ECE 448

ARTIFICIAL INTELLIGENCE

SPRING 2023

MP1: Search Algorithms Report

KE XU (3190110360)
JIARUN HU (3190110383)
ZHEYANG JIA (3190110096)

Prof. Hongwei Wang, Zhejiang University

Prof. Hasegawa-Johnson, University of Illinois Urbana-Champaign

Contents

1 Algorithms (Search) - Algorithms and Data Structures

1.1 BFS

To implement breadth first search, we use a stack to store our frontier(in practice, we store a path branch with latest explored neighbor), which is a FIFO data structure. At the beginning. we store our start point to the queue. Then for each location, we pop current path branch from queue, explore neighbors, and use these neighbors to generate our new path branches, and add these paths to the beginning of queue. And we repeat this procedure , pop the latest branches and generate new branches. With popping each path branch queue,explore their vertex neighbor and update them sequentially, we can explore and update different path branch at the same time. If we find target in one path branch, then stop.

1.2 DFS

To implement depth first search, we use a stack to store our frontier(in practice, we store a path branch with latest explored neighbor), which is a FILO data structure. At the beginning. we push our start point to the stack. Then for each location, we pop current path branch from stack, explore neighbors, and use these neighbors to generate our new path branches, and push these paths to the stack. And we repeat this procedure, pop the latest branches and generate new branches. If one branch is impasse, we pop this visited path branch and explore the path branch at the top of stack now, which means exploring the another neighbor of last point. Repeat this procedure until we find the target.

1.3 Greedy

Greedy algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem. In this maze setting, we can move to four neighbors, the up, down, left and right one, among which we will choose the position closest to the target. At each step, our current state includes the current coordinates and the current distance to the target (i.e. we chose Manhattan as the criterion for the maze problem). That is to say, when we want to select the next step from the neighbors, we will pick up items that are the smallest Manhattan distance from the target. Each time we add a new point to the priority queue (i.e. sort based on Manhattan distance to the objective). We maintain a list of states explored and the source of the node being accessed, which means that the node being accessed is queued as a neighbor. The next time we visit the node being visited and we can simply skip it.

1.4 A*

A state is the current path. A node is the the current point. But actually they are the same because every time we arrive a new point, we add the path to the path list as well. Our goal function is

$$f(n) = g(n) + h(n)$$

where $g(n)$ is the path length it already has gone through g(n)=len(path), and $h(n) = h_min + h_mst$ is the heuristic function. Here we use Manhattan distance for basic distance. h_min is the heuristic distance from current point to nearest objective. h_mst is the heuristic distance for the total minimum cost of the minimum spanning tree(MST) for all the objective nodes. The frontier is a Priority Queue which is always sorted from small to large. We save the explored states list and, but do not use it for search algorithm. Instead, we use it to detect repeated states. To manage repeated states, we just skip to next point in the frontier, pop one node to go backward. On the other hand, we add current point to that explored states list when it is not in that list or the time for changing target(in multiple search problem), which means we are finding path to the target.

2 Algorithms (A* and Greedy BFS) - Heuristic(s)

2.1 A*

We use the same heuristic function for both the single dot and multiple-dot situations Provide proof that the heuristics for A* are admissible. In single dot situation, $h_mst = 0$ because only one node is in that MST. $h(n) = h_min$ is the Manhattan distance from current point to the target. It is always no greater than the real path. So it is admissible. In single dot situation, h_mst is the minimum cost from one node in objectives to all other nodes, h_min is the minimum cost from current point to these objectives. So the total heuristic function $h(n) = h_min + h_mst$ is always no greater than the real path. So it is admissible as well.

2.2 Greedy BFS

For greedy BFS algorithm, the heuristics is the Manhattan distance to the objective. When we select the next state from frontier, we will choose the one with least heuristics.

