University of Puerto Rico
Department of Electrical and Computer Engineering
ICOM5015 Artificial Intelligence

Search Algorithms

Group C

Manuel Alejandro Umaña Rodriguez -Undergraduate Computer Engineering
Dahyna Martínez Pérez -Undergraduate Computer Engineering
Jan Luis Pérez de Jesús -Undergraduate Computer Engineering
Christopher Hans Mayens Matías -Undergraduate Computer Engineering

For: Profesor José Fernando Vega Riveros

Date: March 19, 2025

Agenda

01 Purpose of experiment

02 Hypothesis

03 Concepts

04 Experiments set up

05 Information

06 Conclusion

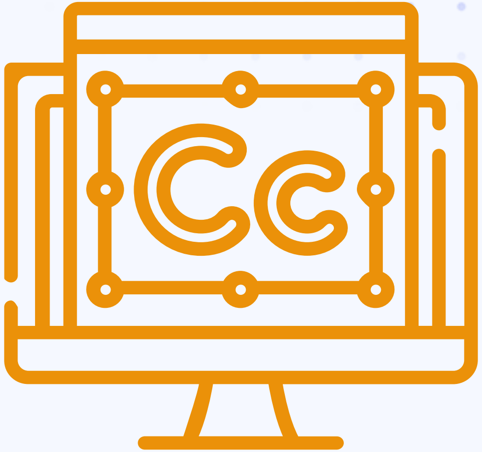
07 Credits & References

01
Purpose of experiment

The experiments were carried out to determine two major things:

- How does a well developed scenario can help identify which algorithm is required to solve a problem.

- How effective are different algorithms at solving the problem statements based on the constraints of the problem and any “hint” that may be given.

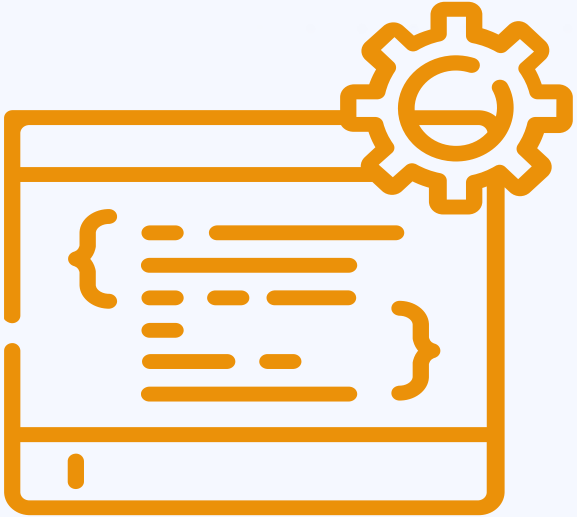


02

Key questions and hypothesis

Does the design of a problem affect the selection of a search algorithm?

- How do obstacles affect the performance of a search algorithm?
- How do the constraints of a problem affect the choice of search algorithms?
- Why do regular people struggle in solving complex state space problems while AI resolves this problems efficiently?



Hypothesis: If the design specifications and constraints of a problem are properly set, the type of search algorithm required to solve the problem can be chosen with more precision.

03

Concepts

Experimental concepts

Platform

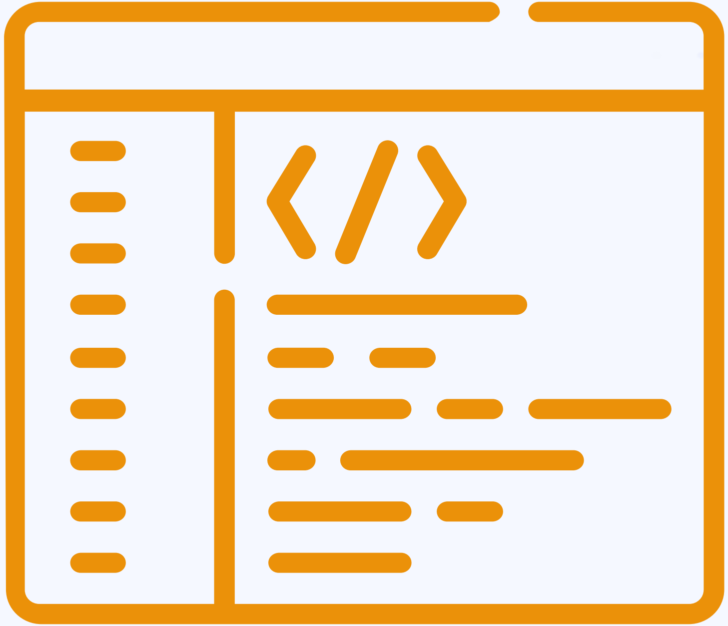
Python: Programming language of high level. Emphasizes code readability with the use of indentation.

Subject

The agent's ability to efficiently find its path, overcoming obstacles and restrictions.

Measure

The amount of steps it takes the agent to reach the goal state form the initial state..

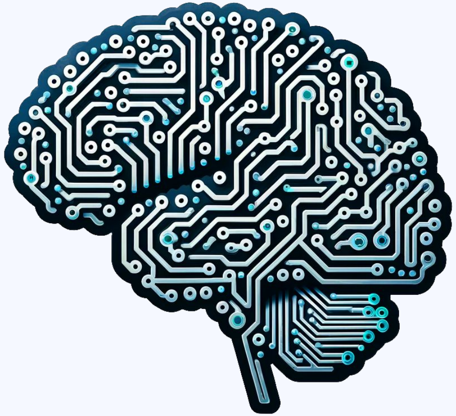


04 Experiments set up.

Tools and Resources Utilized

The following tools and resources were used:

- **Aimacode Repository:** The core repository containing foundational code and algorithms for both problems.
- **For Problem 3.7:** The A* and Greedy BFS algorithms were used for testing, as well as the Straight-Line Distance heuristic.
- **For Problem 3.9:** The BFS and DFS were used for testing with the solution being found with BFS as DFS could not find the solution.



Method for comparing

Criteria Used for Comparison:

For **Problem 3.7**, the criteria used for comparison was:

- **Shortest Path Between Algorithms:** The shortest path was determined by counting the number of states (or steps) visited from the starting position to the goal state.

