# MP 1:Search

ABHI KAMBOJ

ECE 448

Section R: Prof. Hockenmaier

2/3/2019

# Section I:

Algorithms (Search). This section should describe algorithms and data structures used for all four search strategies. Answer questions like: what is a state? what is a node? are they the same or different in your implementations? What is the frontier? Do you maintain an explored states list? How are repeated states detected and managed?

A state is a description of the environment around the agent and the actions it can take. A node is representation of the state in a search tree where actions correspond to branches in the tree leading to other states or nodes. In my case, the states were represented as possible x,y locations the agent could be on the 2d plane of the maze and the possible actions the agent could take so that it doesn’t run into the wall. The nodes were ordered pairs, or in python (x,y) tuples of the location. A dictionary was used to keep track of which tuples the agent has visited so there was no repetition, and a list was used to store the current set unexpanded nodes or the frontier. The dictionary held pairs with the key being a node and the value being the previous node that caused that node to be added to the frontier. Then when the goal is reached, the dictionary can be traversed from the goal to the start state and a path can be created from the set of visited nodes.

The frontier was a list however was implementing a different data structure for each algorithm. For Breadth First Search (BFS), the list was used as a First In First Out (FIFO) queue frontier holding all the nodes that needed to be visited. Depth first search was similar except used a Last In First Out (FIFO) stack as the frontier. The greedy algorithm traversed the frontier every step to find the node with the smallest Manhattan distance and expand it. Finally, the astar algorithm also had the frontier store how far each node is from the start and traversed the frontier to find the smallest Manhattan distance in cost from goal plus cost to goal.

MAKE NOTE THAT EXPANDED NODES IS LARGER THAN TOTAL NODES BECAUSE HAVE TO EXPAND MORE THAN ONCE GOING FROM ONE GOAL TO THE NEXT!

Heuristic for A\* is just Euclidean distance

Check with gau about distances for A\* search—why is manhattan different from Euclidean?? Shouldn’t be!

# Section II:

Section II:

Algorithms (A\* and Greedy BFS). This section should describe the heuristic(s) used for A\* and Greedy BFS, for both the single dot and multiple-dot situations Provide proof that the heuristics for A\* are admissible

For part 1, I implemented the A\* and Greedy BFS algorithms both with Manhattan distance heuristic. This assumes that the cost to goal is just the sum of the absolute value of the difference between the goal’s x coordinate and the agents x coordinate and the goal’s y coordinate and the agent’s y coordinate: hmanhattan = |xgoal - xagent| + |ygoal - yagent|. For part 2, I implemented the A\* algorithm with the Euclidean distance heuristic which assumes the cost to goal is the difference in distance between the goal and agent’s position in 2D space: heuclidean = sqrt((xgoal - xagent)2+( ygoal - yagent)2).

Both heuristics are admissible. An admissible heuristic must never overestimate the cost to goal, so the heuristic’s assumed cost to goal must be less than or equal to the actual distance. This is true for the Manhattan distance because if there were no walls and the agent can only move up, right, down and left the quickest way to the goal would be finding the difference in x and y locations and adding them which is what Manhattan distance does. The Euclidean distance heuristic can be proven to be admissible by the triangle inequality. No one side of a triangle can be greater than its other two sides and the agent cannot move diagonally. Therefore, Euclidean distance be the hypotenuse of a triangle or sum of smaller triangles that the robot would have to take to reach the goal.

# Section III:

## BFS

### Medium Maze



Results

Path Length: 111

States Explored: 645

### Big Maze

Results

Path Length: 183

States Explored: 1285

### Open Maze

  
Results

Path Length: 52

States Explored: 558

## DFS

### Medium Maze

Results

Path Length: 175

States Explored: 372

### Big Maze



Results

Path Length: 537

States Explored: 985

### Open Maze

Results

Path Length: 252

States Explored: 277

## Greedy Algorithm

### Medium Maze



Results

Path Length: 147

States Explored: 347

### Big Maze

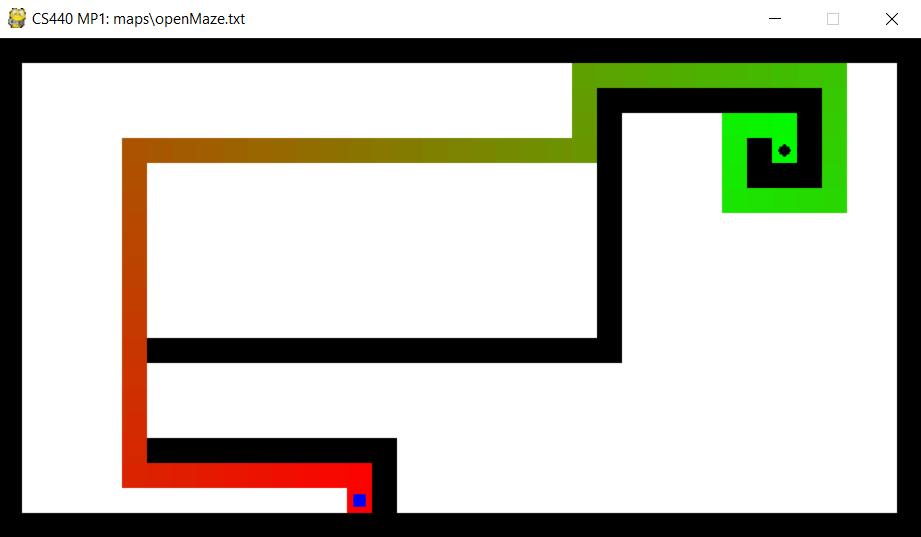


Results

Path Length: 277

States Explored: 456

### Open Maze



Results

Path Length: 70

States Explored: 95

## Astar Algorithm

### Medium Maze



Results

Path Length: 112

States Explored: 417

### Big maze

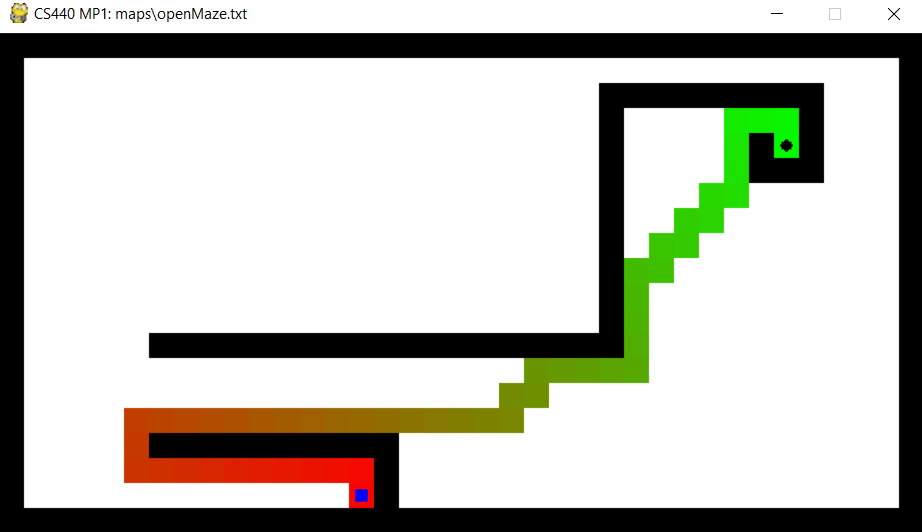


Results

Path Length: 184

States Explored: 1265

### Open Maze



Results

Path Length: 53

States Explored: 436

# Section IV: