Problem Set 6 - Network Flow

Problem A

Cops and robbers actually ended up being much harder than magic potion for me to figure out. I reckon it's because when I did it, I hadn't really grasped the concept of min cut very well. Max flow is super intuitive, but min cut - and its applications to qualitative problem statements didn't really make as much sense to me.

Upon consulting my friends and hassling them about explaining min cut to me, it helped to think about min cut as a means to find the bottlenecks when supplying infinite flow through a graph.

In cops and robbers, we were trying to find what the cheapest way to barricade the robbers in were, and so, if we build a flow graph where each node represents a location, and the incoming edges are the cost to barricade that node - if we flow through that graph with an infinite source, we can find which edges (and thus nodes) fill up first, and thus, are the cheapest to block off the path. Because of this, the max flow would then be equal to the cost to block in the robber.

Jumping more into the details and away from my shotty explanation of min cut, I created a graph where each vertex represented a location on the grid. They are then connected to their next locations (UDLR), with weight equal to the cost of barricading said next location. I made sure that each vertex had a capacity equal to the cost of blocking its curr location (to prevent double counting the barricading of that path). If we can't barricade the location, the incoming edge would be infinity. All the perimeters of the grid would flow to the sink with infinity, and there is a source that flows into the initial starting location (the bank) with infinity.

When we flow through the graph, if the result is equal to infinity, that means there was a path through the grid that had no barricades on the way, and thus, the robber could not be barricaded. If it's not infinity, then the max flow = cost to barricade :)

I think the only big issue I ran into when coding up this question was at first missing the vertex capacities. I then drew a case where there were two robber paths through the same barricade node on the way to the sink, and it would double count the cost to barricade that path. We only needed to pay once to barricade that node.