For **Problem 3.9**,
Comparison was:

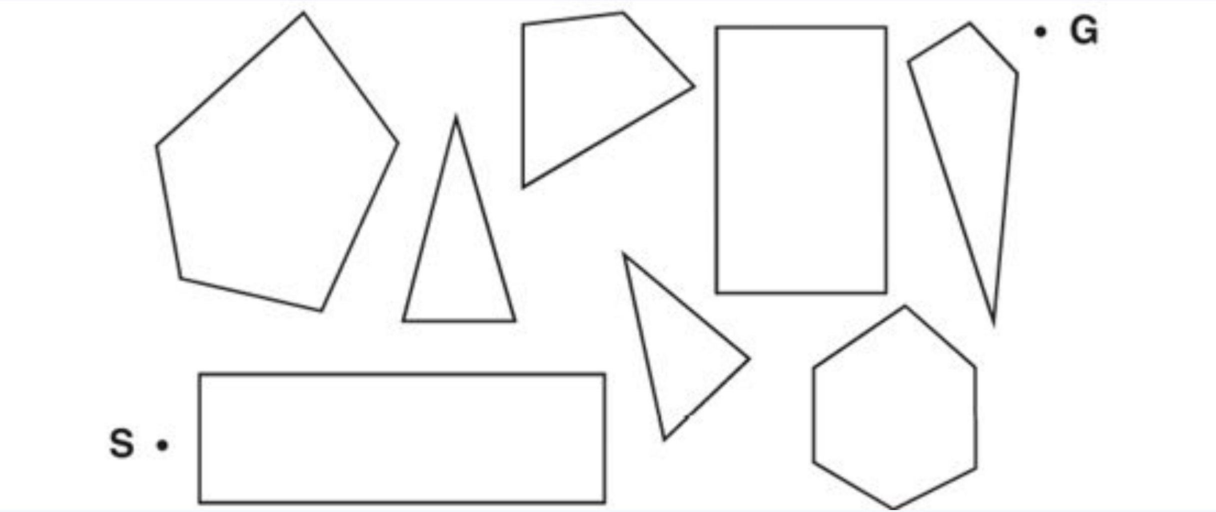
- **Amount of steps to find solution**: The amount of steps the algorithms took determined the efficiency when finding the goal state.

05

Information.

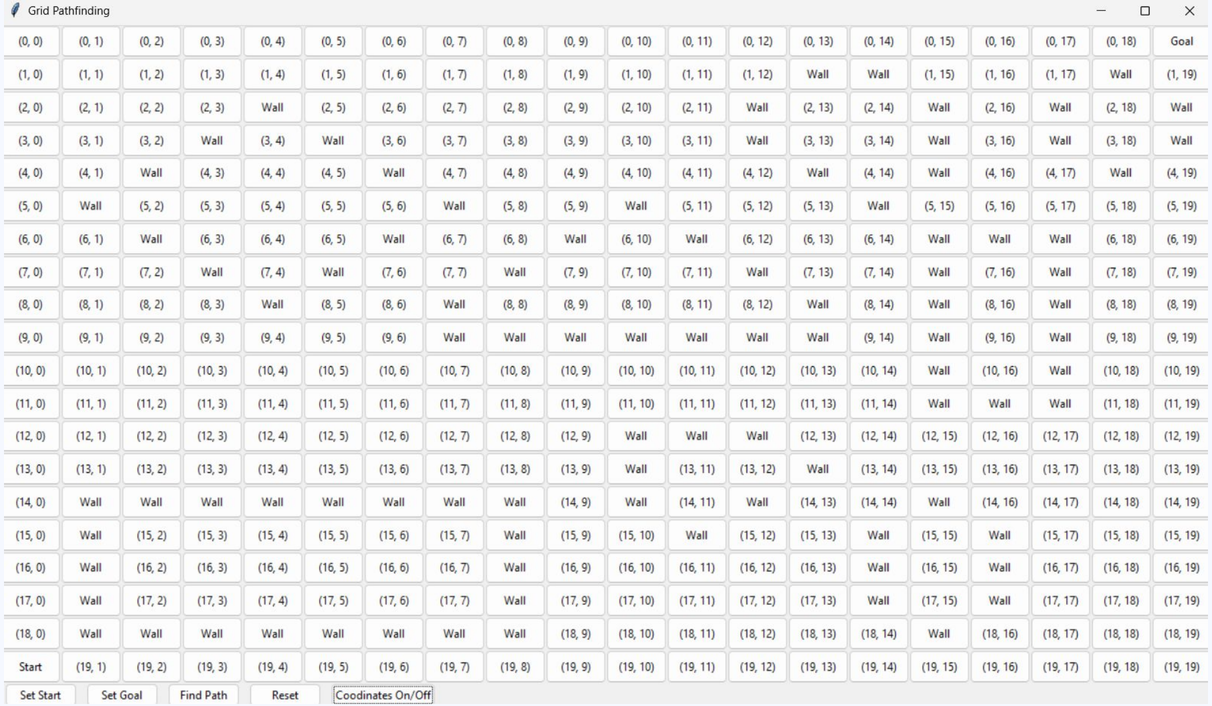
Figures

Figure 1. Figure 3.13 from “A scene with polygonal obstacles. S and G are the start and goal states.”



