# CS-470: Artificial Intelligence: Project 1

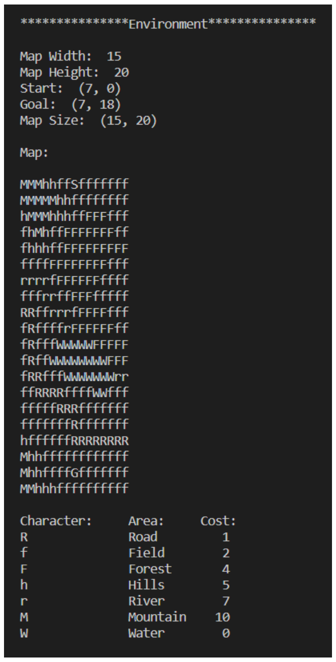
Taylor Martin
University of Idaho
Moscow, Idaho, USA
mart8517@vandals.uidaho.edu

## 1 ABSTRACT

This project utilized AI pathfinding techniques from the field of Artificial Intelligence, specifically as taught in CS-470. The primary objective focused on importing environmental data from a designated data file, and developing an AI agent capable of traversing the map from an initial point to a target destination. To accomplish this objective, a diverse range of searching algorithms were employed. The adopted methodology exhibited highly favorable outcomes, thereby instilling a high level of confidence in the generated results.

## 2 METHODS

The following section outlines the various methodologies employed in the development of the artificial intelligence (AI) pathfinding agent, the environment designed for agent traversal, and the implemented searching algorithms..



```
***************Environment***************

Map Width:  15
Map Height:  20
Start:  (7, 0)
Goal:  (7, 18)
Map Size:  (15, 20)

Map:

MMMhhffSfffffff
MMMMMhhffffffff
hMMMhhhffFFFfff
fhMhffFFFFFFFff
fhhhffFFFFFFFFF
ffffFFFFFFFFfff
rrrrfFFFFFFffff
fffrrffFFFfffff
RRffrrrfFFFFfff
fRffffrFFFFFFff
fRfffWWWWWFFFFF
fRffWWWWWWWWFFF
fRRfffWWWWWWWrr
ffRRRRffffWWfff
fffffRRRfffffff
fffffffRfffffff
hffffffRRRRRRRR
Mhhffffffffffff
MhhffffGfffffff
MMhhhffffffffff

Character:     Area:     Cost:
R              Road         1
f              Field        2
F              Forest       4
h              Hills        5
r              River        7
M              Mountain    10
W              Water        0
```

**Figure 1: Map Data Loaded From Provided Datafile**

### 2.1 Data Preparation

In order to create an environment for our AI agent's search and traversal operations, a custom Python module "environment.py" was developed. This module encompassed the implementation of the custom Environment class. Within this class, the "load_data()" method was devised to facilitate the opening of the designated map data file, the parsing of its contents, and the generation of our map representation. Additionally, several auxiliary methods were created to facilitate the retrieval of information from the loaded data, such as the starting position, goal position, state cost, and a legend detailing the symbols employed on the map. Once the data was successfully loaded, parsed, and stored, it was ready for the search and traversal operations conducted by our AI agent.

*2.1.1 Map.* Below is a figure showing the ASCII format of the Map utilized in this project.

### 2.2 Breadth First Search

Starting with the initial position in the open list, the function iterates until the open list is empty. It selects the next node from the open list as the parent node and checks if it matches the goal position. If a goal is found, the path cost is calculated by summing the state costs along the path, and the path length is determined. The parent node is then returned.

If the parent node is not the goal, the function expands it by generating its child nodes. Unexplored child nodes are recorded and added to the open list for further exploration.

After processing all child nodes, the parent node is added to the closed list to avoid revisiting it.

If no path is found and the open list is empty, the function concludes by returning None.

This Breadth First Search implementation explores the search space systematically, examining all possible paths from the starting point to the goal in a breadth-first manner.

### 2.3 Greedy Best First

Starting with the initial position and its cost of 0 in the open list, the function iterates until the open list is empty. It selects the node with the lowest cost from the open list as the parent node.

If the parent node matches the goal position, the function calculates the path cost by summing the state costs along the path, determines the path length, and returns the parent node and its cost.

The parent node is added to the closed list for tracking explored nodes. The closed list is then sorted based on heuristic values, typically using the Manhattan distance.

