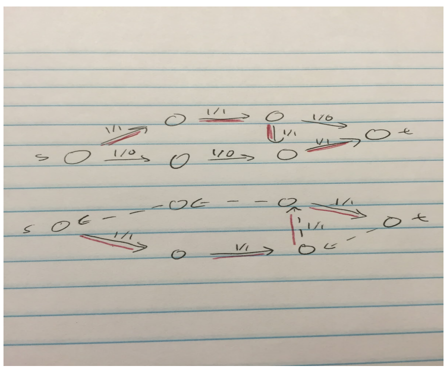
James Sweetman (jts2939)
Jung Yoon (jey283)
Unique #51835

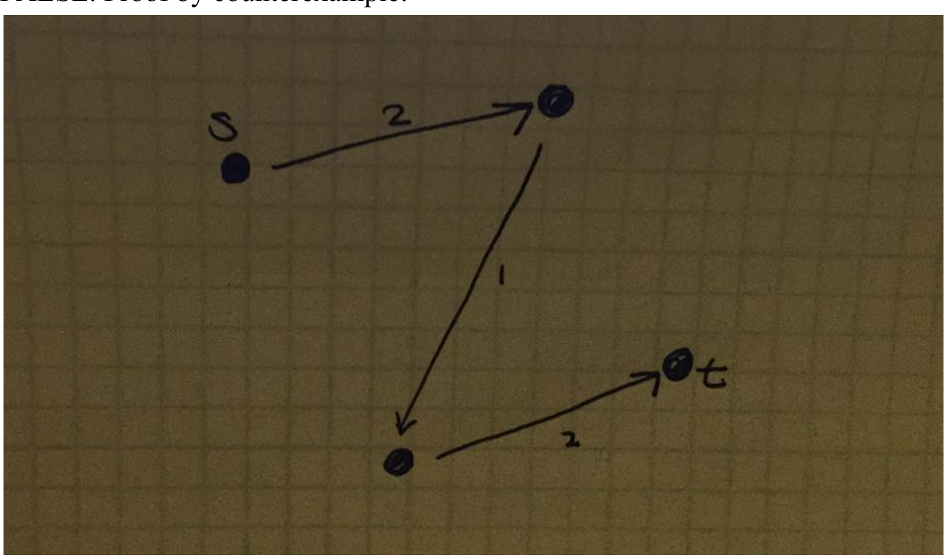
HW #7

1.

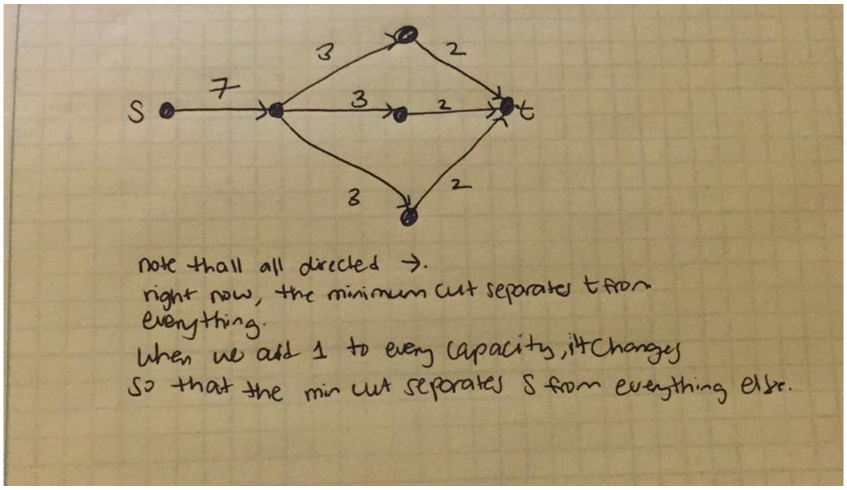
a. Define a set of network flow graphs with with n x n nodes where columns are connected by a rightward pointing edge and rows connected by a downward pointing edge. Let source s connect to all the leftmost edges in the graph and all the rightmost edges connect to sink t. With all edges having a capacity of 1, the max flow will be n. FFA will return n because it is able to traverse the backward edges in case it chooses augments a bad s-t path. The modified version could potentially have a flow of 1 if it moves downward towards t. Alpha will work when alpha <= 1/n, but not work when alpha > 1/n. This means if we have a network flow with a larger n, the alpha will not hold.



b. FALSE. Proof by counterexample:



c. FALSE. Proof by counterexample:



2.

a. Algorithm:
Our algorithm is a variation of the Ford-Fulkerson algorithm in combination with Breadth First Search.

More specifically, we will create a network graph / residual graph from the input that we're given. First, we make a bipartite graph such that {drivers, buses}, such that the edges are defined by the lists L1,...Ln,Ln+1, and such that each driver has a directed edge from the source (each with capacitance of 1) and each bus has a directed edge to the sink (each with capacitance of 1).

Then, we will use BFS to find the shortest path from the source to the sink in a network graph. If there is no path then there is no new schedule available, otherwise we push flow and note the resulting graph to get the new schedule.

Proof:
We already know that Ford-Fulkerson and BFS works. It's trivial to prove these algorithms.

Runtime:
Creating the graph is O(n*n) since there are n drivers and n buses. BFS will take O(n+n). So runtime will be O(n*n) .

b.