Figures

Figure 2. Coordinates recreating the environment in Figure 1



(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	(0, 7)	(0, 8)	(0, 9)	(0, 10)	(0, 11)	(0, 12)	(0, 13)	(0, 14)	(0, 15)	(0, 16)	(0, 17)	(0, 18)	Goal
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)	(1, 7)	(1, 8)	(1, 9)	(1, 10)	(1, 11)	(1, 12)	Wall	Wall	(1, 15)	(1, 16)	(1, 17)	Wall	(1, 19)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	Wall	(2, 5)	(2, 6)	(2, 7)	(2, 8)	(2, 9)	(2, 10)	(2, 11)	Wall	(2, 13)	(2, 14)	Wall	(2, 16)	Wall	(2, 18)	Wall
(3, 0)	(3, 1)	(3, 2)	Wall	(3, 4)	Wall	(3, 6)	(3, 7)	(3, 8)	(3, 9)	(3, 10)	(3, 11)	Wall	(3, 13)	(3, 14)	Wall	(3, 16)	Wall	(3, 18)	Wall
(4, 0)	(4, 1)	Wall	(4, 3)	(4, 4)	(4, 5)	Wall	(4, 7)	(4, 8)	(4, 9)	(4, 10)	(4, 11)	(4, 12)	Wall	(4, 14)	Wall	(4, 16)	(4, 17)	Wall	(4, 19)
(5, 0)	Wall	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)	Wall	(5, 8)	(5, 9)	Wall	(5, 11)	(5, 12)	(5, 13)	Wall	(5, 15)	(5, 16)	(5, 17)	(5, 18)	(5, 19)
(6, 0)	(6, 1)	Wall	(6, 3)	(6, 4)	(6, 5)	Wall	(6, 7)	(6, 8)	Wall	(6, 10)	Wall	(6, 12)	(6, 13)	(6, 14)	Wall	Wall	Wall	(6, 18)	(6, 19)
(7, 0)	(7, 1)	(7, 2)	Wall	(7, 4)	Wall	(7, 6)	(7, 7)	Wall	(7, 9)	(7, 10)	(7, 11)	Wall	(7, 13)	(7, 14)	Wall	(7, 16)	Wall	(7, 18)	(7, 19)
(8, 0)	(8, 1)	(8, 2)	(8, 3)	Wall	(8, 5)	(8, 6)	Wall	(8, 8)	(8, 9)	(8, 10)	(8, 11)	(8, 12)	Wall	(8, 14)	Wall	(8, 16)	Wall	(8, 18)	(8, 19)
(9, 0)	(9, 1)	(9, 2)	(9, 3)	(9, 4)	(9, 5)	(9, 6)	Wall	Wall	Wall	Wall	Wall	Wall	Wall	(9, 14)	Wall	(9, 16)	Wall	(9, 18)	(9, 19)
(10, 0)	(10, 1)	(10, 2)	(10, 3)	(10, 4)	(10, 5)	(10, 6)	(10, 7)	(10, 8)	(10, 9)	(10, 10)	(10, 11)	(10, 12)	(10, 13)	(10, 14)	Wall	(10, 16)	Wall	(10, 18)	(10, 19)
(11, 0)	(11, 1)	(11, 2)	(11, 3)	(11, 4)	(11, 5)	(11, 6)	(11, 7)	(11, 8)	(11, 9)	(11, 10)	(11, 11)	(11, 12)	(11, 13)	(11, 14)	Wall	Wall	Wall	(11, 18)	(11, 19)
(12, 0)	(12, 1)	(12, 2)	(12, 3)	(12, 4)	(12, 5)	(12, 6)	(12, 7)	(12, 8)	(12, 9)	Wall	Wall	Wall	(12, 13)	(12, 14)	(12, 15)	(12, 16)	(12, 17)	(12, 18)	(12, 19)
(13, 0)	(13, 1)	(13, 2)	(13, 3)	(13, 4)	(13, 5)	(13, 6)	(13, 7)	(13, 8)	(13, 9)	Wall	(13, 11)	(13, 12)	Wall	(13, 14)	(13, 15)	(13, 16)	(13, 17)	(13, 18)	(13, 19)
(14, 0)	Wall	Wall	Wall	Wall	Wall	Wall	Wall	Wall	(14, 9)	Wall	(14, 11)	Wall	(14, 13)	(14, 14)	Wall	(14, 16)	(14, 17)	(14, 18)	(14, 19)
(15, 0)	Wall	(15, 2)	(15, 3)	(15, 4)	(15, 5)	(15, 6)	(15, 7)	Wall	(15, 9)	(15, 10)	Wall	(15, 12)	(15, 13)	Wall	(15, 15)	Wall	(15, 17)	(15, 18)	(15, 19)
(16, 0)	Wall	(16, 2)	(16, 3)	(16, 4)	(16, 5)	(16, 6)	(16, 7)	Wall	(16, 9)	(16, 10)	(16, 11)	(16, 12)	(16, 13)	Wall	(16, 15)	Wall	(16, 17)	(16, 18)	(16, 19)
(17, 0)	Wall	(17, 2)	(17, 3)	(17, 4)	(17, 5)	(17, 6)	(17, 7)	Wall	(17, 9)	(17, 10)	(17, 11)	(17, 12)	(17, 13)	Wall	(17, 15)	Wall	(17, 17)	(17, 18)	(17, 19)
(18, 0)	Wall	Wall	Wall	Wall	Wall	Wall	Wall	Wall	(18, 9)	(18, 10)	(18, 11)	(18, 12)	(18, 13)	(18, 14)	Wall	(18, 16)	(18, 17)	(18, 18)	(18, 19)
Start	(19, 1)	(19, 2)	(19, 3)	(19, 4)	(19, 5)	(19, 6)	(19, 7)	(19, 8)	(19, 9)	(19, 10)	(19, 11)	(19, 12)	(19, 13)	(19, 14)	(19, 15)	(19, 16)	(19, 17)	(19, 18)	(19, 19)

