Josh Roznowski & Erik Booher

Weitian Tong

COSC 311 Project

**Premise:**

A given matrix is filled with the data points of ‘O’, ‘G’, and ‘W’, where ‘O’ represents an open space, ‘G’ represents a guard, and ‘W’ represents a wall. Our program will replace the ‘O’ values with the distance from the open space to the nearest guard without going through a wall. It will replace the ‘G’ values with 0, and ‘W’ values with -1, indicating a guard and a wall, respectively. This program utilizes Breadth First Search along with Queueing.

# **Code Analysis:**

First, we store the dimensions of a 5x5 array into the int variables M and N. Then we create a Node class using the int parameters i, j, and dist. Next, we declared two arrays, row and col that are used to get the row and column numbers of the neighbors of a given cell. The methods isValid and isSafe were used to determine whether the given cell is a valid cell (in range), and whether or not it is an open cell and not a guard or wall.

The next method is essentially the driving force of the code, which is to replace all of the ‘O’ values with their distance from the nearest guard. The method receives the char matrix as a parameter, and then creates a new output matrix with M and N as the size, as well as a new Linked List Queue. For finding the Guard’s location, we use a nested for loop for M and N. We initialize the output matrix cells to -1, and then check if the input matrix cell matches ‘G’. If it does, we set that cell to 0, and that Guard has a distance of 0. Finally, we use a while loop to go through the queue, and using peek, update its adjacent cells. Then, using a for loop, we go through each adjacent cell and run isValid, isSafe, and if both are true, add 1 to dist and enqueue, then dequeue the front cell as it’s distance is found. We then print the output matrix.

**Time Complexity:**

The methods isValid and isSafe would run under the time complexity O(n). findDistance would in this specific instance run for O(n2 ), however, overall, because the input for each for loop is dependent on the parameters of M and N, the time complexity for this would be O(MN), which would also apply to the while loop for the queue. Due to those factors, the overall time complexity of this program is O(MN).

**Conclusion:**

This program was a fun and simple way to utilize Breadth First Search and Queueing, giving us an opportunity to learn and implement these data structures. There is still the possibility of optimizing this program to work under faster time complexities, which we will continue to try for.