**CSC358 A3 Report**

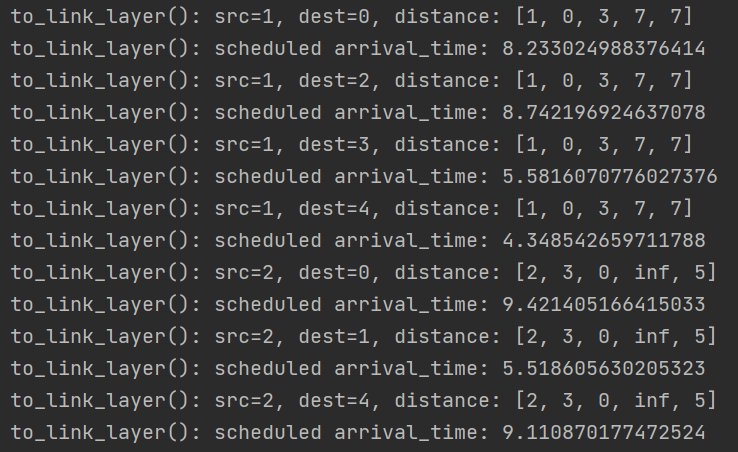
1. **Team members**

Wentao Zhou 1005308490

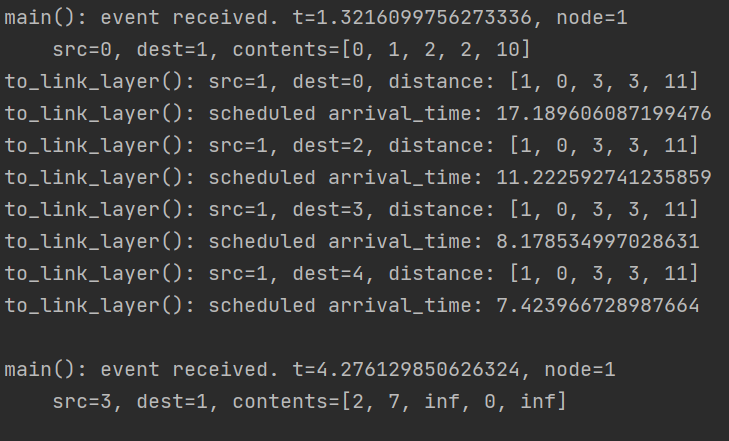
Youan Cong 1005251184

All the requirements were completed.

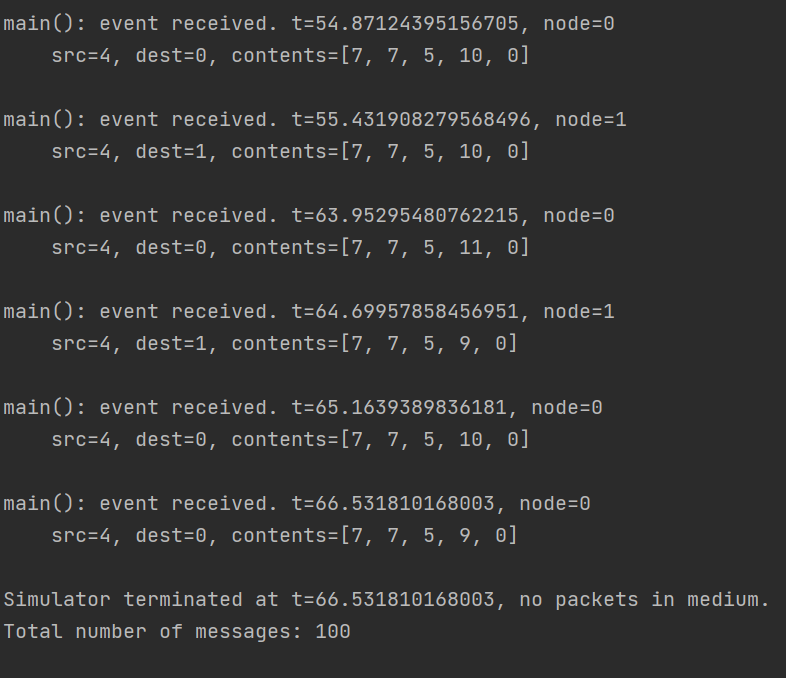
1. **Test 1: No link change NUM\_NODE: 5 Seed: 2**



At first, every node will send its DV to all its neighbors. If the link cost is infinite, that means the two nodes are not connected and will not send their DV to each other.