Set Start | Set Goal | Find Path | Reset | Coodinates On/Off

Figures

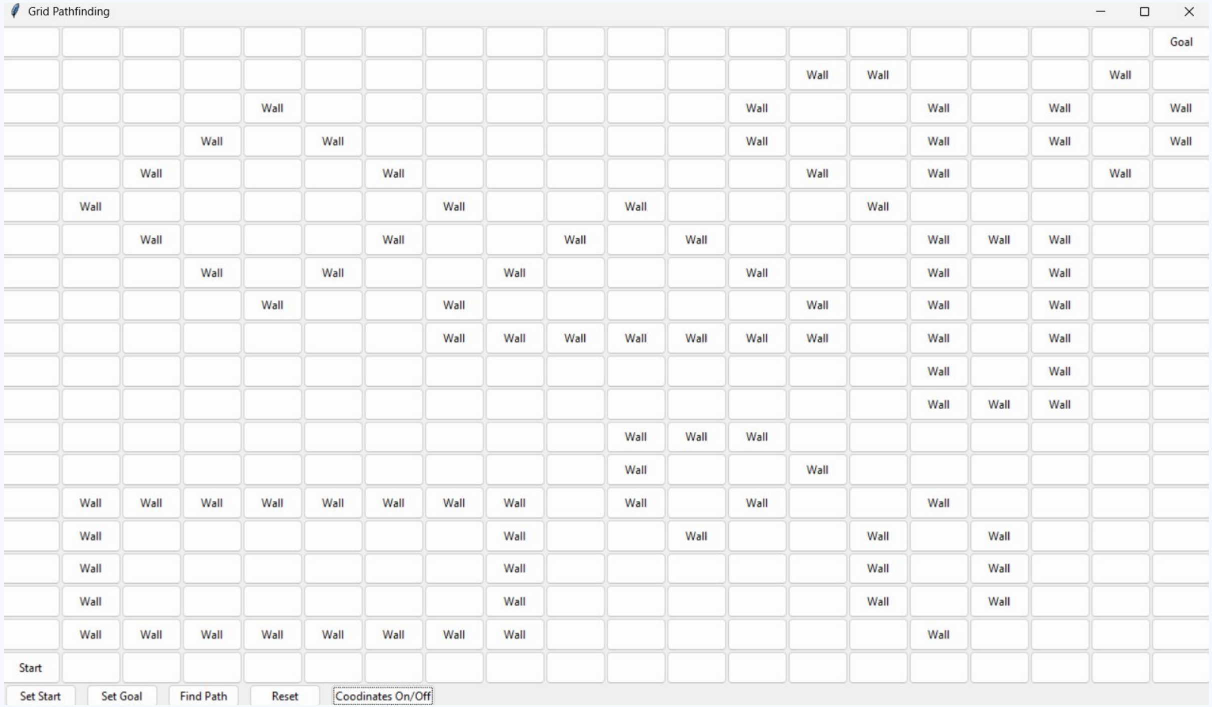
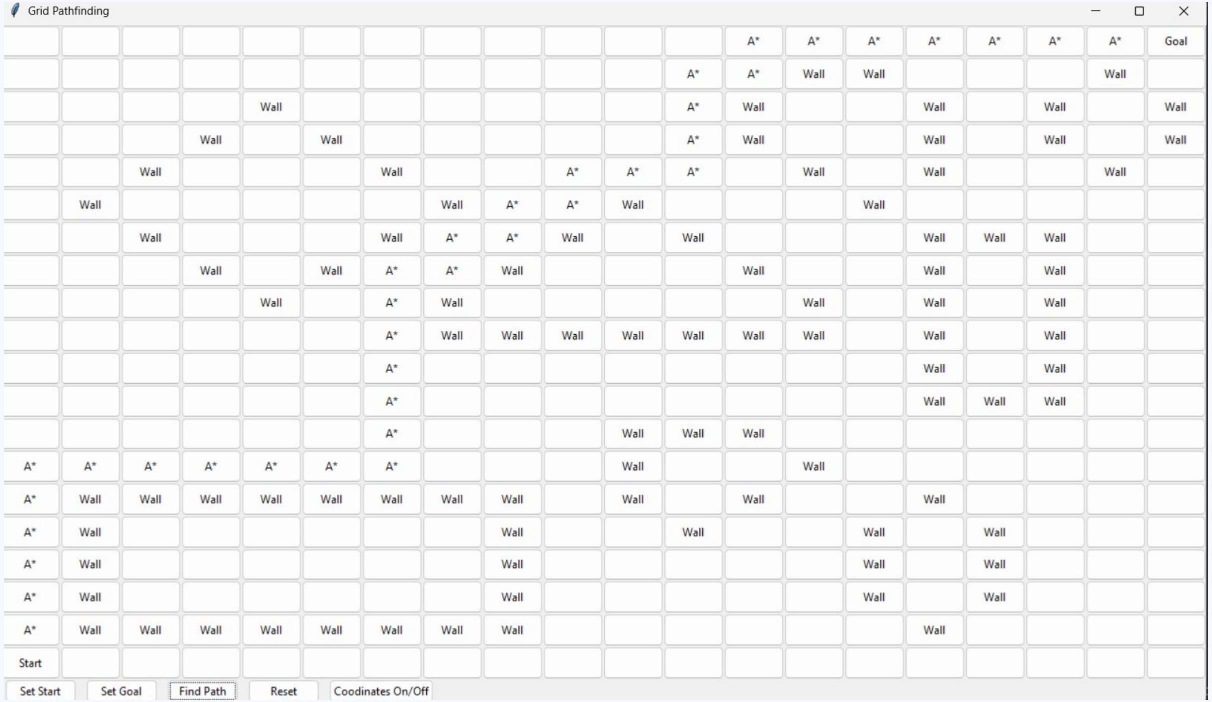


Figure 3. Case 1, Coordinates recreating the environment in Figure 1 without numbers

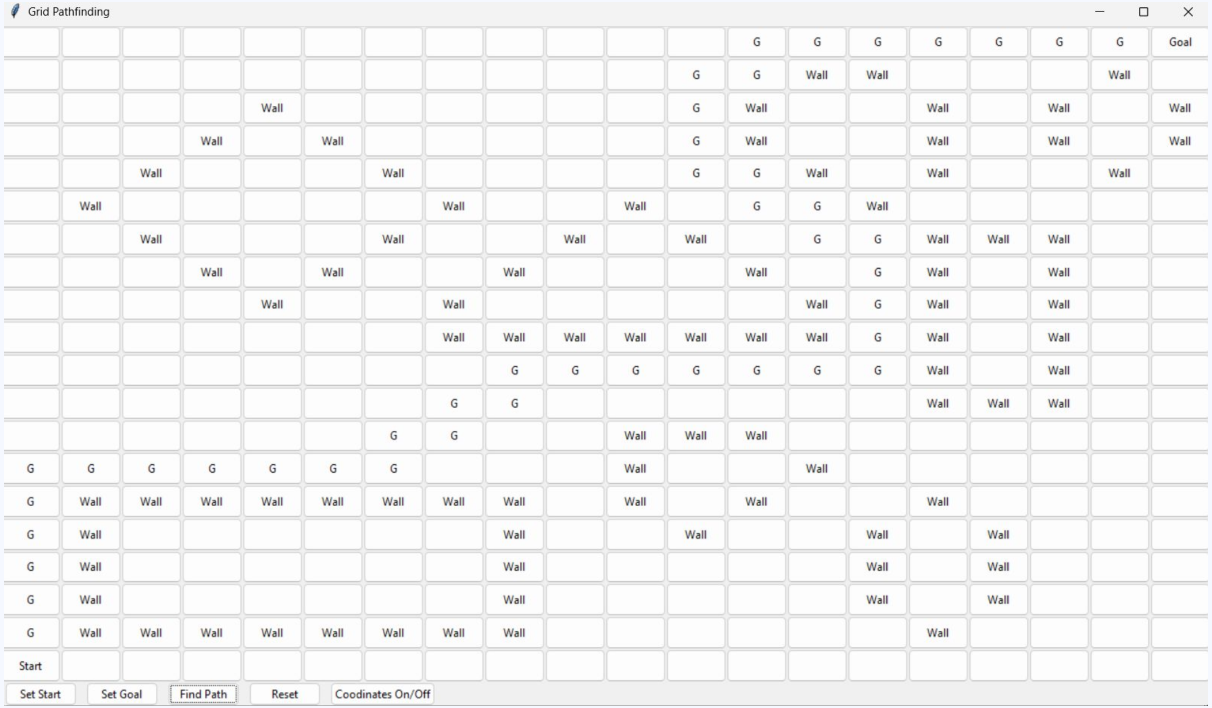
Figures

Figure 4. Solution of shortest path between start and goal utilizing A* search algorithm for case 1