For each child node generated from the parent, the algorithm calculates its heuristic value. If the child is not in the closed list, it is either added to the open list or updated if its heuristic value is lower than the existing one. If the child is in the closed list but has a lower heuristic value, it is updated in the open list.

The open list is sorted in ascending order based on state costs, and the process continues until the open list is empty.

If no path is found and the open list becomes empty, the function concludes by returning None.

This Greedy Best First Search implementation efficiently explores the search space by prioritizing nodes with the lowest heuristic values, aiming to find a path from the starting point to the goal.

## 2.4 A*

Starting with the initial position and a cost of 0 in the open list, the function iterates until the open list is empty. It selects the node with the lowest cost as the parent node.

If the parent node matches the goal position, the function calculates the path cost and length, and returns the parent node and its cost.

The parent node is added to the closed list to track explored nodes.

For each child node generated from the parent, the algorithm calculates its cost based on the specified heuristic. The cost considers the state cost, heuristic value, and accumulated cost.

If the child is not in the closed list, it is added to the open list or updated if it is already present with a lower cost. If the child is in the closed list but has a higher cost, it is updated in the open list.

The open list is sorted based on state costs, and the process continues until the open list is empty.

If no path is found, the function returns None.

This A* Search implementation efficiently explores the search space, considering different heuristic functions, to find an optimal path from the starting point to the goal.

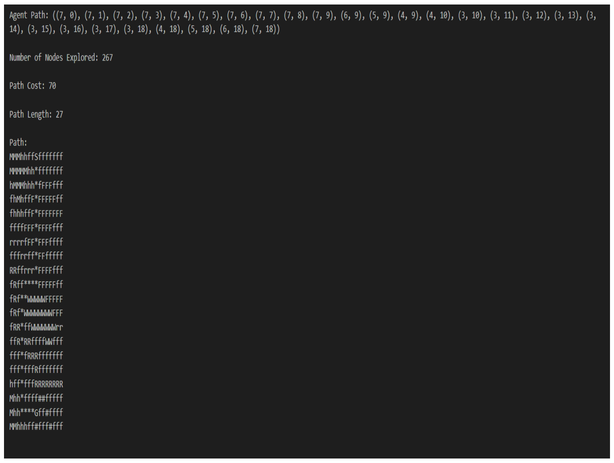
## 2.5 Lowest Cost Search

Lowest Cost Seach, AKA Uniform Search, was implemented the same way as A*. The main difference is that for LCS, we don't use a heuristic. We use the current cost of the state and the cost of the child state when expanding nodes.

## 3 RESULTS

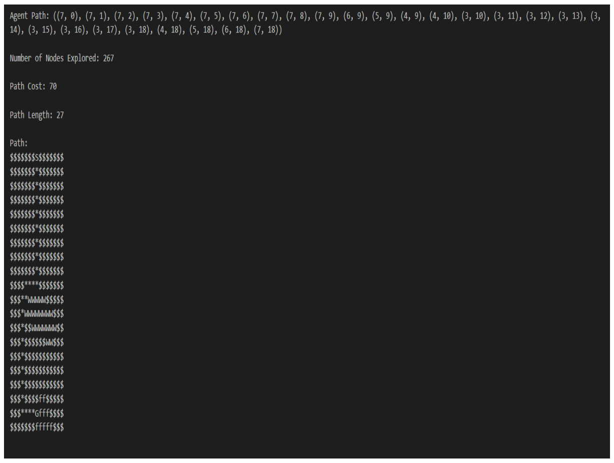
Below are the results generated by this project. These images show the various paths traversed by out AI agent, utilizing several different searching algorithms. They also show the nodes explored by the agent, the nodes on the open list, the nodes on the closed list, path length, and path cost. Providing insights to how these various searching algorithms compare to one another. S = Starting point, G = Goal point, * = agent path, # = open list, and $ = closed list.