3 Results (Basic Pathfinding)

3.1 BFS

Path Length: 69, States Explored: 269

3.1.1 mediumMaze.txt

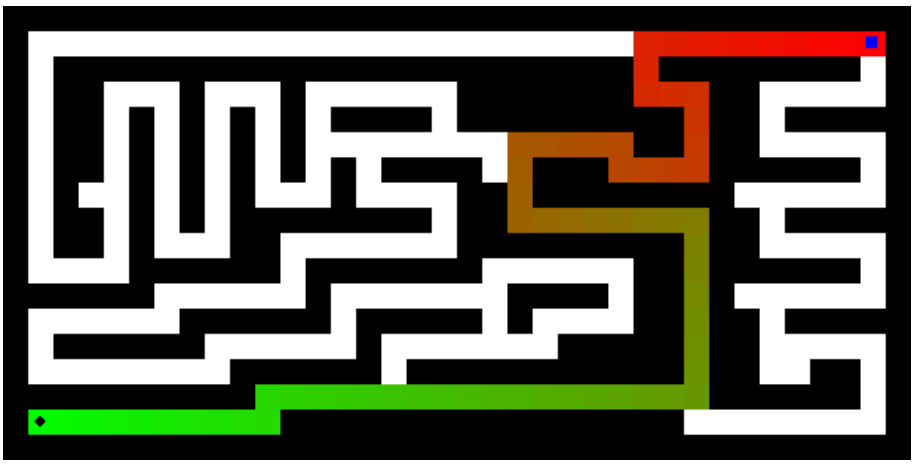

Figure 3.1.1 BFS Solution for Medium Maze

3.1.2 bigMaze.txt

Path Length: 211, States Explored: 619

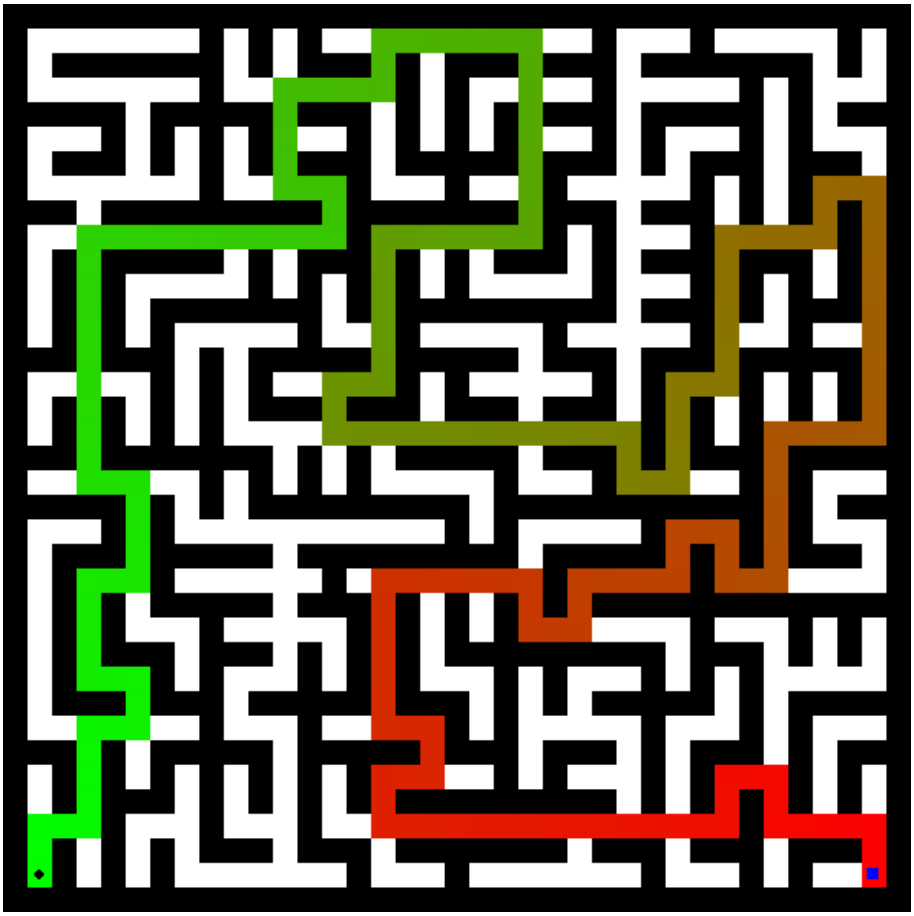

Figure 3.1.2 BFS Solution for Big Maze