Figures

Figure 5. Solution of shortest path between start and goal utilizing Greedy Best-First Search algorithm for case 1



Figures

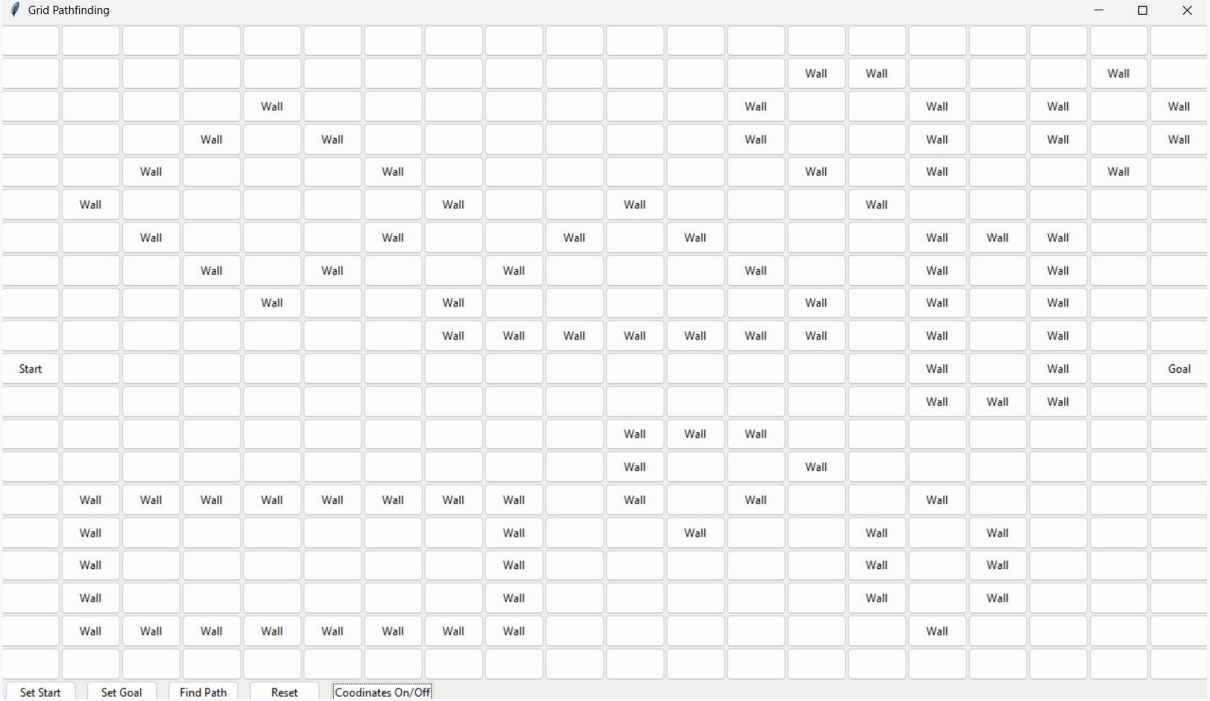
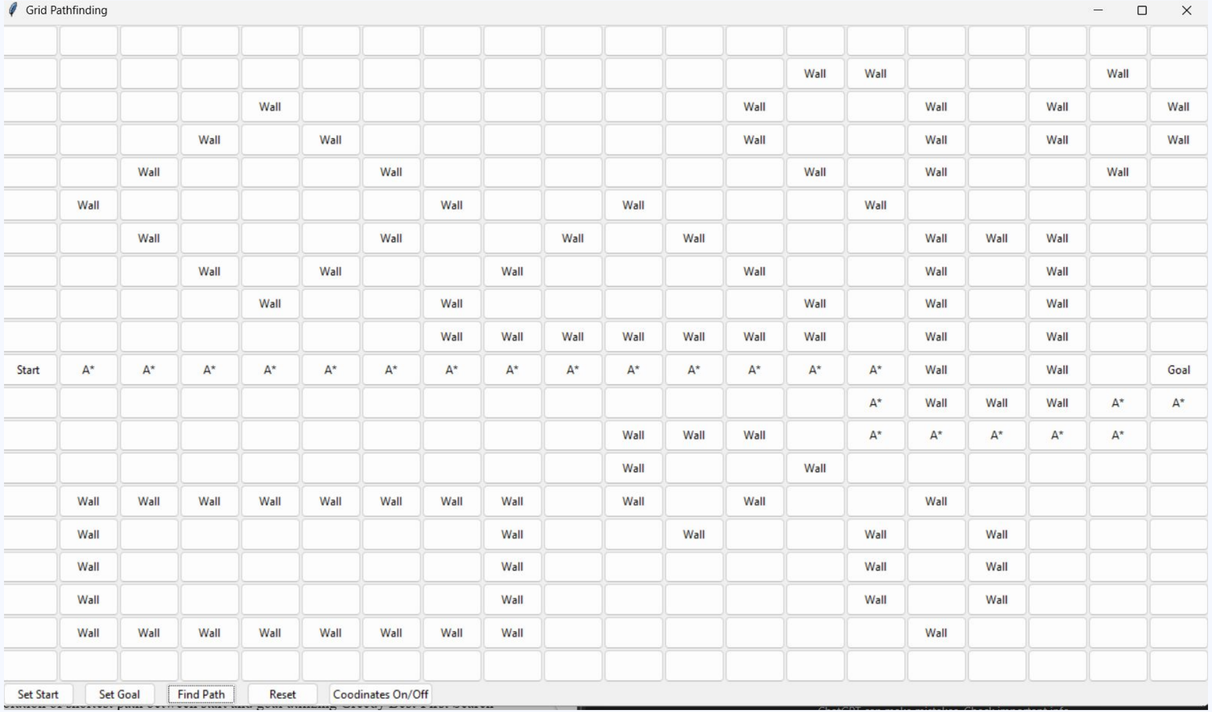