## 3.1 BFS Path & Exploration



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (3,
14), (3, 15), (3, 16), (3, 17), (3, 18), (4, 18), (5, 18), (6, 18), (7, 18))

Number of Nodes Explored: 267

Path Cost: 70

Path Length: 27

Path:
MMMhhffSfffffff
MMMMMhh*fffffff
hMMMhhh*fFFFfff
fhMhffF*FFFFFff
fhhhffF*FFFFFFF
ffffFFF*FFFFfff
rrrrfFF*FFFffff
fffrrff*FFfffff
RRffrrr*FFFFfff
fRff****FFFFFff
fRf**WWWWWFFFFF
fRf*WWWWWWWWFFF
fRR*ffWWWWWWWrr
ffR*RRffffWWfff
fff*fRRRfffffff
fff*fffRfffffff
hff*fffRRRRRRRR
Mhh*ffff##fffff
Mhh****Gff#ffff
MMhhhff#fff#fff
```

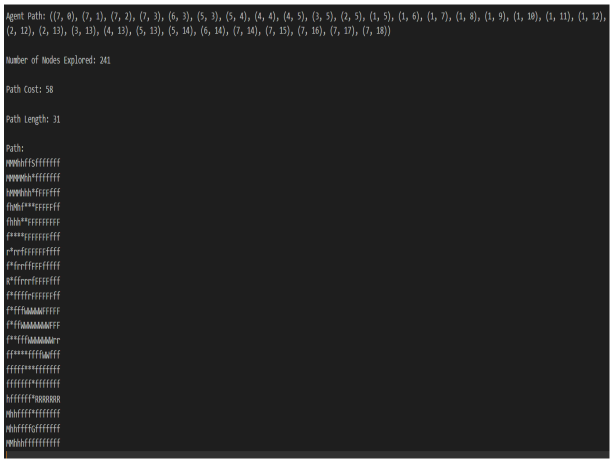
Figure 2: BFS Agent Path



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (3,
14), (3, 15), (3, 16), (3, 17), (3, 18), (4, 18), (5, 18), (6, 18), (7, 18))

Number of Nodes Explored: 267

Path Cost: 70

Path Length: 27

Path:
$$$$$$$S$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$****$$$$$$$
$$$**WWWWW$$$$$
$$$*WWWWWWWW$$$
$$$*$$WWWWWWW$$
$$$*$$$$$$WW$$$
$$$*$$$$$$$$$$$
$$$*$$$$$$$$$$$
$$$*$$$$$$$$$$$
$$$*$$$$ff$$$$$
$$$****Gfff$$$$
$$$$$$$fffff$$$
```

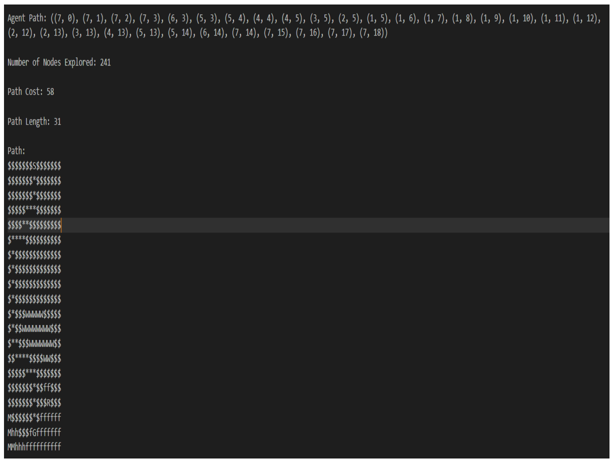
Figure 3: BFS Agent Exploration

## 3.2 Lowest Cost Search Path & Exploration



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (6, 3), (5, 3), (5, 4), (4, 4), (4, 5), (3, 5), (2, 5), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11), (1, 12),
(2, 12), (2, 13), (3, 13), (4, 13), (5, 13), (5, 14), (6, 14), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18))

