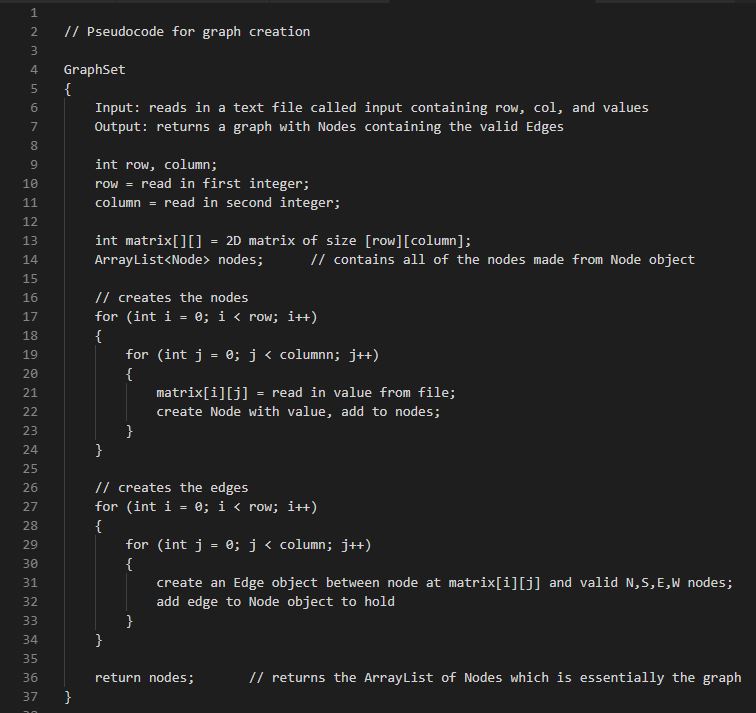
Howard Cheung

11/28/18

COT 4400

Project 3 Report

1. To read in the graph and preserve the integrity of the data, I will model the data in a two dimensional matrix and use it to construct a sort of adjacency list, where every index in the 2D matrix is a Node from a Node class. I will then construct the edges based on the valid jumps from that specific node (source) to the target node. All of the nodes will hold the edges that they are able to jump to, which is similar to the adjacency list representation discussed in class.

The pseudocode of this representation is shown below:

2. The algorithm I will use to solve the problem is Dijkstra’s Algorithm. Initially, all of the nodes, other than the base, will be set to a distance of infinity and the previous node, which will also be the shortest path, will be set to null. While the node with the smallest distance to process is not the target node, it will calculate an alternate distance, which is the current distance plus one (edge weight is treated as a value of one, meaning each jump is of equal value), for all neighboring edges, having four maximum neighbors. The reason why you terminate once the target node (bottom right node) contains the smallest distance in the queue is because the shortest path to that specific node is found (and therefore an existing path) and the previous node is not null, all the way back to the base node.

The pseudocode of this algorithm is shown below (next page):