Figure 6. Case 2, Coordinates recreating a somewhat ideal environment without numbers

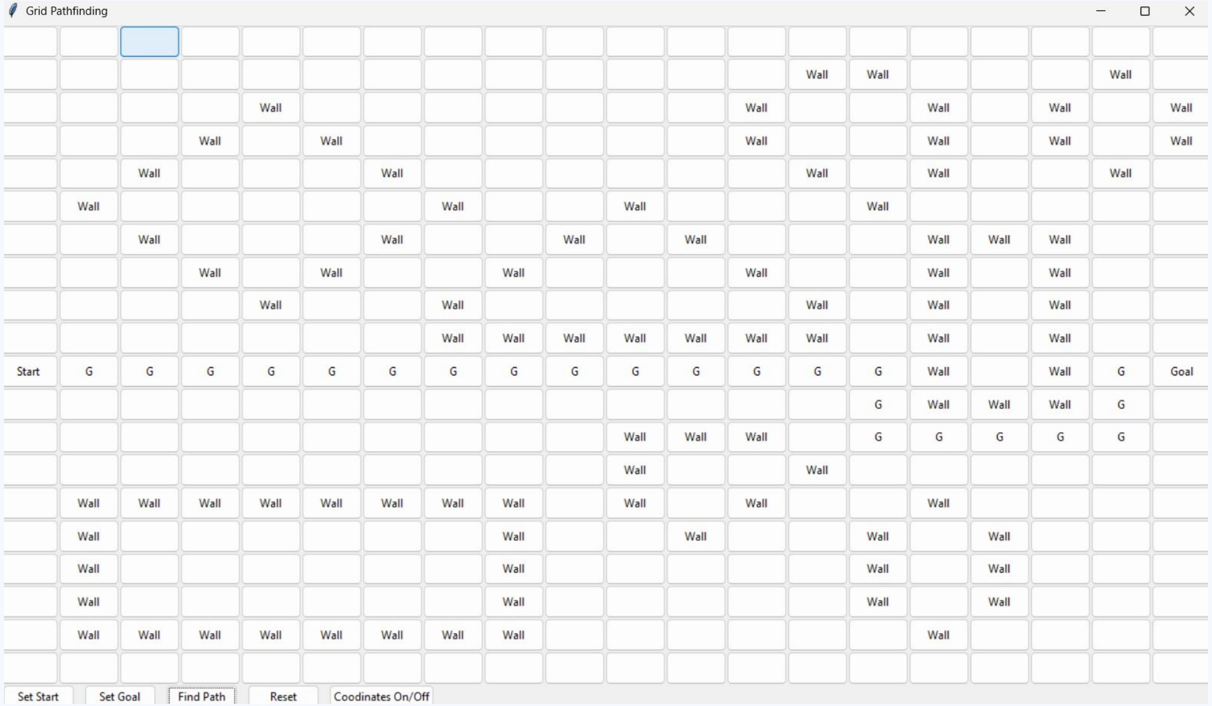
Figures

Figure 7. Solution of shortest path between start and goal utilizing A* search algorithm for case 2



Figures

Figure 8. Solution of shortest path between start and goal utilizing Greedy Best-First Search algorithm for case 2