Number of Nodes Explored: 241

Path Cost: 58

Path Length: 31

Path:
MMMhhffSfffffff
MMMMMhh*fffffff
hMMMhhh*fFFFfff
fhMhf***FFFFFff
fhhh**FFFFFFFFF
f****FFFFFFFfff
r*rrfFFFFFFffff
f*frffFFFffffff
R*ffrrrfFFFFfff
f*ffffrFFFFFFff
f*fffWWWWWFFFFF
f*ffWWWWWWWWFFF
f**fffWWWWWWWrr
ff****ffffWWfff
fffff***fffffff
fffffff*fffffff
hffffff*RRRRRRR
Mhhffff*fffffff
MhhffffGfffffff
MMhhhffffffffff
```

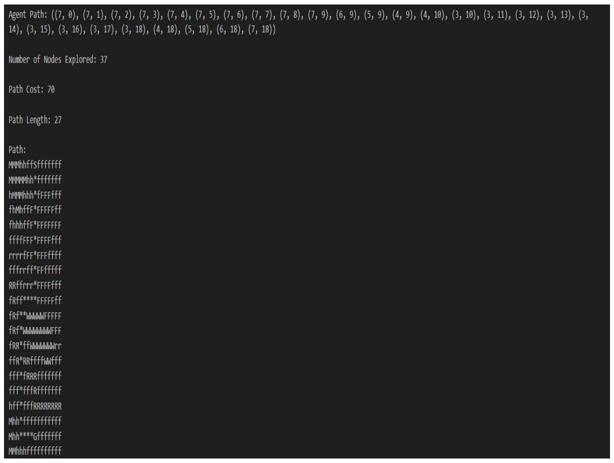
Figure 4: Lowest Cost Search Agent Path



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (6, 3), (5, 3), (5, 4), (4, 4), (4, 5), (3, 5), (2, 5), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11), (1, 12),
(2, 12), (2, 13), (3, 13), (4, 13), (5, 13), (5, 14), (6, 14), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18))

Number of Nodes Explored: 241

Path Cost: 58

Path Length: 31

Path:
$$$$$$$S$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$***$$$$$$$
$$$$**$$$$$$$$$
$****$$$$$$$$$$
$*$$$$$$$$$$$$$
$*$$$$$$$$$$$$$
$*$$$$$$$$$$$$$
$*$$$$$$$$$$$$$
$*$$$WWWWW$$$$$
$*$$WWWWWWWW$$$
$**$$$WWWWWWW$$
$$****$$$$WW$$$
$$$$$***$$$$$$$
$$$$$$$*$$ff$$$
$$$$$$$*$$$R$$$
M$$$$$$*$ffffff
Mhh$$$fGfffffff
MMhhhffffffffff
```

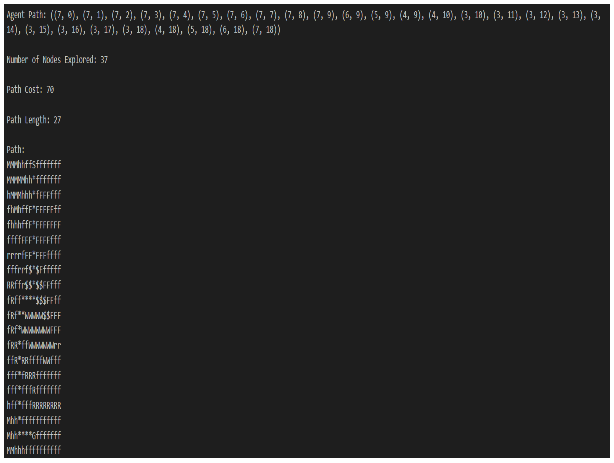
Figure 5: Lowest Cost Search Agent Exploration

## 3.3 Greedy Best First Search Path & Exploration



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (3,
14), (3, 15), (3, 16), (3, 17), (3, 18), (4, 18), (5, 18), (6, 18), (7, 18))