3.1.3 openMaze.txt

Path Length: 55, States Explored: 682

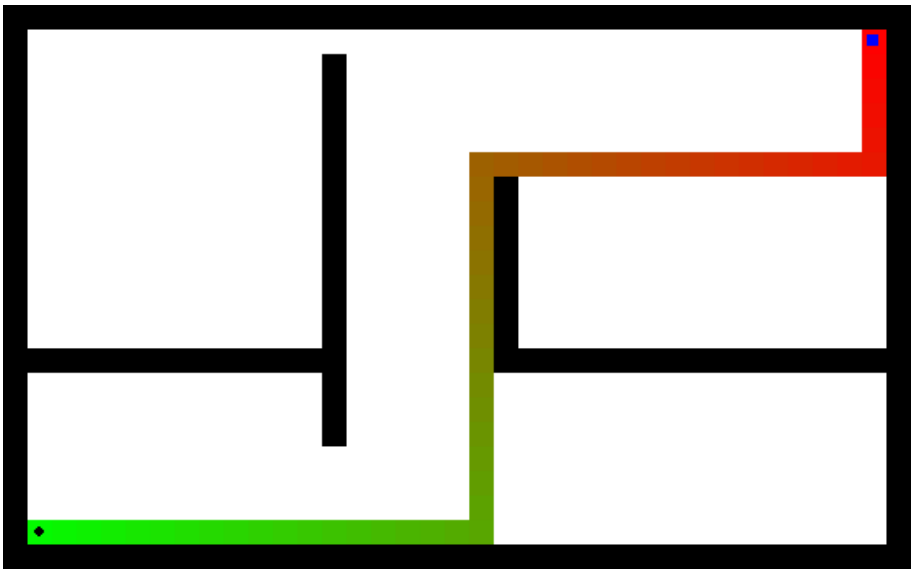

Figure 3.1.3 BFS Solution for Open Maze

3.2 DFS

3.2.1 mediumMaze.txt

Path Length: 131, States Explored: 147

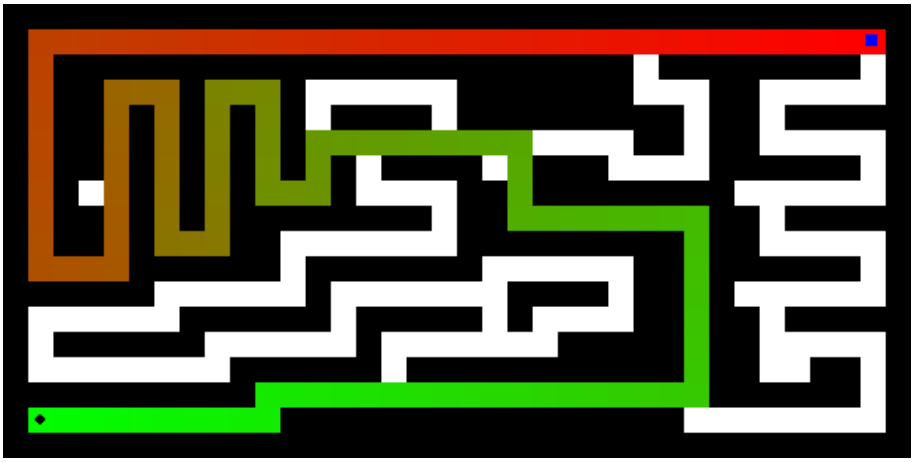

Figure 3.2.1 DFS Solution for Medium Maze

3.2.2 bigMaze.txt