Figures

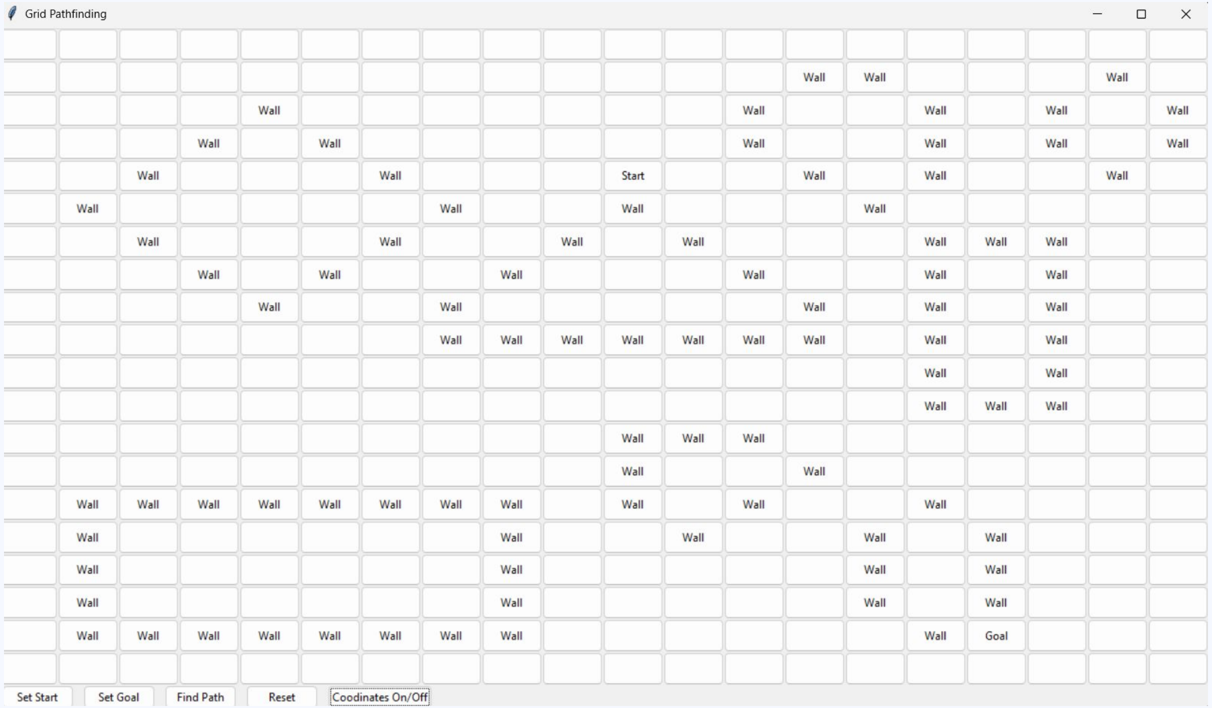
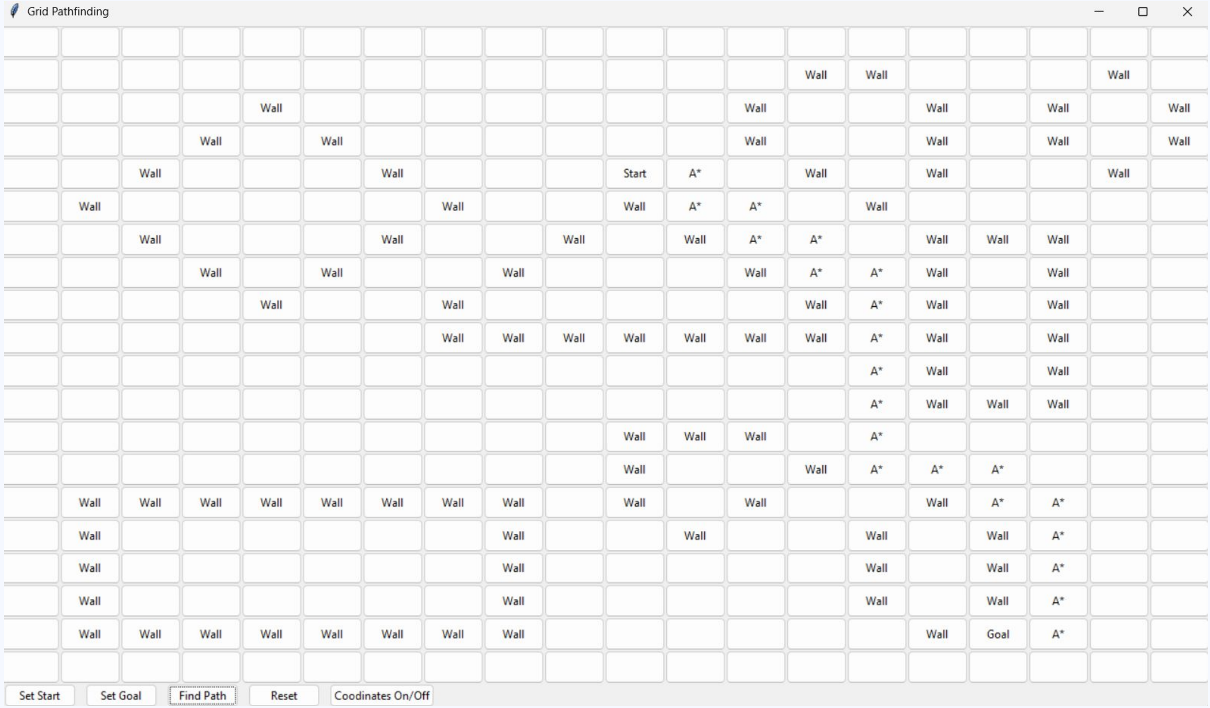


Figure 9. Case 3, coordinates recreating a tight corner environment without numbers

Figures

Figure 10. Solution of shortest path between start and goal utilizing A* search algorithm for case 3



Figures

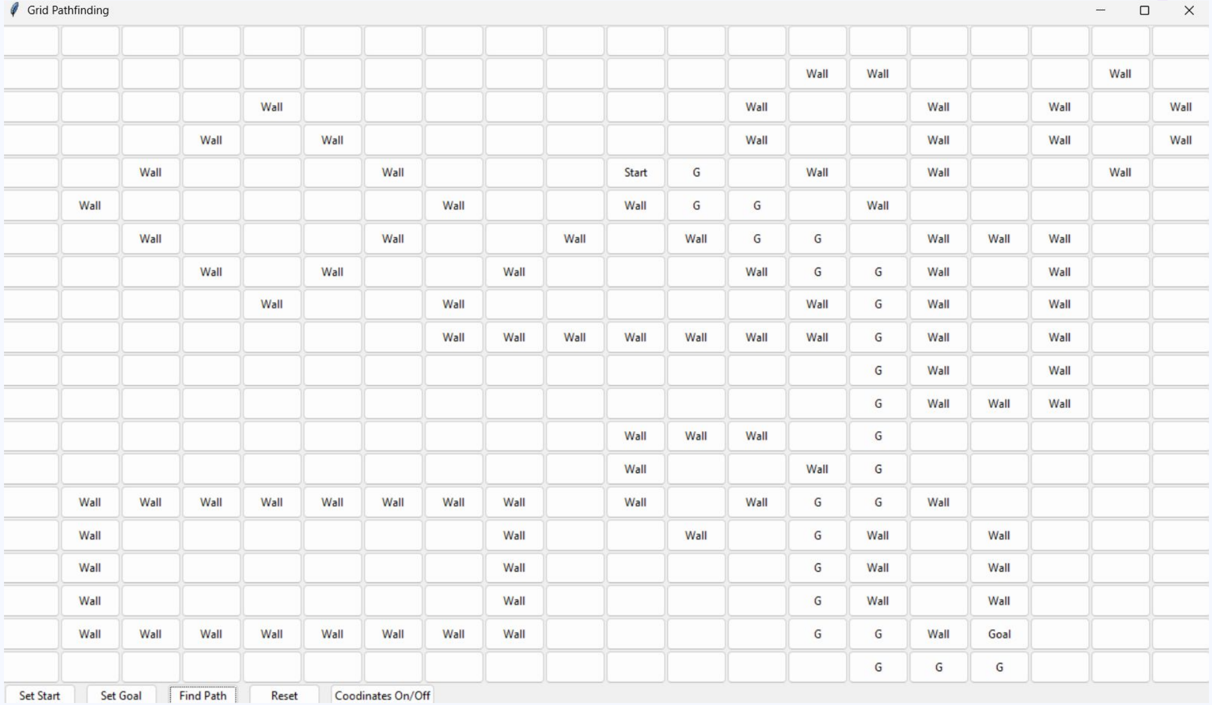
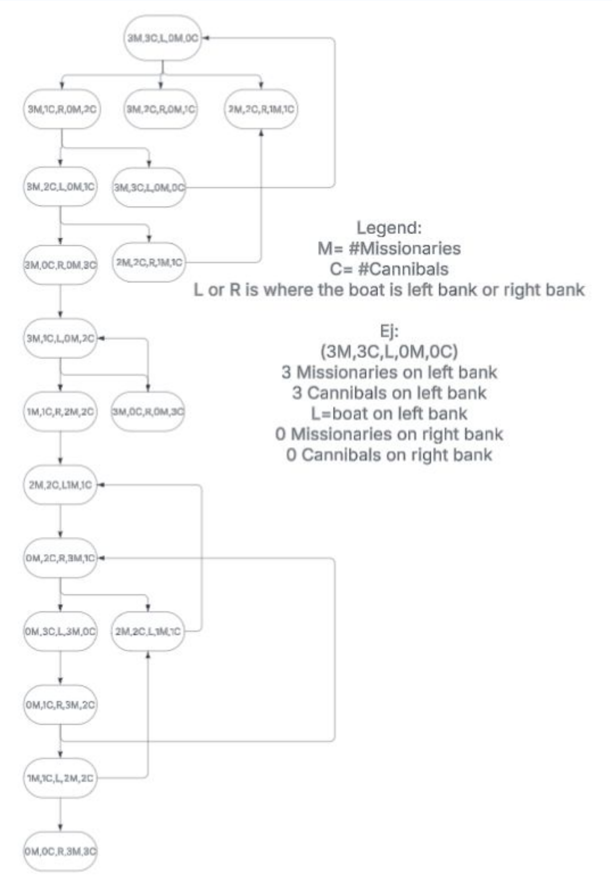


