Module 6 - Graphs: Prim's and Dijkstra's

1. Prove or give a counter-example for the following claim: If an undirected, connected graph G has all unique integer edge weights, then the Minimum-Spanning Tree of G must be unique as well.

2. *This problem was in the slides but we didn't get to it in lecture.* Suppose Floryan is trying to schedule a flight from city A to city B and wants an itinerary that is as efficient as possible. However, instead of a traditional flight optimization (e.g., getting to destination in the least amount of total travel time), Floryan only cares about minimizes his layover time in airports. Floryan doesn't mind being on the plane for a long time, or having multiple legs to his trip. He dispises sitting in airports and not making any progress. Given a start city A, an end city B, and a list of flights (each flight contains the start city, end city, departure date/time, and arrival date/time), describe an algorithm that finds the itinerary with he minimum layover time possible.

   HINT: You don't need to develop a new algorithm here. One of the ones from class will work. However, you might need to alter how you structure the problem as a graph so that the algorithm works as intended.

3. This problem is about robots that need to reach a particular destination. Suppose that you have an area represented by a graph $G = (V, E)$ and two robots with starting nodes $s_1, s_2 \in V$. Each robot also has a destination node $d_1, d_2 \in V$. Your task is to design a schedule of movements along edges in G that move both robots to their respective destination nodes. You have the following constraints:

   1. You must design a schedule for the robots. A schedule is a list of steps, where each step is an instruction for a single robot to move along a single edge.
   2. If the two robots ever get close, then they will interfere with one another (robot social distancing, you know!?). Thus, you must design a schedule so that the robots, at no point in time, exist on the same or adjacent nodes.
   3. You can assume that s_1 and s_2 are not the same or adjacent, and that the same is true for d_1 and d_2.

   Design an algorithm that produces an optimal schedule for the two robots. Make sure to address each of the following:

   - Describe your algorithm (*HINT: Try changing the graph into a different one to remove the constraints of the Robots not being able to be on the same or adjacent nodes*)
   - What is the worst-case runtime of your algorithm? Specifically, how big is the new graph you constructed as a function of the size of the original graph?
   - How would the runtime change as the number of robots grows? Does the algorithm get faster or slower?