Path Length: 211, States Explored: 427

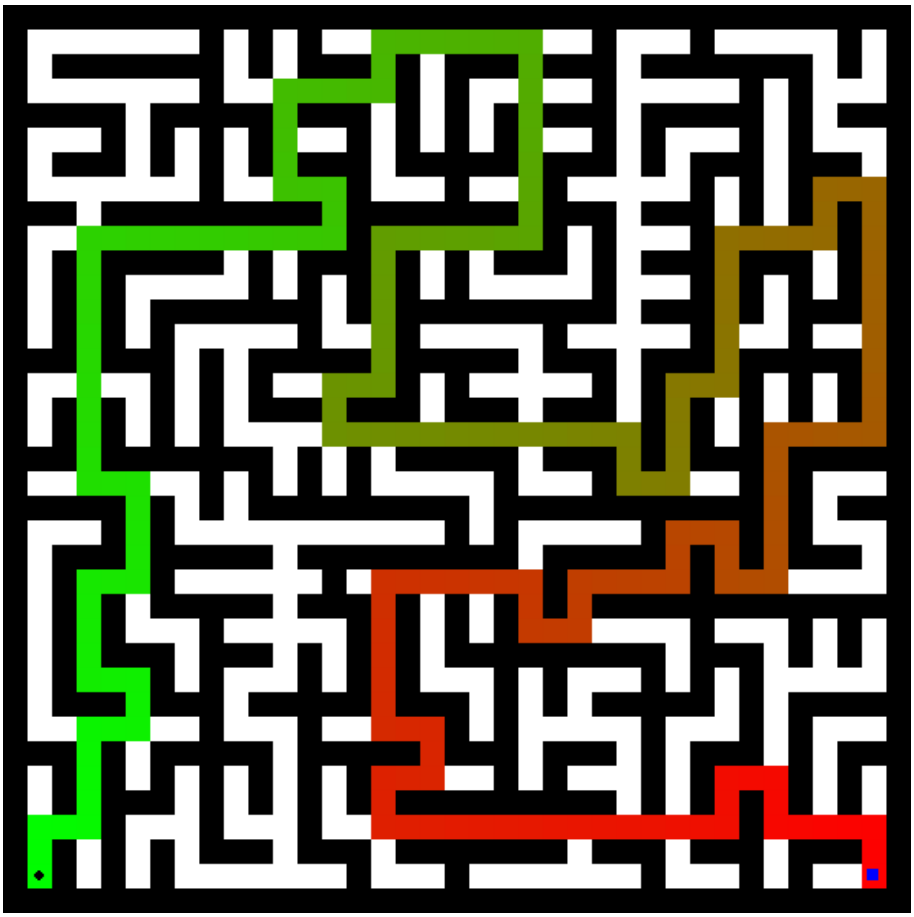

Figure 3.2.2 DFS Solution for Big Maze

3.2.3 openMaze.txt

Path Length: 299, States Explored: 650

Figure 3.2.3 DFS Solution for Open Maze

3.3 Greedy

3.3.1 mediumMaze.txt

Path Length: 153, States Explored: 158

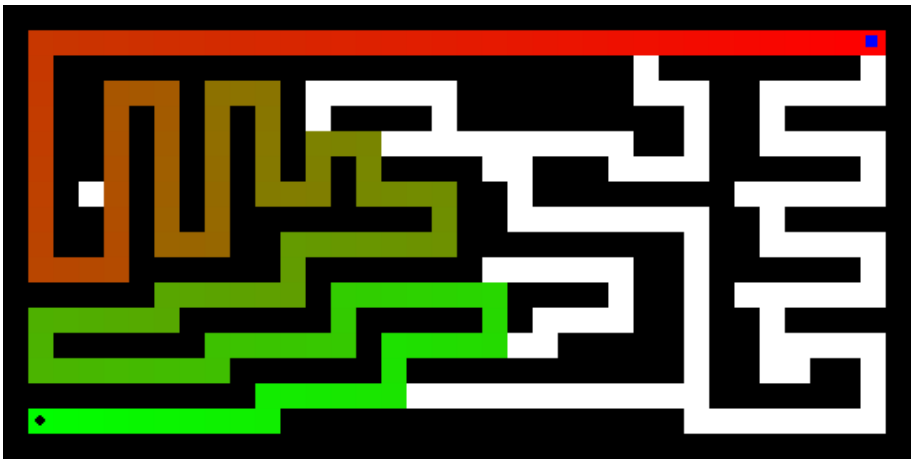