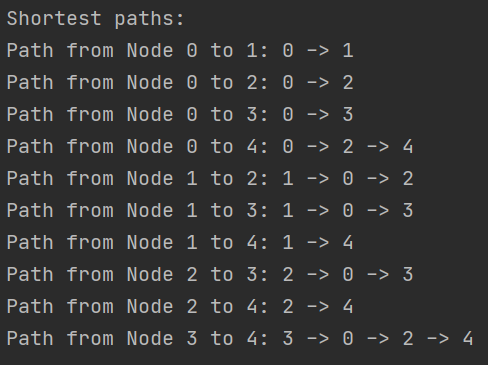


Once a DV was received, the node will compare it to its dist\_table and check whether the upcoming DV is a new one. If the upcoming DV is not the same as the one in dist\_table, the node will update its own DV with the new one and send its updated DV to all its neighbors. Otherwise, nothing will happen.



This process will continue until all the nodes have no updates with the new DV and no longer sending their DV to their neighbors. They just store the new DV to their dist\_table.





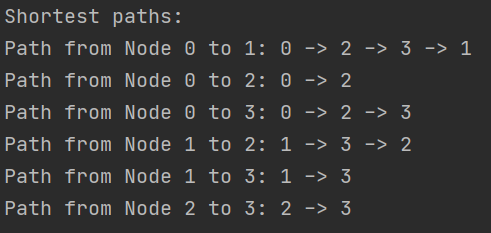
The results are the same as the manual calculations.

1. **Test 2: Link change between node 1 and 2 with cost 1 NUM\_NODE: 4 Seed: 6**

At first, Node 0 and 1 are not connected.

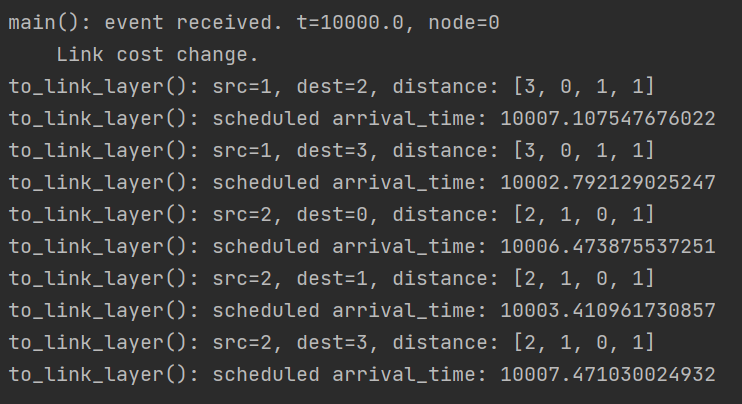
The number of messages sent when there is no link change were 46, and here is the expected result of the node 2 without link change:



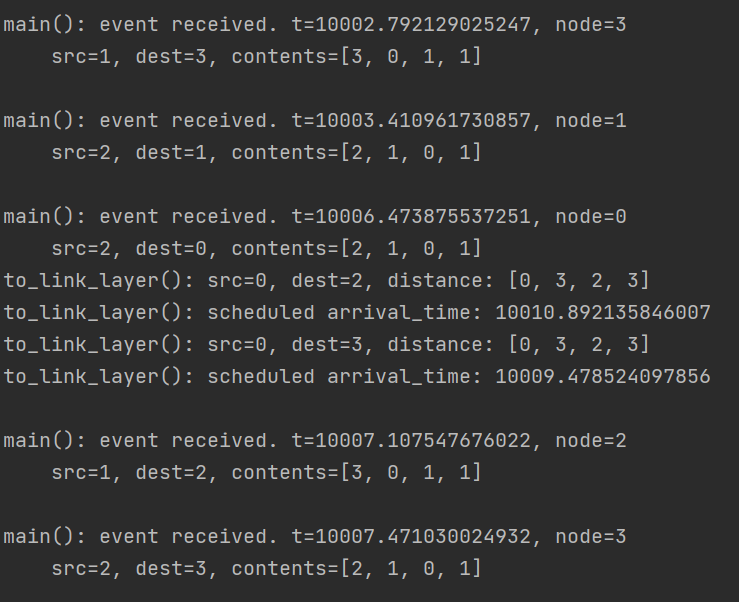


Now, let’s look at the case with link change.

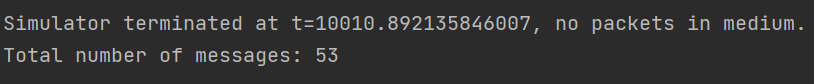
Everything is the same as the one above before link cost change happens.



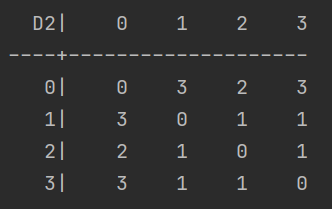
Now, link change between 1 and 2 will cause them to send their new DV to all its neighbors.

