Problem Set 6 - Network Flow

Problem D

Delivery Bears wasn't too bad either, the fact that 'no bear could rest', meant that we needed a weight that would allow us to flow every single bear through the network. In some instances, having a weight of 1 wouldn't allow us to flow every bear through the network, so the weight needed to be lowered. But having all bears flow through the network wasn't enough to solve the problem, as we wanted the max weight we can carry through the city. In some cases, all bears can flow through with a weight of 1, but also all bears can flow through with a weight of 2 - and thus, this materialised into a pretty certain goal, which was to find the maximum weight such that all bears could flow through the network.

With that, came part two of trying to solve this problem, which was how to actually find that. My initial thought about this was to run the max flow algorithm twice, thinking there was some relationship between the max flow that with weight = 1 and the max weight that we could flow through with. But it seems there was no mathematical relationship (?), or at least none that I could think of. My suspicion and hypothesis about this was that changing the weight would change the way bears can walk through the graph - i.e. they would take different paths through the city, and thus, this meant there was no way to relate the flow from the weight = 1 graph to the flow with our ideal weight graph.

I ultimately came to realise that this is a monotonically increasing function - 'at weight x, can all bears flow through?' and thus, binary search would work in this case, bringing me to my final approach. Implementing binary search was the actual more difficult part though, since we were working with doubles. I had a feeling binary search wouldn't converge since we could now micro divide intervals with doubles, and that I probably needed to use an epsilon value to check that we've reached our answer.

I also got the tip from a friend who spoke to a tutor about this, that a good way to prevent precision errors is to work in ints, and multiply everything up to an integer before dividing back down, which I did. Another (possibly) unique thing I did in my implementation was regenerating the graph with modified edge weights every run of the binary search in order to check if we could flow through. I couldn't think of a way to actually flow different units (other than 1) through a graph, so I just rebuilt the graph with different weights.