 0
D -----> 11
Routing table of router D:
Destination Cost
A -----> 1
B -----> 7
C -----> 11
D -----> 0
==========
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF

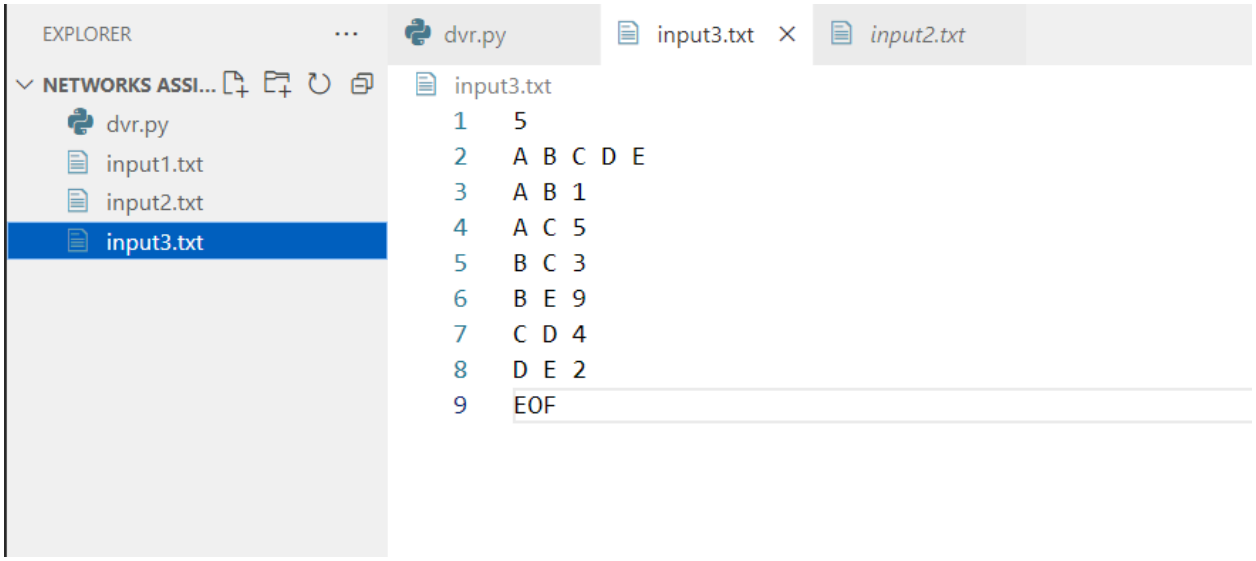

EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Iteration:1
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 2 -----> B
* C -----> 5 -----> B
D -----> 1 -----> D
Routing table of router B:
Destination Cost Next Hop
A -----> 2 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
* D -----> 3 -----> A
Routing table of router C:
Destination Cost Next Hop
* A -----> 5 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
* D -----> 10 -----> B
Routing table of router D:
Destination Cost Next Hop
A -----> 1 -----> A
* B -----> 3 -----> A
* C -----> 10 -----> B
D -----> 0 -----> D
OUTLINE
TIMELINE
Live Share
Ln 119, Col 44
Spaces: 4
UTF-8
CRLF

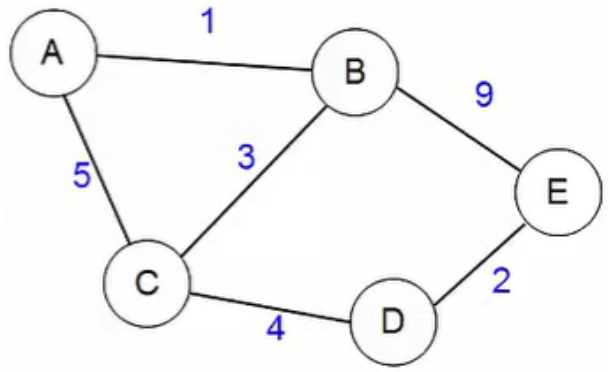

EXPLORER
NETWORKS ASSI...
dvr.py
input1.txt
input2.txt
C:\Users\mdsho\Desktop\s6\Computer Networks\Networks Assignment 3\input2.txt
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
29 while not queueList[i].empty(): # While there are items in the queue
30 next_fwd = queueList[i].get() # Get the next forwarding table from the queue
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Iteration:2
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 2 -----> B
C -----> 5 -----> B
D -----> 1 -----> D
Routing table of router B:
Destination Cost Next Hop
A -----> 2 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
D -----> 3 -----> A
Routing table of router C:
Destination Cost Next Hop
A -----> 5 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
* D -----> 6 -----> B
Routing table of router D:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 3 -----> A
* C -----> 6 -----> A
D -----> 0 -----> D
OUTLINE
TIMELINE
Live Share
Ln 119, Col 44
Spaces: 4
UTF-8
CRLF