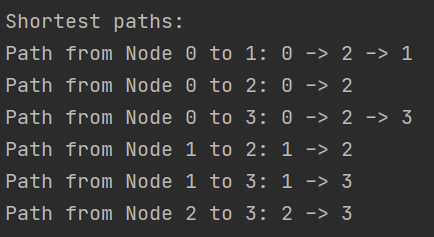


Majority nodes have no updates on their DV since they are not taking the link change path at the first place. Only those who were taking that path will update the DV and send to their neighbors.



As we can see, only a few more messages were sent, and the “count to infinity” problem didn’t happen, which perfectly fit the “good news travels fast” principle, since only those affected nodes need to update their DV to the new minimum once or few times.



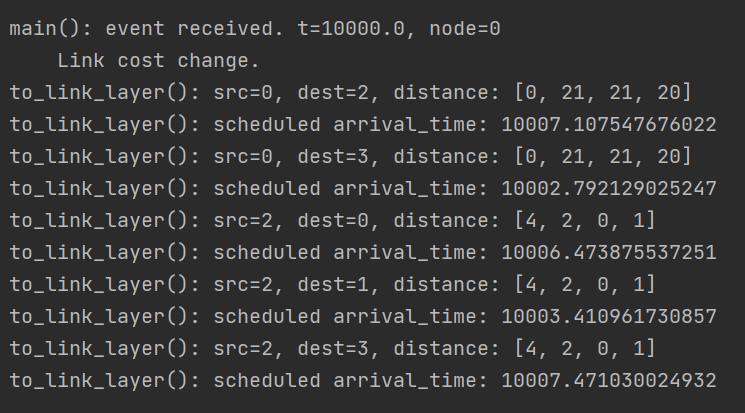


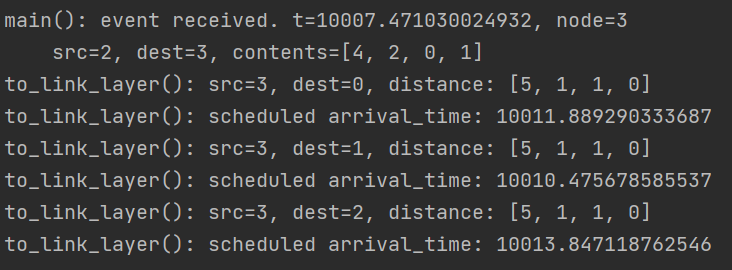
New tables and paths after link change. Same as manual calculation.

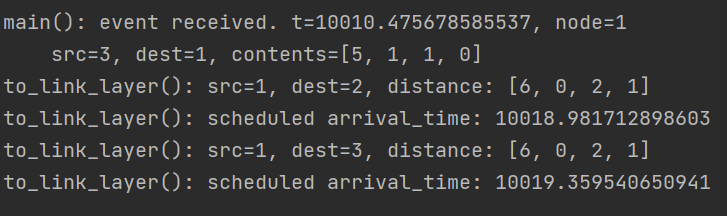
1. **Test 3: NUM\_NODE: 4 Seed: 6 Link change between node 0 and 2 with cost 60**

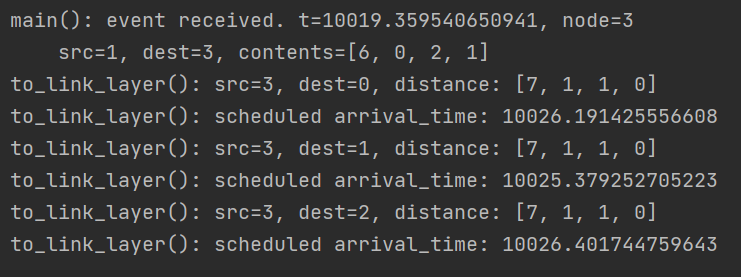
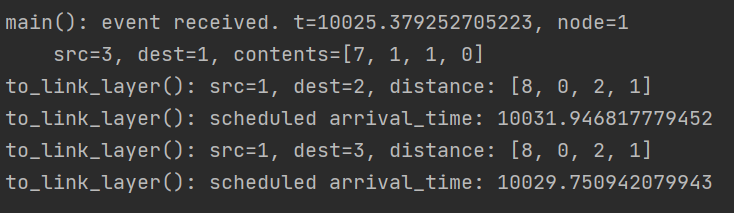
At first, Node 0 and 1 are not connected.

The number of messages sent when there is no link change were 46, and the expected result can be found above.

