Capstone Proposal - Robot Motion Planning

Domain Background

This project is motivated by Micromouse competitions that began in the 1970s. A micromouse competition is an event where a small robotic mouse attempts to navigate an unfamiliar grid maze. The maze is made up of a 16 by 16 grid of cells. The micromouse starts in a corner and must find its way to a designated goal area commonly located near the center. The mouse will need to keep track of where it is, discover walls as it explores, map out the maze and detect when it has reached the goal. Having reached the goal, the mouse will typically perform additional searches of the maze until it has found an optimal route from the start to the center. Once the optimal route has been found, the mouse will run that route in the shortest possible time. In this project, our micromouse will attempt to navigate a virtual maze and determine the fastest times possible to traverse an optimal route to the goal area in a series of test mazes.

PROBLEM STATEMENT

The objective is simple - get to the middle of the maze as fast as possible. The robot will start in the bottom-left corner of the maze and must navigate from the origin to the goal in the shortest possible time, and in as few moves as possible. The maze exists on an n x n grid of squares, n even. The minimum value of n is twelve, the maximum sixteen. The whole boundary of the maze is surrounded by walls preventing the micromouse from traveling outside the maze. The initial position will have walls on the left, right, and back sides, so the first move will always be forward. In the center of the grid is the goal room consisting of a 2 x 2 square; the robot must make it here from its starting square in order to register a successful run of the maze.

The micromouse will route a first trial that is exploratory in class where its main aim is to build information about the structure and shape of the maze, that includes all possible routes to the goal area. The micromouse need to travel the goal area in order to complete a successful exploration trial, but is permissible to continue its exploration after reaching the goal. In the second trial, the micromouse will recollect the information from the first trial and will try to reach the goal area in an optimal route. The micromouse's performance will be recorded by adding the following:

- Total number of steps in the exploration trial divided by 30
- Total number of steps to reach the goal in the optimization (second) trial

Each of the trial is capped at 1000 steps. The micromouse can only make 90 degree turns (clockwise or counterclockwise) and can move up to 3 spaces forward or backwards in a single movement.

DATASETS AND INPUTS

The information of maze design is specified in a text file. The first line of the text file explains the number of squares for each side of the maze. The remaining lines describe the actual structure of the maze through comma-separated binary values that explains where all of the walls and openings are situated in each square.

For each and every move, the micromouse should have information about its current environment - its exact location in the maze, the number of walls surrounding it, what its future action will be, and any prior information and routes pertaining to the maze from previous trials.

The position of the micromouse will be defined by a 2-digit pair, such as [2,16]. Each action comprises movement which will be a number from -3 to 3, and rotation is expected to be an integer taking one of three values: -90, 90, or 0, indicating a counterclockwise, clockwise, or no rotation, respectively.

SOLUTION STATEMENT

Each trial tries to solve a different problem. In the first trial, the main problem is to solve - what is the actual layout of the maze? A possible solution in the first trial will be a thorough knowledge of the maze layout and all possible paths leading to the goal area. In the second trial, we are trying to reach the goal in an optimized route, so the target solution would be reaching the goal in the fewest possible moves.

This environment is easily quantifiable, because we are constantly measuring the number of total moves taken along with the total time the micromouse is navigating the maze. After the exploration trial, the micromouse must have recorded knowledge of multiple possible routes to the goal, and must choose the shortest route (or the route that requires the fewest steps) during its optimization round. The solution can be simulated by repeating the optimization trial for additional rounds. Supposing the micromouse initially identified and traveled in an optimal route, its path to the goal should remain the same if it were subjected to additional trials.

BENCHMARK MODEL

The benchmark model will be calculated by the performance score discussed in the ***Problem Statement*** section: **score = [Number of Steps in Run 2] + [Number of Steps in Run 1 / 30]**. A maximum of one thousand time steps are allotted to complete both runs for a single maze, this will be the most basic form of a benchmark performance. The benchmark model will consist of a micromouse that makes a random movement decision at each step. If the mouse is able to navigate to the goal in fewer than 1000 steps during its exploration trial, a new benchmark should be set to the total number of steps in Run 1. Because the micromouse has gained knowledge of the maze layout in its exploration trial, it should then optimize its route to the goal in Run 2. Regardless of the number of steps taken during Run 1, by nature of design, each maze has an optimal route - which is the minimum amount of steps required to get from the starting location to the goal area.

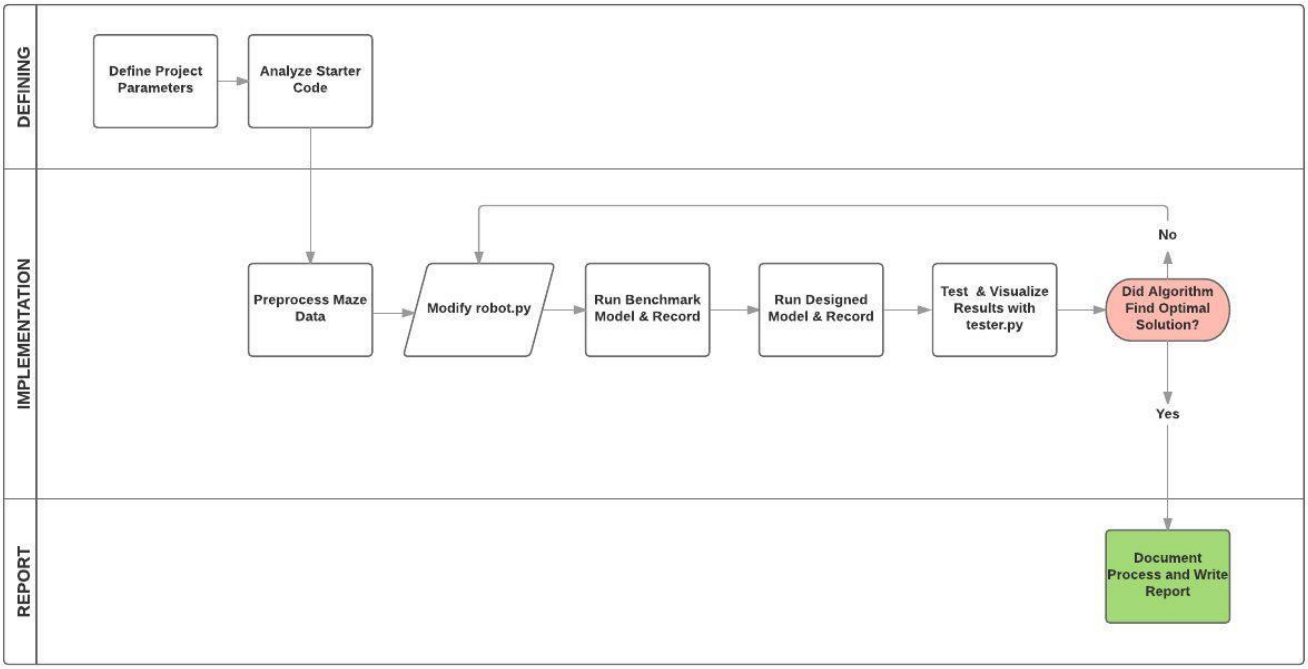
EVALUATION METRICS

As per the previous sections, the main evaluation metric to quantify the performance of the benchmark and solution models is: **score = [Number of Steps in Run 2] + [Number of Steps in Run 1 / 30].** This evaluation metric is impacted by both the exploration run and optimization run, however the optimization run will impact the score significantly more than the exploration run. Our goal is to minimize this score, which would be the result of an optimal run.

For example: let's assume the micromouse took 400 steps during its exploration run, and 16 steps during its optimization run. The score would be equal to: **16**(***steps from Run 2***) + **400/30**(***steps from Run 1 divided by 30***) = **29**. If the optimization algorithm found the optimal route for this particular maze, then 29 would be the optimal score.

