Judah Goldring
LinelandNavigation write-up
Professor Leff

Essentially, I realized this is just a graph problem that needed to be solved by finding the shortest path between two verticies. Since weights aren't required and distance between adjacent verticies is all the same, bfs can be used to solve for the shortest path. All that needed to be implemented in the bfs is adding a return value of int and created a distTo map that maps each vertexs distance to the root. As soon as the program find the vertex that is either the endGoal or greater it returns the map.get of that vertex. The code for the bfs is basically the same thing as sedgwicks just adds a return if the vertex being taken off the queue is the required one, and returns 0 if that vertex is never found.

Pseudo-code for the creation of graph:

```
m = length of move
mine = set of mines
Graph g = new Graph()
For int i = 0; i < endGoal; i++
        if !mine.contains(i)
                graph.addVertex(i)
                if !mines.contains(i + m)
                        graph.addEdge(i, i + m)
                add m – 1 edges less than i if theyre not mines and greater than -1

//use bfs and sedgwicks disTo data structure to find shortest path between root and
endgoal
//return 0 or value of distTo at first vertex equal to or greater than endgoal
return bfs(graph, 0, endGoal)
```