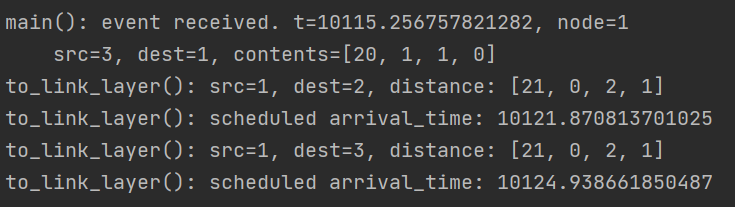


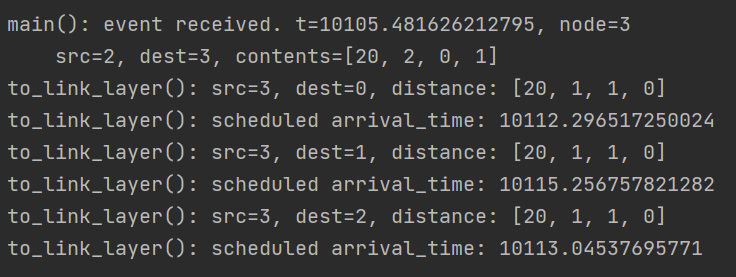


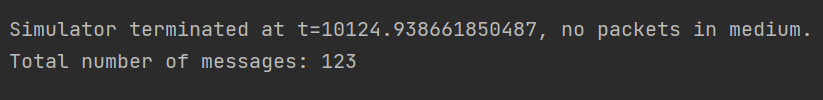


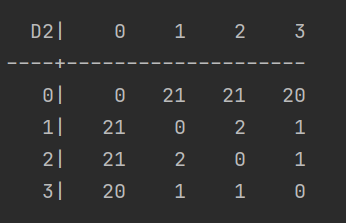
As the screen shots show, the shortest path is slowly increasing one by one. The “count to infinity” problem happens.

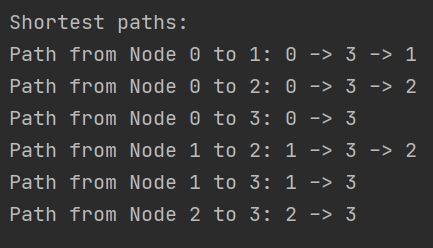






It stops until it reaches the new shortest path. As you can see, there are tons of messages sent in between, which shows how bad news travels slow.





New tables and paths after link change. Same as manual calculation.

1. **Discussion:**

Experiments:

If we keep the same seed, disable the link cost change events, the messages sent for different number of nodes are:

3 nodes 20 messages,

4 nodes 57 messages,

5 nodes 100 messages,

6 nodes 225 messages, which is close to O(n^3)

**However, we cannot conclude a general complexity for this distance vector routing algorithm we implemented. There are too many factors that affect the number of messages sent and the speed of the convergency.**

Suppose there are n nodes, every time node A receives a message about an update from other nodes, it checks the table and recalculates it. (The recalculation process takes up to O(n^2) runtime in general, but we are not considering the recalculation process because the handout only asks for the complexity based on the messages sent). If the DV for node A is changed, it must notify all its neighbours about this change. The worst case is when all the nodes are connected (dense graph), and a node would have n-1 neighbours, that is, n-1 messages sent for each node update, so the notification process may take place O(E) runtime, whereas, on a sparse graph, a node may only have one or even zero neighbours, so now the runtime may as low as O(1). **We can only say that the message complexity for each node is the degree of the node.**

Secondly, we cannot only describe the general complexity for a node update but also cannot determine how many iterations need to achieve the convergency. There may have the “counting to infinity” problem if there is a link cost change. **The speed of the convergency varies for different graphs.**

Let’s see an example, suppose all nodes need to tell everybody else about its DV at the start, then the first round of everyone sending messages to one other takes place O(n^2) times. Now, all nodes should have the DV of other nodes in their dist\_table and the optimal solution based on the table. Assume all the nodes need an update to their own DV after receiving other node’s DV during this process, a node receives n-1 messages, each one leads to n-1 messages sent, so O(n^2) for each node and O(n^3) in total, and so on. The overall complexity of this is at least O(n^3) so far. However, if all nodes are lineup, the complexity could be O(n^2). In the best case, all nodes already have the best DV from the beginning, or even not connected so that no update is required at all. The algorithm only takes O(n) or O(1) in this case.

**As you can see, we cannot generally conclude how many rounds/iterations are needed to achieve convergency, and the problem of link cost change will seriously interfere with it. Maybe before everything close to being converged, there is always a new link cost change happen. The potential runtime may increase or decrease repeatedly. There is no universal complexity, different graphs and events will affect it differently.**