Figure 3.3.1 Greedy Solution for Medium Maze

3.3.2 bigMaze.txt

Path Length: 211, States Explored: 454

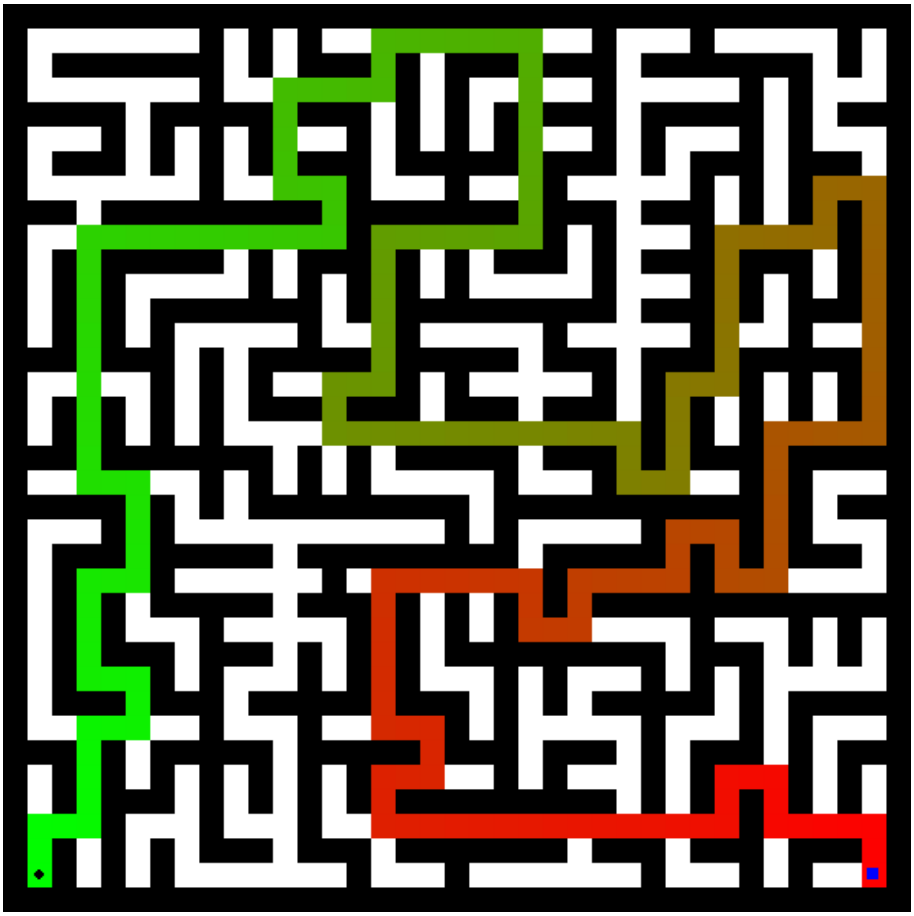

Figure 3.3.2 Greedy Solution for Big Maze

3.3.3 openMaze.txt

Path Length: 55, States Explored: 212

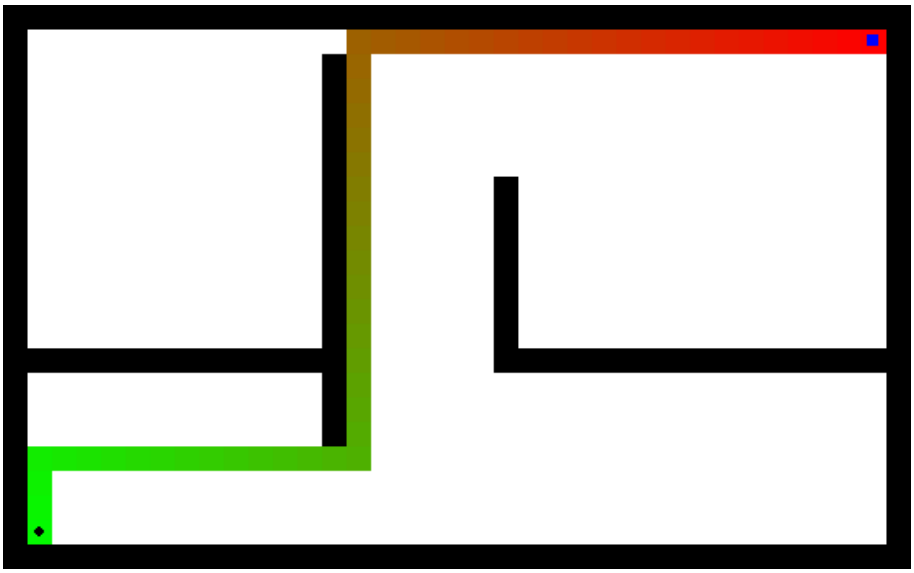

Figure 3.3.3 Greedy Solution for Open Maze

3.4 A*

3.4.1 mediumMaze.txt

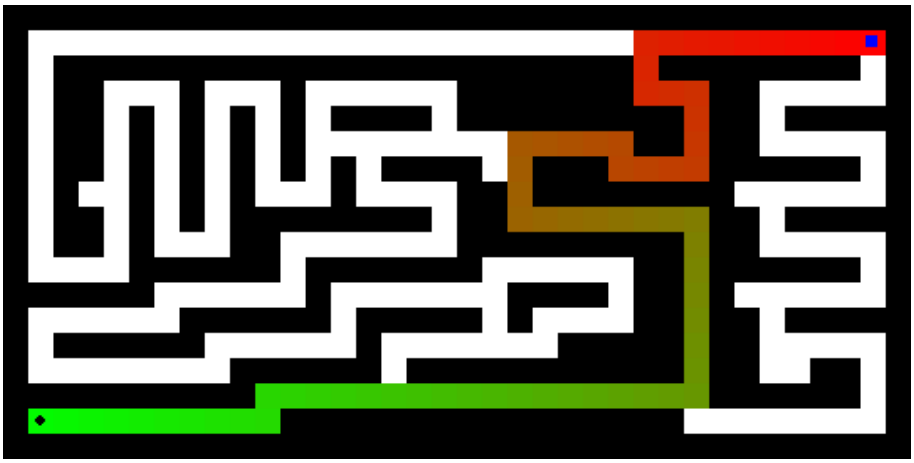

Figure 3.4.1 Astar Solution for Medium Maze

3.4.2 bigMaze.txt

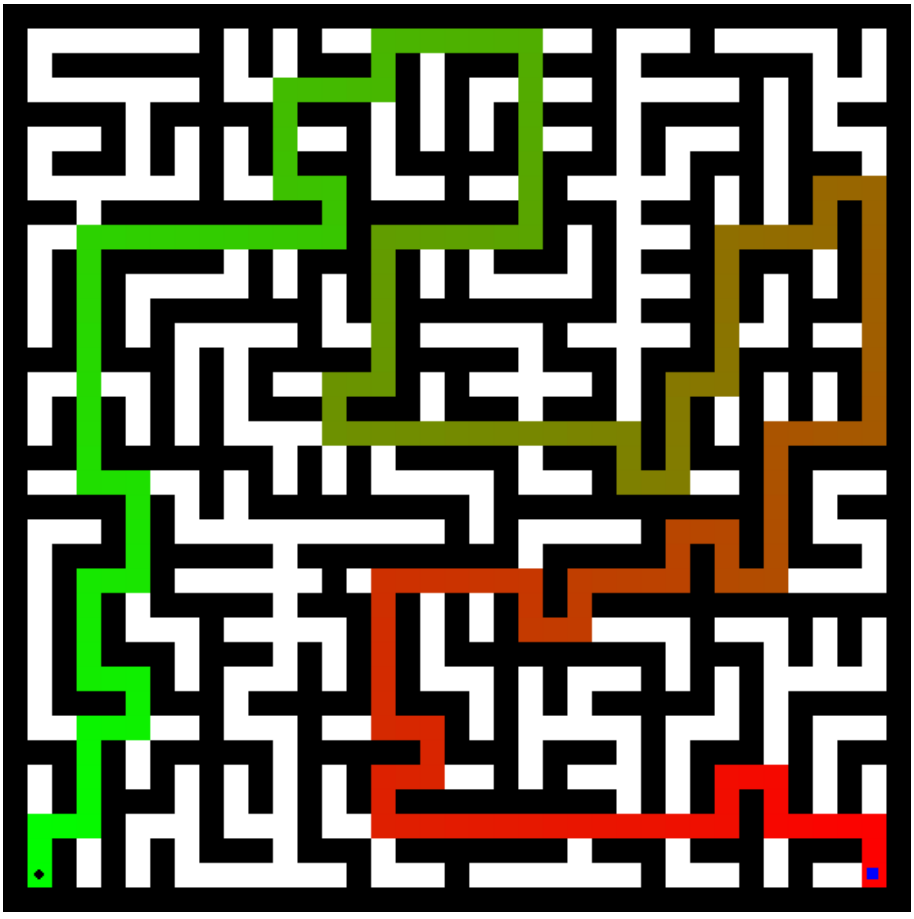

Figure 3.4.2 Greedy Solution for Big Maze

3.4.3 openMaze.txt

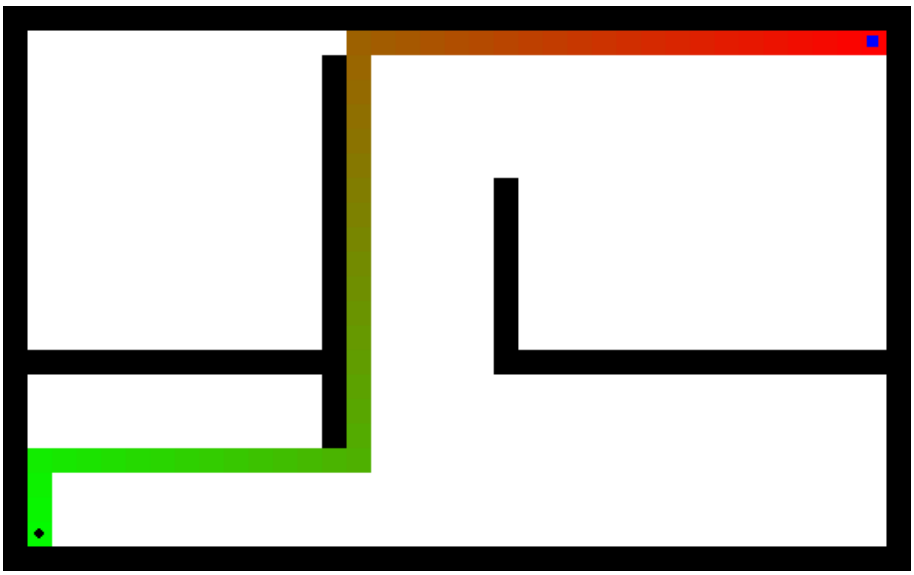

Figure 3.4.3 Astar Solution for Open Maze

