Matthew Wen

For the third project, I decided to apply a min heap solution to find the shortest path from one node to another node. To apply this, I correct struct is needed to organize the corresponding node while also caring about the efficiency of the program when it tries to find the shortest path given a certain number of nodes.

For each node, it keeps the relative path rather than the actual address of the node. A relative path refers to if an array is created, than the distance from the start of the node in terms of how big the data type is determined by the address of it. For example, the coordinates for x and y are labeled as node 5; therefore, it will be stored in an array placed at index 5. This can minimize the data type size of storing an index rather than an address in memory. In addition, it becomes easier to reference for the end user if the user wants to start at index 0 and go to index 20 within the shortest path.

For each data type, it will have an array of all the coordinates that are connected to one node. Within each element of the array, it may be helpful to store the distance from that current node to the next node. This is known as data that can be recycled instead of resused. Therefore, if the user inputs more data where it wants to know the shortest distance between two paths, then it is easier to refer to data that is already calculated. Each data type itself would also contain the previous node. If an element is not touched or needs to be set to NULL as if there is no previous node, then it would be set to -1 since the distance between two nodes cannot be a negative number, nor the reference to a previous node can be negative.

To apply this algorithm, I will be using a min heap method to do so. A min heap have O(nlog(n)), and the min heap will be a datatype where the allocated heap size would equal to the size of the array allocated to store for the worst case scenario where one node or the starting node is connected to every single node on the map. The starting node previous would be set to -1. Each node that is connected to the starting node will calculate its distance to the current node, and each node would be appended to min heap based off its distance from the starting node. Once that operation is complete, it would take whichever node is the smallest distance from start and calculate all the nodes distance from itself, then add it to the distance of the current node from A. If the node overrides a value already set because the new calculated distance is smaller than the preset, than it changes it and reorder itself through the heap. Otherwise, it follows the same operation of calculating its distance from the start node and adding it to the min heap. Afterwards, it would get the smallest node and continue the process again. The process stops when the min heap outputs the node of the final destination.

When this progress progress, it is important to keep track of the previous node. If a new sorter path is discovered, it is important to change the previous of that node to the correct one. Then, for the node to find the path, it would follow the trail laid out from the variable previous node index inside the datatype.