Name: Keaton Whitehead
ID: 104668391

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.

Dear team,

I believe I have finally figured out the problem that we have been having with our algorithm. It's more than obvious that we have a clear destination and that we need to find a path that we can safely navigate and do so in a timely matter. As you know the timely matter is the big issue. We currently have an algorithm that is set up as a Breath-First Search algorithm which collects all the information on the vertices and edges that map to our desired location. We'll right there lies the problem! We are wasting time on continually collecting pathways between nodes to find the most optimal path, even after a safe path has been found. The algorithm won't stop until it is sure it has collected all the possible paths and returns the best option at the end of all of the algorithm. Now from my experience in this field of study, I propose that if we wish to calculate a safe route for our rover to follow we need to determine this path as quickly as possible.

Now we need to make sure the path still holds safe to the surface temperature and relief that our rover can handle. This can be an easy conditional statement implemented when we look at each possible edge of the node in our queue. Second we need to modify our search algorithm from a Breath-First Search to a Depth-First Search. What I mean by this is instead of collecting every single neighbor of the current node we are on we just have the algorithm pick the first safest path that it can take. We continue this till either we hit the desired node we were looking for, or we have run out of edges therefore hitting a dead-end and so we move back to the parent node and search the queue until another safe route is found. This will also of having the functionality of stepping all the way back up to the original node as well to grab other options if the choice was indeed that bad.

Name: Keaton Whitehead
ID: 104668391

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

Now I am not stating that this algorithm will provide us the optimal path every single time, rather, it will overall provide a pathway for our rover to travel that will get us to our destination. Best part, in a timely manner. That is because we have only searched through the nodes necessary to get us to where we needed to go instead of trying to collect all the data and waste all that time on information that is no longer relevant. After implementing this, or I suppose you can implement this to begin with, and if it still does not execute fast enough, we can also change how we store the data by storing all of the nodes and their neighbors in an adjacency list. This is simply a list where a vector is sorted for every vertex on our queue. For each of these vectors there is a pointer to a linked list of all the neighbors that node has. This will improve the search efficiencies from an order of $O(V^2)$ to O(n). V being the number of vertices we have and n being the length of the list of that given linked list. This will increase the efficiencies with our algorithm but I only mention it as a side note because I do believe just alternating our search from a Breath-First to a Depth-First algorithm will solve most of our inefficiencies.