Number of Nodes Explored: 37

Path Cost: 70

Path Length: 27

Path:
MMMhhffSfffffff
MMMMMhh*fffffff
hMMMhhh*fFFFfff
fhMhffF*FFFFFff
fhhhffF*FFFFFFF
ffffFFF*FFFFfff
rrrrfFF*FFFffff
fffrrff*FFfffff
RRffrrr*FFFFfff
fRff****FFFFFff
fRf**WWWWWFFFFF
fRf*WWWWWWWWFFF
fRR*ffWWWWWWWrr
ffR*RRffffWWfff
fff*fRRRfffffff
fff*fffRfffffff
hff*fffRRRRRRRR
Mhh*fffffffffff
Mhh****Gfffffff
MMhhhffffffffff
```

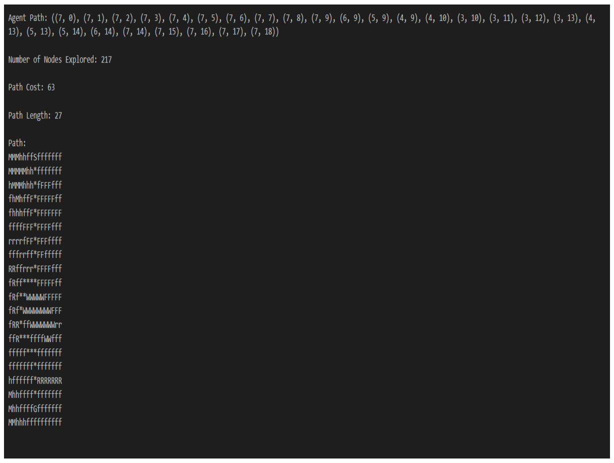
Figure 6: Greedy Best First Agent Path



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (3,
14), (3, 15), (3, 16), (3, 17), (3, 18), (4, 18), (5, 18), (6, 18), (7, 18))

Number of Nodes Explored: 37

Path Cost: 70

Path Length: 27

Path:
MMMhhffSfffffff
MMMMMhh*fffffff
hMMMhhh*fFFFfff
fhMhffF*FFFFFff
fhhhffF*FFFFFFF
ffffFFF*FFFFfff
rrrrfFF*FFFffff
fffrrf$*$Ffffff
RRffr$$*$$FFfff
fRff****$$$FFff
fRf**WWWWW$$FFF
fRf*WWWWWWWWFFF
fRR*ffWWWWWWWrr
ffR*RRffffWWfff
fff*fRRRfffffff
fff*fffRfffffff
hff*fffRRRRRRRR
Mhh*fffffffffff
Mhh****Gfffffff
MMhhhffffffffff
```

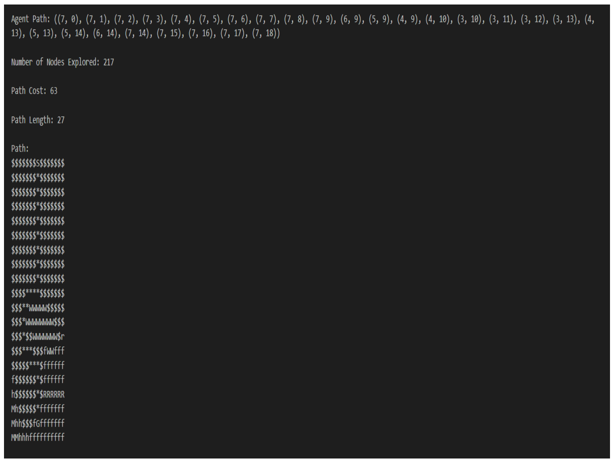
Figure 7: Greedy Best First Agent Exploration

## 3.4 A* Search Path & Exploration



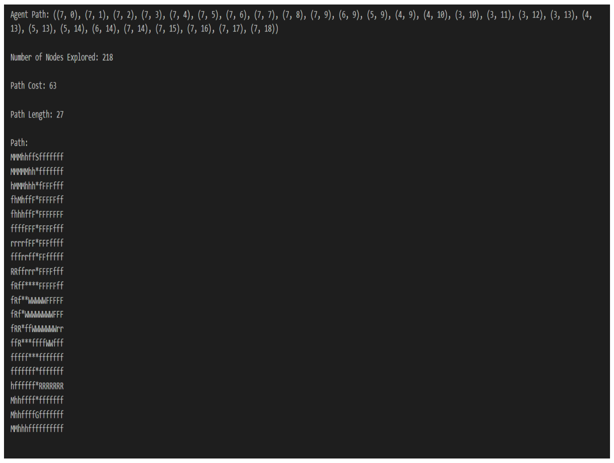
```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (4,
13), (5, 13), (5, 14), (6, 14), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18))

Number of Nodes Explored: 217

Path Cost: 63

Path Length: 27

Path:
MMMhhffSfffffff
MMMMMhh*fffffff
hMMMhhh*fFFFfff
fhMhffF*FFFFFff
fhhhffF*FFFFFFF
ffffFFF*FFFFfff
rrrrfFF*FFFffff
fffrrff*FFfffff
RRffrrr*FFFFfff
fRff****FFFFFff
fRf**WWWWWFFFFF
fRf*WWWWWWWWFFF
fRR*ffWWWWWWWrr
ffR***ffffWWfff
fffff***fffffff
fffffff*fffffff
hffffff*RRRRRRR
Mhhffff*fffffff
MhhffffGfffffff
MMhhhffffffffff
```

