ECE361 Lab1 Report

Ian Hu 1006939244

Santiago Ibagon 1006831591

Ethan Sovde 1006747040

Shawn Zhai 1006979389

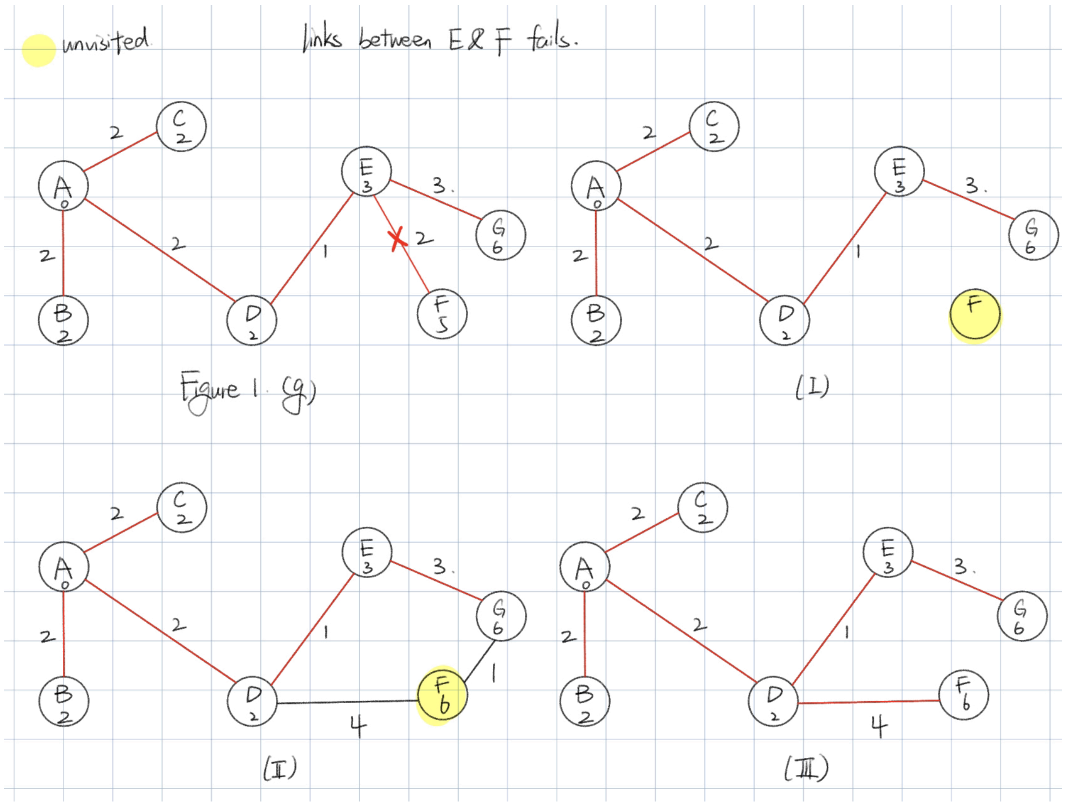
5.1 Dijkstra Algorithm Questions

In Figure 2, why the algorithm did not select the link from router F to G in step f?
Because in the Dijkstra algorithm, if two links have the same weight, the router with the smallest ID (in this case, E is selected) is selected.

In the Dijkstra algorithm, when considering how to reach an unvisited node from all visited nodes that link to the unvisited node, the one with the shortest distance to the source is processed first due to the nature of priority queue.

At step (f), the algorithm sees two paths reaching G, through E and F. It first considers the patch through E as it has the shortest distance to A (distance[A, E] = 3, while distance[A, F] = 5), and calculates the shortest distance from A to G through E has a total cost of 6 (A→D→E→G). This path has the shortest distance to G so far, so the algorithm sets this path to be the shortest. It then considers the path through F, and calculates the shortest distance from A to G through F has a total cost of 6 as well (A→D→E→F→G). It has the same cost as the existing shortest path through E. Since the path through E has already been chosen and doesn't have a higher cost, the algorithm does not need to consider the link from F to G.

Taking Figure 2(g) into account, what happens if suddenly the link between routers E and F fails? Please depict the steps involved from the time of failure until all routers can be reached from router A with shortest path condition? Use the network in Figure 1 as your reference for the network topology prior to the failure.



