Name: Jon Abrahamson
ID: 107084898

CSCI 3104, Algorithms | **Profs. Grochow & Layer**
Explain-It-Back 8 | **Spring 2019, CU-Boulder**

You are collaborating with a geology team that is using a rover to explore lava fields. They have mapped out the surface of the lava field as a grid where each edge is annotated with the likelihood that the robot can successfully navigate the corresponding terrain. This likelihood integrates physical properties such as surface temperature and relief. Their current algorithm finds a path by considering all of the edges at its current position, then all of the edges that are one step away, two steps away, and so on until they reach the desired destination. Unfortunately, this process takes so much time to complete that the physical properties of the edges change before a route can be calculated rendering the path useless. You have the insight that the robot does not need the best path, just one that can be calculated quickly and has a reasonable likelihood of success. Help your team understand the issues with the current solution, and how a simple algorithm change could help.

Name: Jon Abrahamson
ID: 107084898

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 8 **Spring 2019, CU-Boulder**

We can optimize the current algorithm using Dijkstras algorithm. The idea is to traverse all vertices of graph using Breadth First Search and use a Min Heap to store the vertices not yet included in a Shortest Path Tree. This algorithm would optimize the time it takes to find the shortest most optimal path based on the weighted values assigned. The weights of the values are given to the edges based on the probability of success. This algorithm would run through every single node and thus run in Theta(E + VLogV). The issue is that every single path must be analyzed in order to pick the most optimal path even though there exists many paths that are suitable. This extra time could cause the lava fields to change and thus render and optimal paths no longer suitable. However, if we do not necessarily need the shortest most optimal path, rather a path that will likely succeed that can be quickly calculated, we can instead assign each path between nodes a value of either 0 or 1. It is up to the discretion of the experts, to which I am not, to determine what original path value is deemed to have a high enough probable success rate to be taken. If the path is deemed worth being taken, than it can be assigned a value of 1. Paths not suitable to be taken are assigned a value of 0. From here we can begin looking through the available paths from the robots current position. If the path has a value of 0, we move on to the next. As soon as a path with a value of 1 is found, the robot will take that path. This will ensure that the robot does not need to analyze every single available path, but just find one that works. So now instead of the algorithm taking Theta(E + VLogV) to run every time, now the absolute worst case run time will be O(E + VLogV). This will only be the case if every path except for the last path analyzed has a value of 0. This is very unlikely and the majority of the time the algorithm will run in a far quicker time than O(E + VLogV). We now have a algorithm that, while not necessarily the best path, but one that is calculated quickly and has a reasonable likelihood of success.