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
# Bellman-Ford algorithm implementation
def Bellman_Ford(routers, i):
temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
while not queueList[i].empty(): # While there are items in the queue
next_fwd = queueList[i].get() # Get the next forwarding table from the queue
received_from = -1
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Iteration:3
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 2 -----> B
C -----> 5 -----> B
D -----> 1 -----> D
Routing table of router B:
Destination Cost Next Hop
A -----> 2 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
D -----> 3 -----> A
Routing table of router C:
Destination Cost Next Hop
A -----> 5 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
D -----> 6 -----> B
Routing table of router D:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 3 -----> A
C -----> 6 -----> A
D -----> 0 -----> D
Live Share
Ln 119, Col 44
Spaces: 4
UTF-8
CRLF

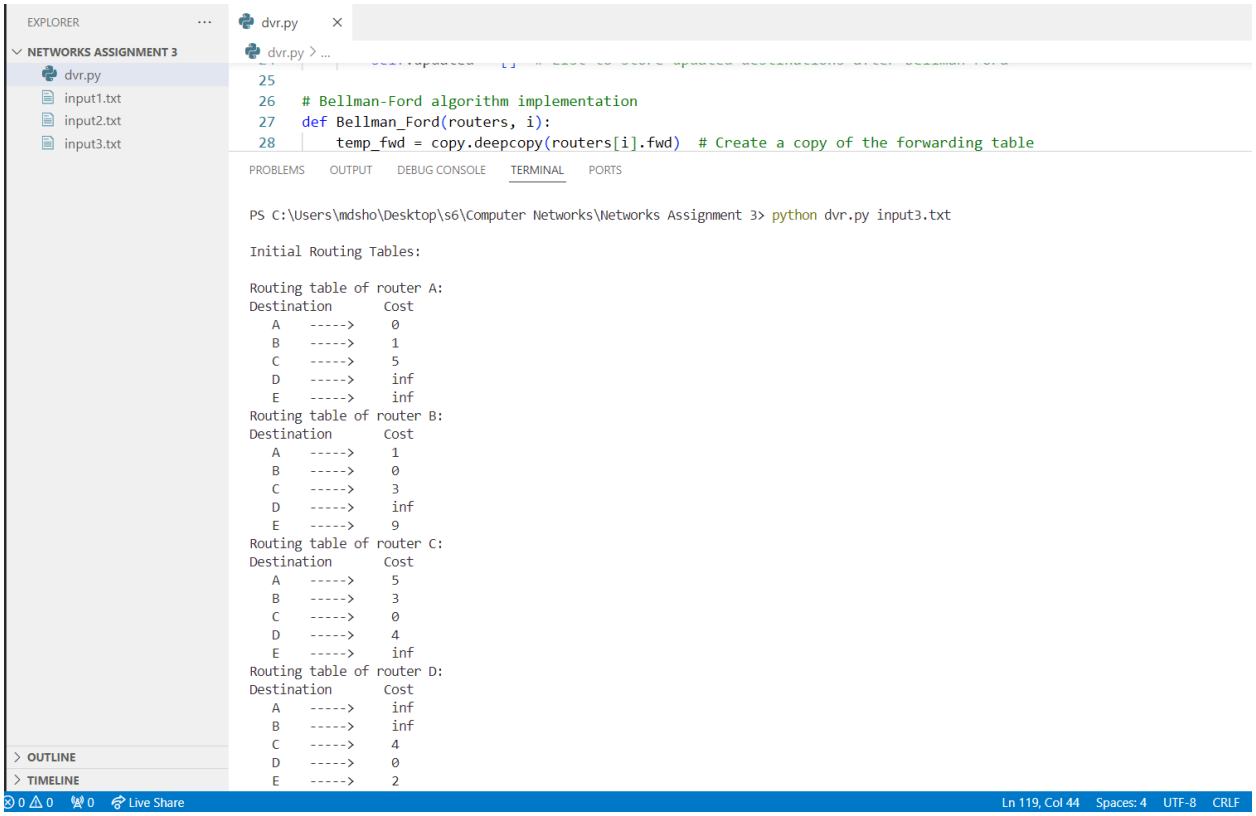

EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
# Bellman-Ford algorithm implementation
def Bellman_Ford(routers, i):
temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
while not queueList[i].empty(): # While there are items in the queue
next_fwd = queueList[i].get() # Get the next forwarding table from the queue
received_from = -1
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Iteration:4
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 2 -----> B
C -----> 5 -----> B
D -----> 1 -----> D
Routing table of router B:
Destination Cost Next Hop
A -----> 2 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
D -----> 3 -----> A
Routing table of router C:
Destination Cost Next Hop
A -----> 5 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
D -----> 6 -----> B
Routing table of router D:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 3 -----> A
C -----> 6 -----> A
D -----> 0 -----> D
Live Share
Ln 119, Col 44
Spaces: 4
UTF-8
CRLF