4 Results (Search with multiple dots)

4.1 tinySearch.txt

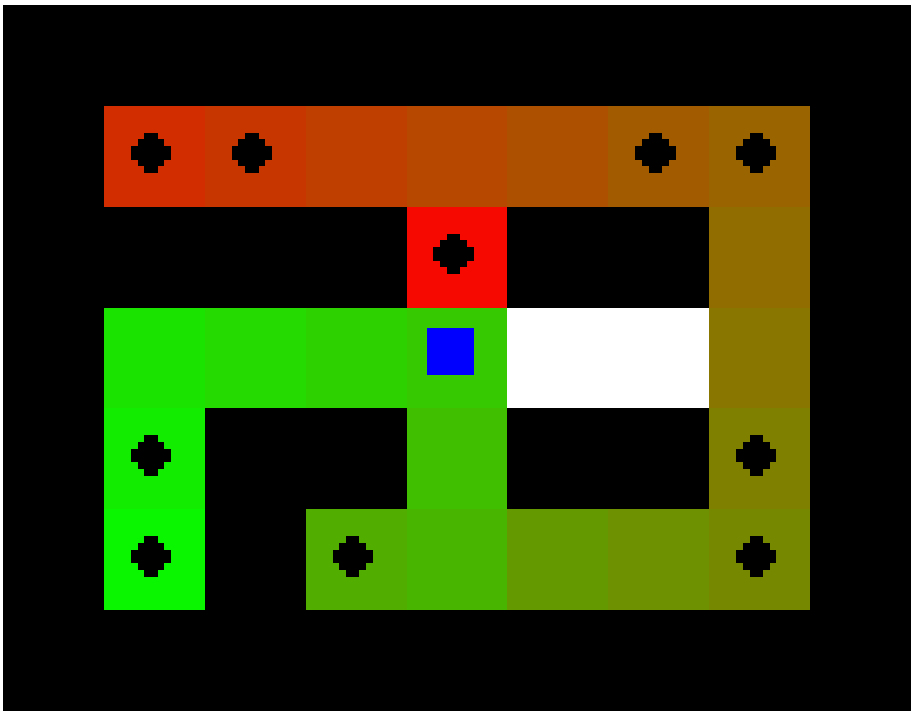

Figure 4.1 Astar Solution for Tiny Search

4.2 smallSearch.txt

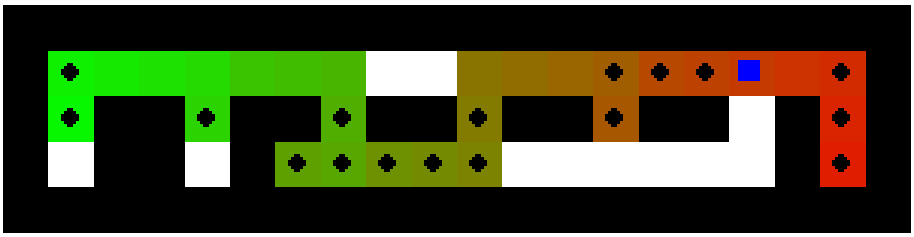

Figure 4.2 Astar Solution for Small Search

4.3 mediumSearch.txt

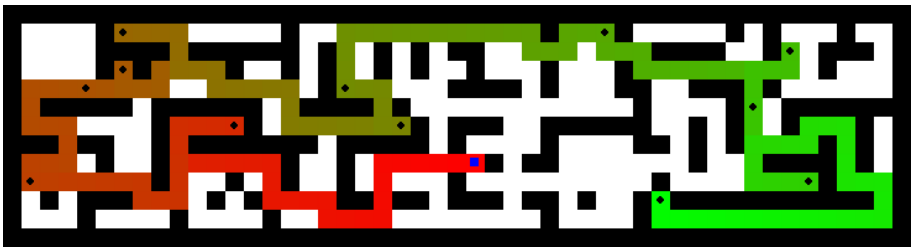

Figure 4.3 Astar Solution for Medium Search

5 Extra Credit

6 Statement of Contribution

6.1 Ke Xu

Ke Xu mainly focuses on the Greedy Algorithm implementation and Report.

6.2 Jiarun Hu

Jiarun Hu mainly focuses on DFS Algorithm implementation and Report.

6.3 Zheyang Jia

Zheyang Jia mainly focuses on BFS and A* Algorithms implementation.