# 3050 Final Project Report

# Collaborators: Shelby Heffron, Christian Caldwell, Jason Tersreau

Work Break Down:

* Shelby Heffron:

- Created the major base for final.c and the initial setup.

- Wrote Key Algorithms Such as DFS/graphToMatrix/Reachables

- Continuously re-wrote code as new changes forced prior functions to be changed.

- Took the position of “Project Leader” for the group and helped direct the team to the solution.

* Christian Caldwell:

- Editor (Fixed minor/logical mistakes throughout)

- Output style and stack operations

- Implemented vector.c in final.c

- Created README

- Created Report

* Jason Tesreau:

- Main Researcher

- Editor throughout

- Implemented graph.c in final.c

- Parsing of input file and logical flow of main.

* All members met as a team throughout several weeks to complete the objective.

# Time Complexities

* Depth First Search: O(|V|+|E|)
* Stack Insertions/Deletions: O(1)
* matrixToGraph: O(n\*m) [strictly depends on size of the input matrix(maze)]
* findCoordS/findCoordE/findCoordF/findCoordL: All are O(n\*m)
* isReachableE/isReachableL: Both are O(1)
* printDistances: O(n\*m)
* printSolution: O(n\*m)

# Code Break Down

1. Parse input file into a 2d array.
2. Convert the input maze matrix into a graph with the matrixToGraph function. Insert edges only between valid spaces (no walls, etc) (Prepare for DFS).
3. Create two instances of each graph and send each robot through a Depth First Search to find its exit. (Uses stack within. Start locations are determined by coordinate searches for each robot). Robot 1 will find its destination first, and not be allowed adjacent to robot 2’s starting space. After robot 1 has found its destination, robot 2 will start and not be allowed adjacent to robot 1’s finish area.
4. After DFS is ran, the solutions are printed out for each respective robot to the standard output.
5. It is then determined if the robots can reach their objective through their respective “reachable” functions.
6. If reachable, the distances are printed through finding objective coordinates and calculating the number of vertices taken.
7. The graph is then freed through the free\_graph function, along with all other helper structs, etc.