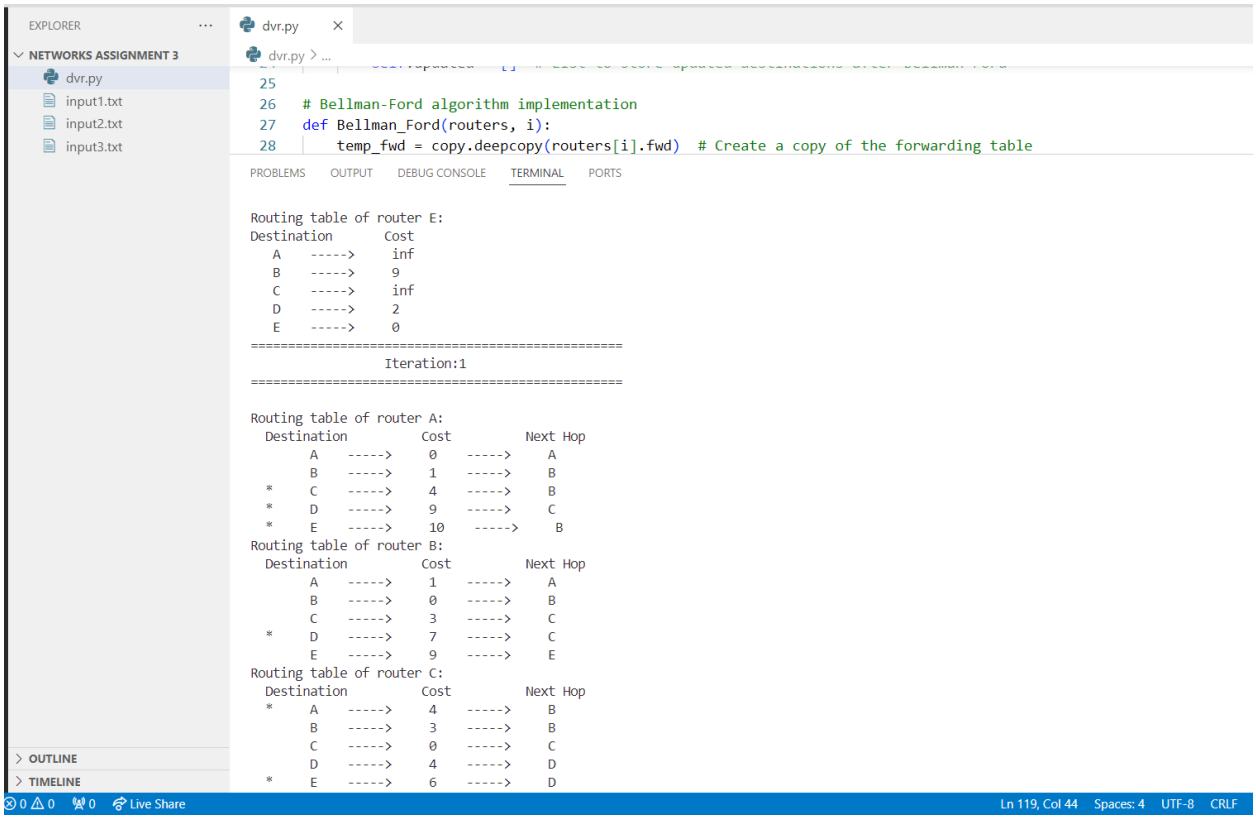
Test case 3:
Input:

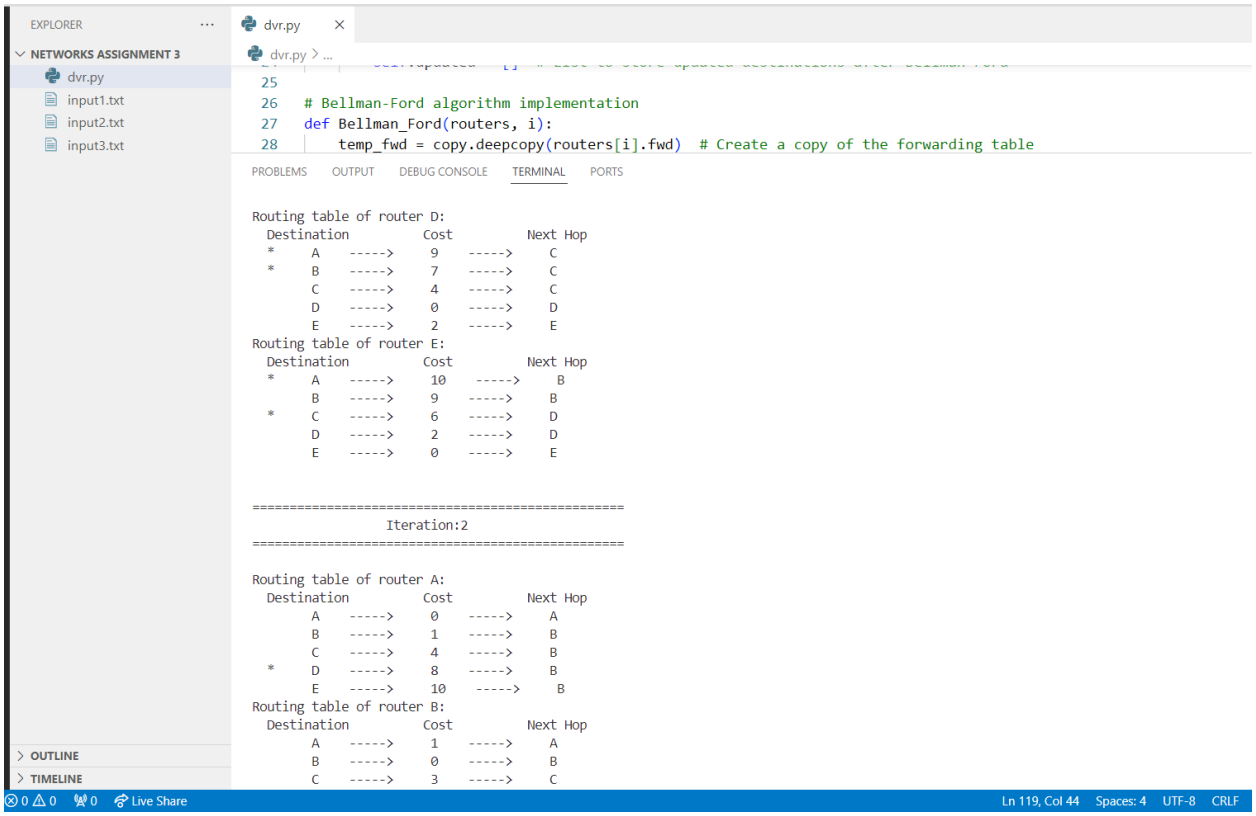


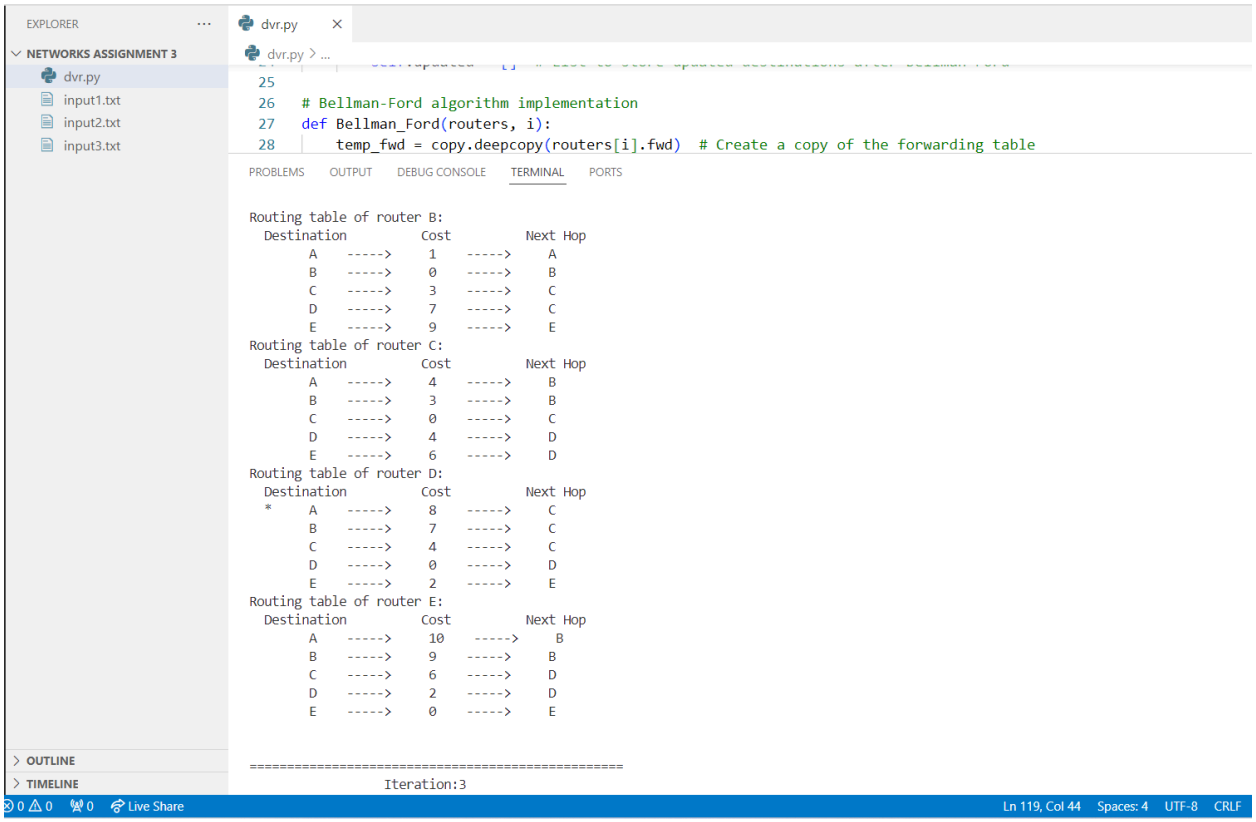


