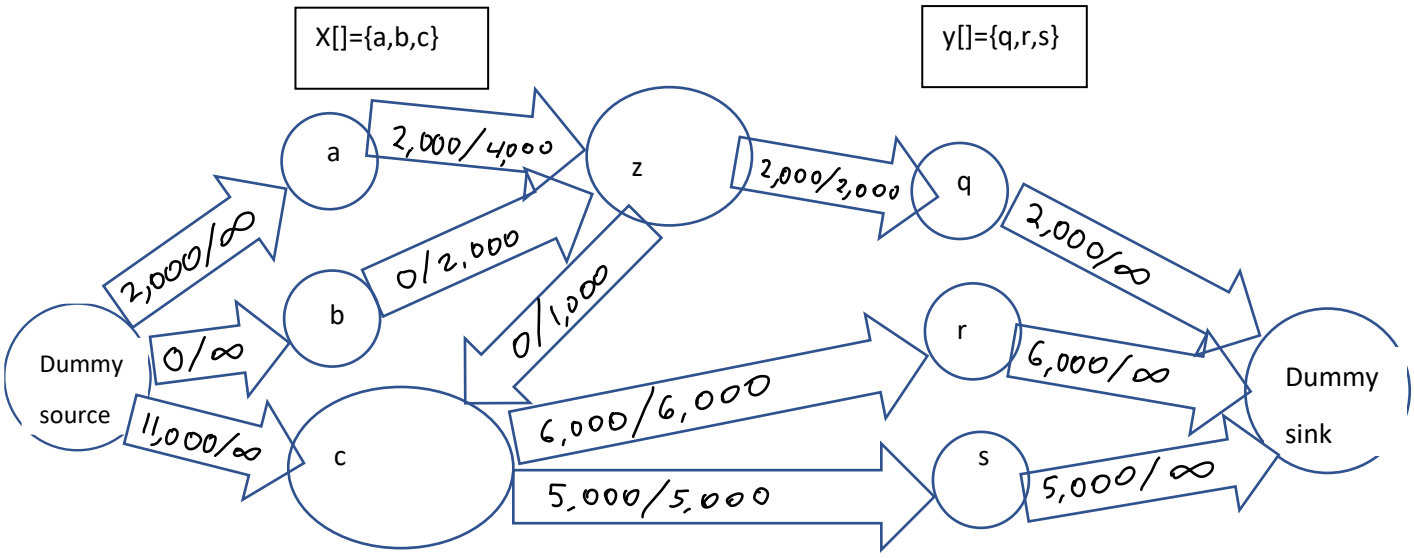
The goal of this system is to move bushels along a series of roads with capacities of bushels per hour, thus this does seem to somewhat resemble a network flow problem. The difference here is twofold. First, there are multiple sources – each with 10,000 extra bushels, and multiple possible destinations, and second, that we don't care about shoving as much through the network as possible, but rather about how long it will take to push through a specified number of bushels.

The second difference is trivial. The network flow provides a number that can pass through per hour, so a simple ration of: 10,000/max flow, will provide the amount of time it will take to send 10,000. The first difference, the multiple source and destination issue is slightly trickier, but with a simple tweak, it can be changed to a network flow. We are told that each granary in X has 10,000 extra bushels that we can choose to transport, so we can create a dummy source vertex with an edge to each granary in X with a capacity of Integer.MAX_VALUE per hour. Thus, the flow can push anything between 0 bushels and Integer.MAX_VALUE bushels to each starting granary, representing the amount of extra that will subsequently be transported from that granary. We can do the same on the backend, creating a dummy sink granary with an edge from each granary in Y to the dummy sink with capacities of Integer.MAX_VALUE bushels per hour, signifying the possibility of transporting anywhere from 0 to Integer.MAX_VALUE bushels to that granary destination in Y. The fact that the bushels are transported from dummy source to the starting granaries is irrelevant, because the amount of f^{in} of each starting granary will simply parallel the amount of overflow that can be transported from it, and the amount of f^{out} of the destination granaries to the dummy sink can just represent the amount of bushels being transported to the destination, as seen from the flow conservation principle of network flow.

To restate what was said above a bit more formally, creating the dummy source with edges to each source in X with capacities of Integer.MAX_VALUE bushels will generate a flow from dummy source to each node in x, that, by the law of conservation of flow, will parallel the number of bushels that can flow out of that node and be transported, from there, to any of the destinations. Creating a dummy sink with edges into it from each destination node in Y, will just result in the same amount of flow possible into each destination node flowing from there to the dummy sink by the same conservation law. Thus, my model will respect the conservation of flow. Given that a vanilla Ford Fulkerson is being run to determine flow, edge capacities will still be respected.

See the diagram below for illustration.



The code for FlowEdge, FlowNetwork and FordFulkerson is based on code from Robert Sedgewick.

The original code can be found here: https://algs4.cs.princeton.edu/64maxflow/FordFulkerson.java

here: https://algs4.cs.princeton.edu/64maxflow/FlowEdge.java.html

and here: https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/FlowNetwork.java

I edited these classes, adapting them to suit this problem, and tweaking the code to conform to what I need, but they are based on the classes found at the above links