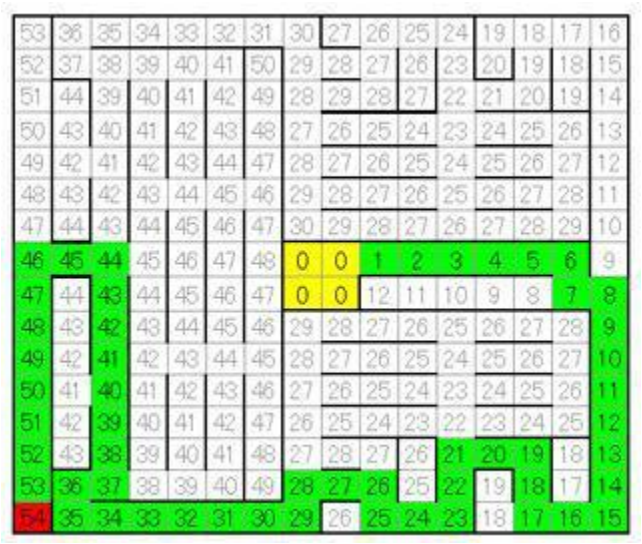
PROJECT DESIGN

To obtain a solution, my workflow will begin with defining and understanding all essential project parameters. I will then examine all the starter code that is given: robot.py, maze.py, tester.py, showmaze.py, and test_maze_##.txt. Next, I am going to preprocess all the data which is entirely handled in the Maze class, where the contents of the maze are given in a text file and decoded into a virtual maze. After getting the mazes set up, I will begin to modify robot.py by adding appropriate algorithms and functionality suited for exploration and optimization runs. I will then run the algorithms with my benchmark model and record the performance, then run the optimized robot.py model and record the performance. I will test my results using tester.py, and will graphically visualize the micromouse performance in each of maze. If the micromouse did not find an optimal solution, then I will revise robot.py and repeat the process until it finds the optimal solutions. Finally, I will document my process and write up a detailed report.

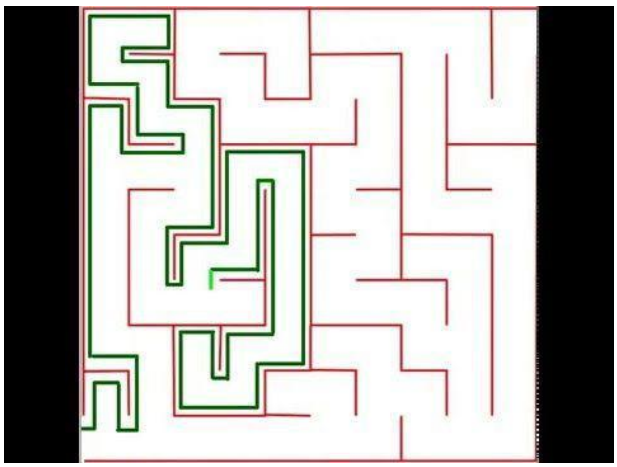


There are several algorithms that I am considering for this project. Both the exploration and the optimization run will need suitable algorithms because each run serves a different purpose.

Flood Fill: Algorithm that starts in the center area and will fill each surrounding cell with a number of its relative distance from the goal area. After updating the wall information for the current cell, the robot starts to flood the matrix to find the shortest path to the goal.



Wall Follower: A Wall Follower algorithm works by simply leading the micromouse to take every left or right turn possible, which should finally lead to the goal area (assuming the micromouse does not get caught in a circular loop). An example of a Left Hand Rule (LHR) is shown below.



A*: A* is a best-first search algorithm that searches for all possible solutions to the goal area and will choose the shortest path from the starting node to the ending node. Starting from a specific node, it creates a tree path of each possible direction until it reaches the goal area.

Dijkstra's: Dijkstra's algorithm is an algorithm that finds the shortest path from one node to all other nodes in its network. By finding the distance to all nodes in its path, it will inevitably create a connection with the nodes represented by the goal area. Unlike A*, which focuses on finding the path between two single nodes, Dijkstra's algorithm links a path between all nodes on a network.

Breadth-First Search: Is an algorithm that will start at a tree root and consider its closest neighbors before looking on to its next level neighbors.

Depth-First Search: This is similar to Breadth-First Seach, but will start at a root and follow a branch as far as possible before considering a different path.

REFERENCES

Udacity Project Files
https://docs.google.com/document/d/1ZFCH6jS3A5At7_v5IUM5OpAXJYiutFuSIjTzV_E-vdE/pub

Maze Solving Algorithms on YouTube
https://www.youtube.com/watch?v=NA137qGmz4s

Micromouse USA
http://micromouseusa.com/

USC Micromouse Project
http://robotics.usc.edu/~harsh/docs/micromouse.pdf