Alejandro Rubio
Alejandro Serrano
Prof.: Swaroop Joshi
Assignment 10
12/5/2019

Pacman Analysis:

1) If we ignore the obstacles in the puzzle and calculate the Manhattan distance, we can say that our complexity would be of O(1) since we would only be using a formula to calculate the shortest distance between two points while our actual algorithm has to check the neighbors and obstacles from the given puzzle.
2) A good example where the manhattan distance would work much better than our current algorithm is if we consider a puzzle with the same height and width. For example a 10x10 puzzle without obstacles and having S at the upper left corner and G at the bottom right. In this case, the distance would be shorter if we simply calculate the distance by coordinates instead of traversing it. If we traverse the puzzle by BFS it will take significantly longer since it will need to go south until it reaches the end and then will have to go east. The problem will be that basically all the neighbors will have to be checked during the BFS algorithm.
3) There are many factors that we would need to take into account when having a puzzle of the same height and length. For instance, if we have no obstacles and S and G are straight from each other, the algorithm will work fine regardless. The work BFS will have to do won't be as demanding. On the other hand, if we consider this in terms of coordinates, if S is at (1,1) and G at (8,8) the time for the algorithm will be of O(N^2) since it will have to check every single neighbor. Also, if the puzzle has more walls, the algorithm will perform faster since if it detects a wall, the loop of that search breaks and it has to find a new path.
4) Some of the problems we could potentially encounter when using DFS is that we might not be able to find the shortest path but just a path to the goal, which in this case is not of much use when solving the Pac-Man problem. Another thing we need to consider is that

DFS could potentially look at every single Node inside the adjMatrix, regardless of where the Goal of the puzzle is. Nonetheless, if we only have a straight path from S to G, DFS will be faster than BFS since it'll go in a straight line, while BFS will have to traverse the graph in relation to its neighbors. On the other hand, BFS will be faster if G has a significant distance from S, while DFS will perform much slower. Finally, the size of the graph also comes into play when considering the performance of BFS or DFS. DFS will be much slower because it will have to traverse the entire width of the puzzle for every row.

5) The algorithm of finding the shortest path without edges would be to store the neighbours of each node in a variable. We consider neighbours of a node blanks spaces, the letter S and G, we do not include walls. With that, the bfs algorithm will go to each neighbour and try to see if the value at that position is G, if it's not G that means its a blank space because walls are not part of neighbours and the algorithm will stop once it finds G or it visits all the neighbors. That's why it is also important to store a variable named visited to see if it has already been visited or not.
6) We spend around 10-12 hours completing this assignment.