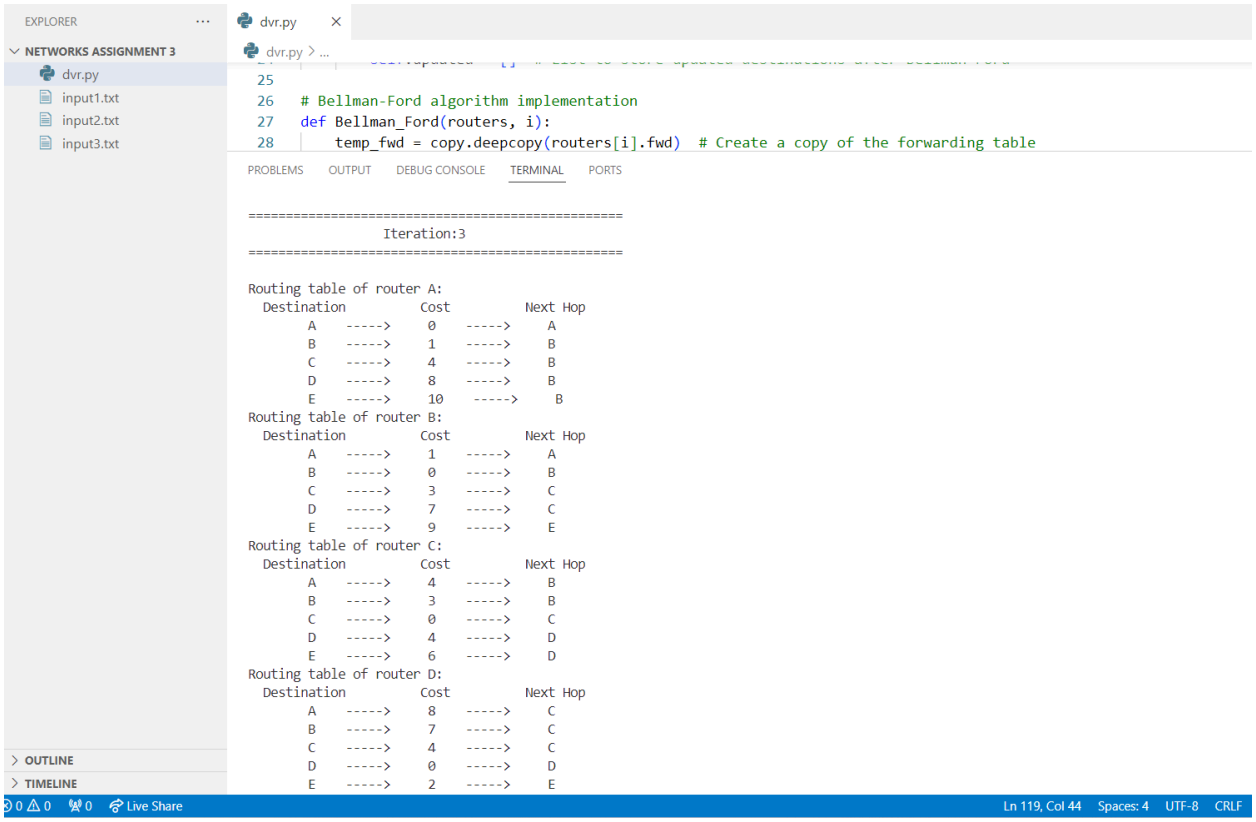
Output:

