Artificial Intelligence 3007 - Assignment 1
Trek Hopton - a1696375

Implementation and Results of Robot Path Planner

The program takes an environment, start, and goal as input and returns a list of moves to get from the start to the goal.

An A* search algorithm was implemented in c++ to find an optimal path from the start to the goal, the heuristic used is the manhattan distance from the cell to the goal.

Design

The algorithm works as follows:

1. Check if the right number of arguments are given.

2. Make sure the input is valid:
    - Check if start and goal are within environment bounds, if not return error
    - Check if start and goal are in an unoccupied cell, if not return error
    - Check if the start is the same as the goal, if so return empty move list

3. Two lists are created; open, and closed lists. Open is a list of cells ordered by their f value, implemented using a set data structure. Closed is a grid of cells of width and height matching the environment. Cells in the closed grid start unoccupied and as the are closed, they are marked by setting their occupied state to true.

4. Take a cell from open with the lowest f value and add it to closed, initially this will be the start coordinate.

5. If the cell is the goal, trace the path back to the start by following the parent pointers of each cell, then return the found path in the form of a string specifying moves (L, R, U, D).

6. Otherwise check all the nearby cells and see if they are valid options to move to. If they are:
    - Calculate their g, h and f values; g being incremented each step, h being the manhattan distance from the cell to the goal, f being the sum of the two.
    - Set their parent to the current cell
    - Set their direction to L, R, U or D depending on the relationship to the current cell
    - Add them to the open list (the list is a set so they are automatically sorted by f)

7. Go back to step 4 until the open list is empty (no possible path) or the goal is found.

Results

Below are the results of the program when using the provided test environment, “testgrid_large.txt”. The first line is the input, the following line/s is the output. Below the output is an image of the path in the cases where one has been found (only used for debugging and not an output of the final program).

testgrid_large.txt 0 1 4 14
goal not found

testgrid_large.txt 0 8 19 14
R U R R R R R U R R R R R D R R R R R R R D D D D D R D D

testgrid_large.txt 10 3 5 0
L L U U L U L L

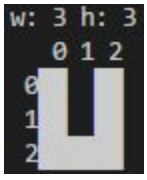
testgrid_large.txt 5 14 2 0
start or goal is occupied
error

testgrid_large.txt 0 13 0 0
start or goal is occupied
error

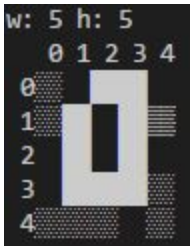
Below are the results of some of my own test cases:

testgrid_large.txt 2 0 16 23
goal is not within environment bounds
error

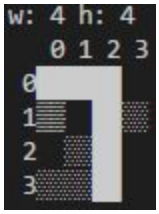
test01.txt 0 0 2 0
D D R R U U



testgrid_small.txt 2 0 1 1
R D D D L L U U



test02.txt 0 0 2 3
R R D D D



testgrid_large.txt 2 0 16 11
R D D D L D L L D D D R R R R R R U R R R R R D R R R R R R R D D D D D L L U