When the link between E and F fails, F becomes an unvisited node (graph I). Then we reconsider the other links that connect the visited nodes and F, which is the link from D to F and from G to F (graph II). The algorithm finds that the shortest path from A to F through D has a total cost of 6 (A→D→F), and the shortest path from A to F through G has a total cost of 7 (A→D→E→G→F). It then chooses the path with the link from D to F as the shortest path to F (graph III).

It is known that Dijkstra algorithm cannot process network topologies with negative weights. Can you elaborate on why using negative weights can be an issue?

Because Dijkstra's algorithm's logic is based on the assumption that once a node's shortest path is determined, it will not be changed. For each node, it selects the node with the smallest known distance. However, if there is a negative weight, adding a negative-weight edge could make it shorter than previously calculated paths, and this does not match the assumption.

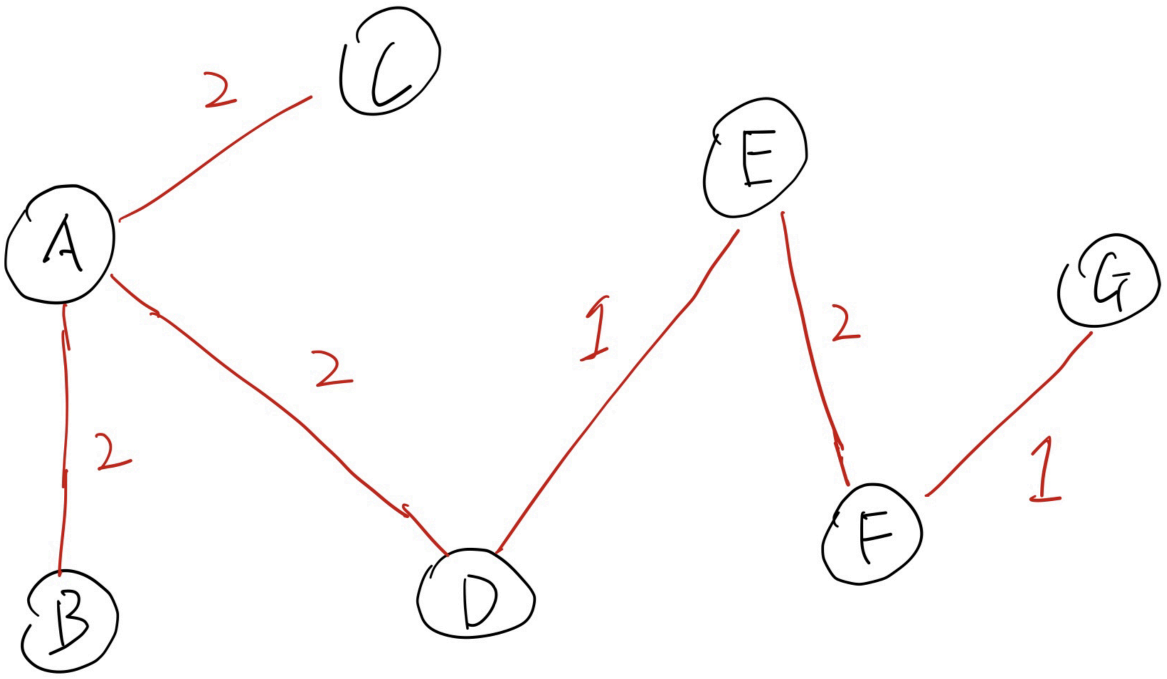
Also, there will be a risk of infinite loops with negative cycles. If a network where the sum of all edge weights is negative, links can become infinitely short by looping through the cycle repeatedly. Dijkstra's algorithm is not able to detect the cycles.

6.1 Bellman-Ford Algorithm

Do you think it is necessary for the Bellman-Ford algorithm to run N - 1 iterations?
It is not strictly necessary for all N - 1 iterations to be run if the algorithm finds a converging solution before that point. If this occurs, the algorithm has found all shortest paths and it can be terminated earlier to conserve runtime. This can be achieved by exiting if the current iteration made no changes to the graph, since if no changes were made this iteration, the next iteration will be identical and also make no changes.

Using the topology provided in Figure 1, after how many iterations does the network converge when the Bellman-Ford algorithm is used?
After the first iteration, the network converges and all of the path lengths are the same as in the final iteration.

Are there any other shortest path trees that can satisfy the shortest path condition from router A to all other routers/destinations in the network?
Yes, there is another tree with different paths but the same path lengths.


C
2
E
A
G
1
2
2
2
1
F
B
D