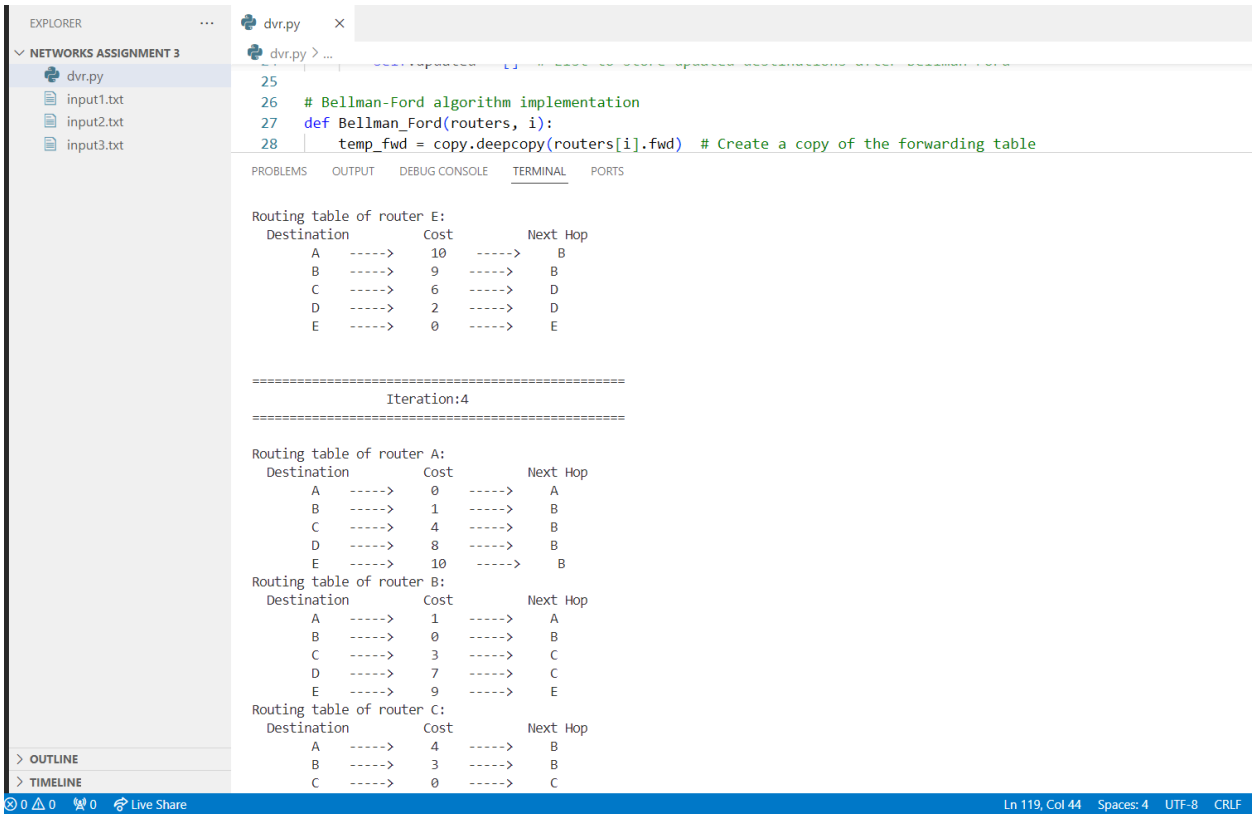

EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\mdsho\Desktop\s6\Computer Networks\Networks Assignment 3> python dvr.py input3.txt
Initial Routing Tables:
Routing table of router A:
Destination Cost
A -----> 0
B -----> 1
C -----> 5
D -----> inf
E -----> inf
Routing table of router B:
Destination Cost
A -----> 1
B -----> 0
C -----> 3
D -----> inf
E -----> 9
Routing table of router C:
Destination Cost
A -----> 5
B -----> 3
C -----> 0
D -----> 4
E -----> inf
Routing table of router D:
Destination Cost
A -----> inf
B -----> inf
C -----> 4
D -----> 0
E -----> 2
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF

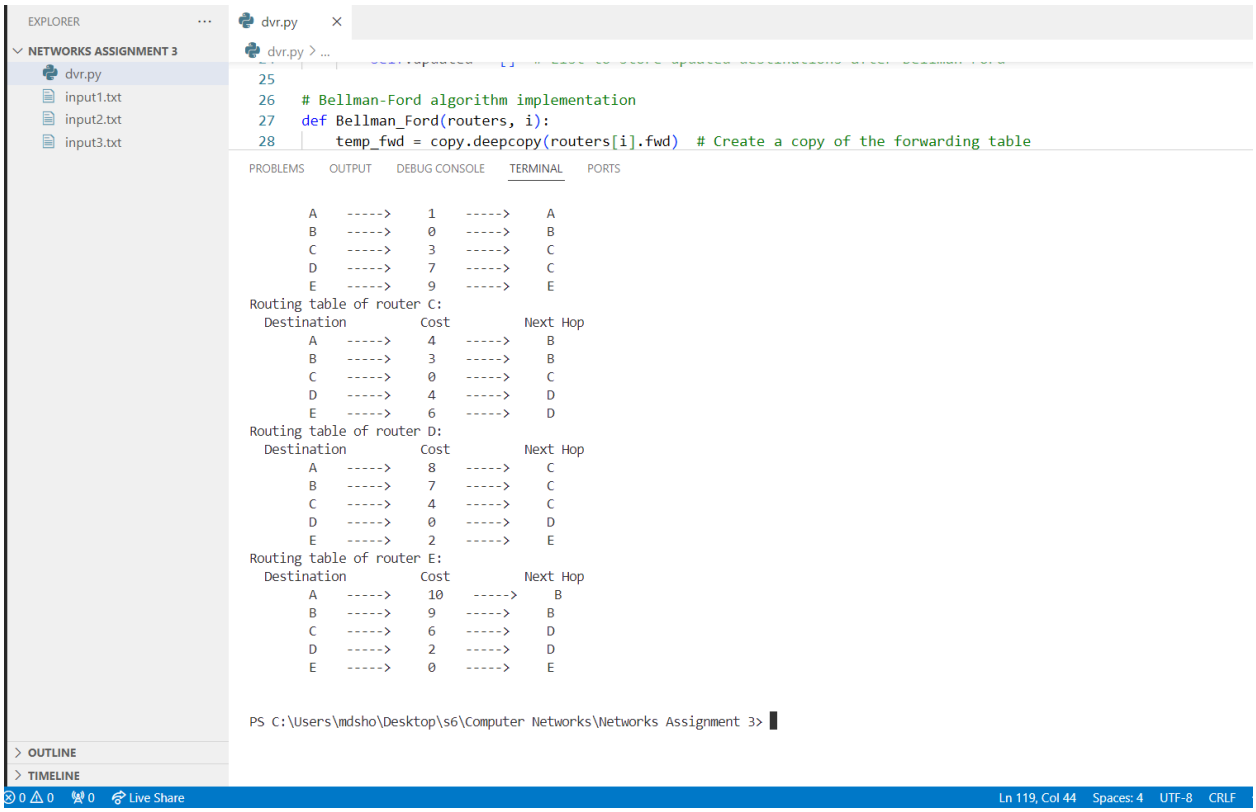

EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Routing table of router E:
Destination Cost
A -----> inf
B -----> 9
C -----> inf
D -----> 2
E -----> 0
Iteration:1
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 1 -----> B
* C -----> 4 -----> B
* D -----> 9 -----> C
* E -----> 10 -----> B
Routing table of router B:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
* D -----> 7 -----> C
E -----> 9 -----> E
Routing table of router C:
Destination Cost Next Hop
* A -----> 4 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
D -----> 4 -----> D
* E -----> 6 -----> D
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Routing table of router D:
Destination Cost Next Hop
* A -----> 9 -----> C
* B -----> 7 -----> C
C -----> 4 -----> C
D -----> 0 -----> D
E -----> 2 -----> E
Routing table of router E:
Destination Cost Next Hop
* A -----> 10 -----> B
B -----> 9 -----> B
* C -----> 6 -----> D
D -----> 2 -----> D
E -----> 0 -----> E
==========
Iteration:2
==========
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 1 -----> B
C -----> 4 -----> B
* D -----> 8 -----> B
E -----> 10 -----> B
Routing table of router B:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Routing table of router B:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
D -----> 7 -----> C
E -----> 9 -----> E
Routing table of router C:
Destination Cost Next Hop
A -----> 4 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
D -----> 4 -----> D
E -----> 6 -----> D
Routing table of router D:
Destination Cost Next Hop
* A -----> 8 -----> C
B -----> 7 -----> C
C -----> 4 -----> C
D -----> 0 -----> D
E -----> 2 -----> E
Routing table of router E:
Destination Cost Next Hop
A -----> 10 -----> B
B -----> 9 -----> B
C -----> 6 -----> D
D -----> 2 -----> D
E -----> 0 -----> E
==========
Iteration:3
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
==========
Iteration:3
==========
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 1 -----> B
C -----> 4 -----> B
D -----> 8 -----> B
E -----> 10 -----> B
Routing table of router B:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
D -----> 7 -----> C
E -----> 9 -----> E
Routing table of router C:
Destination Cost Next Hop
A -----> 4 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
D -----> 4 -----> D
E -----> 6 -----> D
Routing table of router D:
Destination Cost Next Hop
A -----> 8 -----> C
B -----> 7 -----> C
C -----> 4 -----> C
D -----> 0 -----> D
E -----> 2 -----> E
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF


EXPLORER
NETWORKS ASSIGNMENT 3
dvr.py
input1.txt
input2.txt
input3.txt
OUTLINE
TIMELINE
dvr.py
dvr.py > ...
25
26 # Bellman-Ford algorithm implementation
27 def Bellman_Ford(routers, i):
28 temp_fwd = copy.deepcopy(routers[i].fwd) # Create a copy of the forwarding table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Routing table of router E:
Destination Cost Next Hop
A -----> 10 -----> B
B -----> 9 -----> B
C -----> 6 -----> D
D -----> 2 -----> D
E -----> 0 -----> E
==========
Iteration:4
==========
Routing table of router A:
Destination Cost Next Hop
A -----> 0 -----> A
B -----> 1 -----> B
C -----> 4 -----> B
D -----> 8 -----> B
E -----> 10 -----> B
Routing table of router B:
Destination Cost Next Hop
A -----> 1 -----> A
B -----> 0 -----> B
C -----> 3 -----> C
D -----> 7 -----> C
E -----> 9 -----> E
Routing table of router C:
Destination Cost Next Hop
A -----> 4 -----> B
B -----> 3 -----> B
C -----> 0 -----> C
0 0 0 Live Share
Ln 119, Col 44 Spaces: 4 UTF-8 CRLF



CODE EXPLANATION

1. Utility Functions:
    - get_node_id(name): This function converts a node name (e.g., 'A', 'B', 'C') to its corresponding node ID (0, 1, 2, etc.).
    - get_node_name(num): This function converts a node ID to its corresponding node name.
2. Router Class
    - The Router class represents a router in the network. Each router has a name, an ID, a forwarding table (fwd), a list of next-hop destinations (next_hop), a list of neighboring routers (neighbors), and a list of updated destinations (updated).
    - The forwarding table is initialized with infinity cost for all destinations, except for the cost to the router's own node, which is set to -1.
3. Bellman-Ford Algorithm:
    - The Bellman_Ford function implements the Bellman-Ford algorithm, which is used to update the forwarding tables of each

router based on the received forwarding tables from neighboring routers.
    - The function creates a temporary copy of the current router's forwarding table, updates it based on the received forwarding table, and then updates the router's forwarding table accordingly.
    - The function also keeps track of the updated destinations, which will be used to print the routing tables after each iteration.
4. Propagation Function:
    - The Propagate function is responsible for propagating the current router's forwarding table to its neighboring routers.
    - It first sends the current router's forwarding table to all its neighbors by putting it in their respective queues.
    - Then, it waits until the queue for the current router is full, indicating that all the neighboring routers have received the forwarding table.
    - After that, it calls the Bellman_Ford function to update the current router's forwarding table based on the received forwarding tables.
    - Finally, it clears the queue for the current router.
5. Main Function:
    - The main function reads the input file, which contains the number of nodes, the names of the nodes, and the edge information (from-node, to-node, and weight).
    - It creates a Router object for each node and initializes their forwarding tables and neighbor information based on the input data.
    - It then creates a queue for each router, which will be used to propagate the forwarding tables.
    - The main function prints the initial routing tables for all routers.
    - It then runs the Bellman-Ford algorithm for 4 iterations, creating a separate thread for each router to propagate its forwarding table to its neighbors.
    - After each iteration, it prints the updated routing tables, indicating the destinations with updated cost and next-hop information.