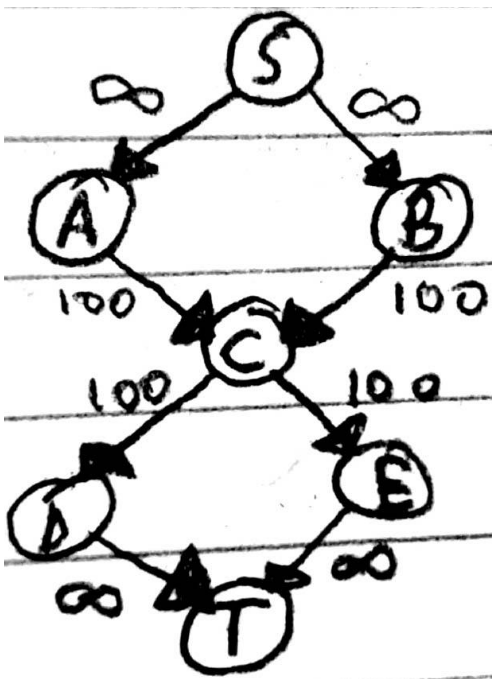
OverfullGranaries Writeup

I modeled the problem as a network flow problem where my goal was to maximize the flow rate between the overfull granaries and the underfull granaries because apparently Yosef wasn't smart enough to put the grain in the right place the first time. I modeled the granaries as vertices, roads as edges, and road capacity as, well, capacity. To get the amount of time required to push grain between the granaries, I found the maximum capacity and divided the volume of grain needed to be moved (10,000) by the capacity.

The main difference between my problem and a standard network flow problem is that there are many source nodes (overfull granaries) and many sink nodes (underfull granaries). However, since it doesn't matter how much flow goes in and out each of the source and sink nodes, the solution was simple. I created a dummy source node that had edges going to each overfull granary and a dummy sink node that each underfull granary had edges going to. Because it did not matter how much grain left each individual overfull granary or entered each individual underfull granary, I gave these nodes infinite capacity. Maximizing the flow from the dummy source to the dummy sink maximizes the flow from the overfull granaries to the underfull granaries.



In the picture on the left, nodes A and B are overfull granaries and D and E are underfull granaries, while C is an intermediate granary. The edges between the granaries have the capacities they were given (in this example, 100 per road). There is the dummy node S, which has edges of infinite capacity to overfull granaries A and B. There is the dummy node T, where edges of infinite capacity enter it from underfull granaries D and E. Using network flow to find the maximum capacity from S to T would find a capacity of 200 bushels per hour, for a duration of 50 hours.

Formal reduction to Network Flow:

1. Create a graph G where granaries are vertices and roads between them are edges with the given capacities
2. Add new vertices *s* and *t* to G
3. Add an edge to G with infinite capacity from *s* to each vertex in `X`
4. Add an edge to G with infinite capacity from every vertex in `Y` to *t*
5. Solve the maximum flow problem on G using Ford-Fulkerson
6. Return 10,000 / max flow

The problem is asking us to minimize the time taken to move grain from the overfull to the underfull granaries. Time is measured by the equation $time = \frac{volume}{flow}$. Since $volume$ is a constant 10,000 bushels and *flow* is a constant number of bushels per hour once it is selected, maximizing the selection of *flow* minimizes the time, no calculus required. I can use network flow to maximize *flow*. Adding dummy nodes doesn't change the resultant flow, because the problem does not care how much grain flows through each individual granary, only between `X` and `Y`. Since (by construction) every path contains $x \in$ `X` and $y \in$ `Y`, network flow will find that flow. While some edges have infinite capacity, this also doesn't violate the requirements of network flow, because only the edges connecting *s* and $x \in$ `X` and $y \in$ `Y` with *t* have infinite capacity, and since no element can be in both `X` and `Y`, all paths are guaranteed to have a finite capacity. So I can use maximum *flow* from network flow to minimize *time*. QED.

I borrowed my Network Flow code from GeeksForGeeks, found at the following link: https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/. I modified it so heavily (it used matrices instead of adjacency lists, and didn't find the min cut) that it feels like I shouldn't have to have to credit it, but if it's enough to trigger TurnItIn, I guess I have to cite it. Apparently it implemented Edmonds-Karp.