**Figure 8: A* Agent Path With Manhattan Heuristics**



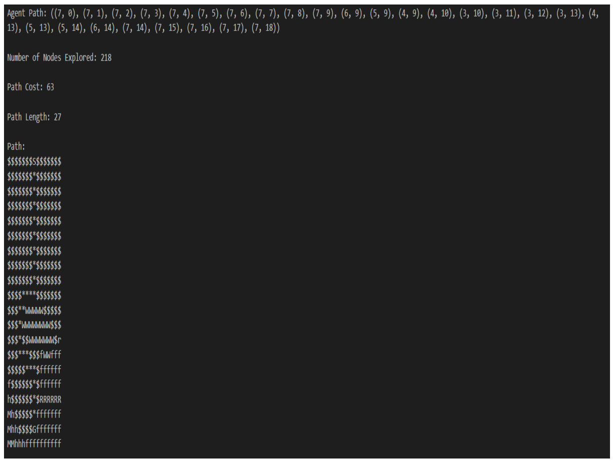
```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (4,
13), (5, 13), (5, 14), (6, 14), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18))

Number of Nodes Explored: 217

Path Cost: 63

Path Length: 27

Path:
$$$$$$$S$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$****$$$$$$$
$$$**WWWWW$$$$$
$$$*WWWWWWWW$$$
$$$*$$WWWWWWW$r
$$$***$$$fWWfff
$$$$$***$ffffff
f$$$$$$*$ffffff
h$$$$$$*$RRRRRR
Mh$$$$$*fffffff
Mhh$$$fGfffffff
MMhhhffffffffff
```

**Figure 9: A* Agent Exploration With Manhattan Heuristics**



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (4,
13), (5, 13), (5, 14), (6, 14), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18))

Number of Nodes Explored: 218

Path Cost: 63

Path Length: 27

Path:
MMMhhffSfffffff
MMMMMhh*fffffff
hMMMhhh*fFFFfff
fhMhffF*FFFFFff
fhhhffF*FFFFFFF
ffffFFF*FFFFfff
rrrrfFF*FFFffff
fffrrff*FFfffff
RRffrrr*FFFFfff
fRff****FFFFFff
fRf**WWWWWFFFFF
fRf*WWWWWWWWFFF
fRR*ffWWWWWWWrr
ffR***ffffWWfff
fffff***fffffff
fffffff*fffffff
hffffff*RRRRRRR
Mhhffff*fffffff
MhhffffGfffffff
MMhhhffffffffff
```

**Figure 10: A* Agent Path With Euclidean Heuristics**



```
Agent Path: ((7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (6, 9), (5, 9), (4, 9), (4, 10), (3, 10), (3, 11), (3, 12), (3, 13), (4,
13), (5, 13), (5, 14), (6, 14), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18))

Number of Nodes Explored: 218

Path Cost: 63

Path Length: 27

Path:
$$$$$$$S$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$$$$*$$$$$$$
$$$$****$$$$$$$
$$$**WWWWW$$$$$
$$$*WWWWWWWW$$$
$$$*$$WWWWWWW$r
$$$***$$$fWWfff
$$$$$***$ffffff
f$$$$$$*$ffffff
h$$$$$$*$RRRRRR
Mh$$$$$*fffffff
Mhh$$$$Gfffffff
MMhhhffffffffff
```

**Figure 11: A* Agent Exploration With Euclidean Heuristics**

## 4 CONCLUSION

In conclusion, this project successfully applied AI pathfinding techniques derived from the field of Artificial Intelligence, with specific reference to the principles taught in the CS-470 course. The main goal was to import environmental data from a designated data file and create an AI agent capable of navigating the map from an initial starting point to a specified target destination. To achieve this objective, a variety of searching algorithms were utilized. Breadth First Search, Lowest Cost Search, Greedy Best First, and A* (Utilizing two different heuristics, manhatten and euclidean). The chosen methodology yielded highly favorable outcomes, resulting in a strong level of confidence in the accuracy and reliability of the obtained results. This project highlights the effectiveness and practicality of AI pathfinding techniques in addressing real-world challenges and underscores the value of incorporating AI principles into problem-solving endeavors.