Figure 11. Solution of shortest path between start and goal utilizing Greedy Best-First Search algorithm for case 3

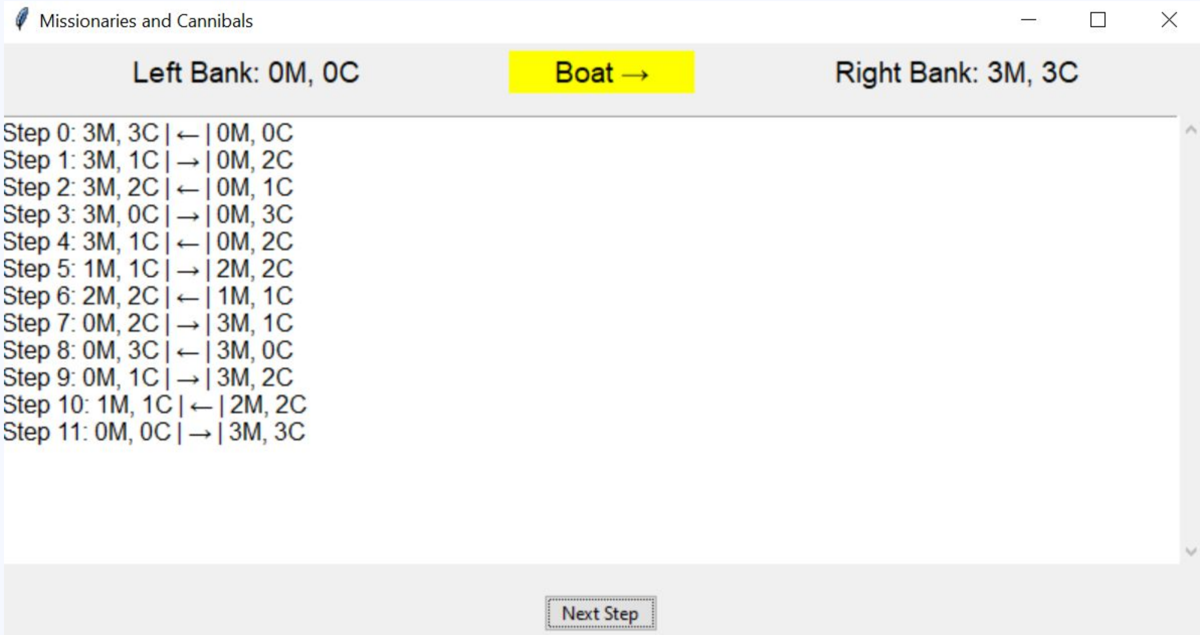
Figures

Figure 12: Complete state space for valid moves in missionaries and cannibals problem.



Figures

Figure 13: Optimal solution found for the missionaries and cannibals problem by BFS



06

Conclusion and Lessons Learned.

Conclusions and Lessons Learned

01 — Heuristic function utility

Informed search algorithms require information about the environment. The Heuristic function estimates the cost from the current position to the goal. This is an efficient way for a search algorithm to traverse from point A to point B, but having additional information like the cost from start to the current position significantly reduces the amount of mistakes the agent can make.

02 Speed VS Optimality

Having additional information makes the code optimal but it also adds complexity. An increase in complexity decreases the speed because it makes the agent analyze more data before making a decision. Understanding which search algorithm to utilize in a specific environment is key to finding the best solution.

Conclusions and Lessons Learned

03 — **Problem formulation**

By formulating the problem definition correctly it is possible to find the most appropriate uninformed algorithm for the task at hand.

04 — **Choosing the best uninformed algorithm**

Based on the provided information some problems may prefer certain responses depending on the length of the search or the memory constraints. Uninformed algorithms may be more suitable for environments where there are not many variables affecting the agent or are unknown.

07

Credits and References.

References

[1] R. Belwarier, “A* Search Algorithm,” *geeksforgeeks.org*, Jul 30, 2024. [Online]. Available: [A* Search Algorithm - GeeksforGeeks]() .[Accessed Mar 18, 2025]

[2] C.Yang, S. Kumar, and M. Wardhani. “Greedy Best-First Search,” *codeacademy.com*, Dec 4, 2024. [Online]. Available: [AI | Search Algorithms | Greedy Best-First Search | Codecademy]() . [Accessed Mar 18, 2025]

[3] S. Russell, and P. Norvig, “Artificial Intelligence: A Modern Approach,” *aima.cs.berkely.edu*, Aug 22, 2022. [Online]. Available : https://aima.cs.berkeley.edu . [Accessed Mar 18, 2025]

[4] Aimacode, “aimpa-python,” *github.com*, Feb. 1, 2016. [Online]. Available:https://github.com/aimacode/aima-python . [Accessed Mar. 17, 2025].

References

[5] Sonuk, “Uninformed Search Algorithms,” *allinformativepost.com*, Jan. 19,2024 [Online]. Available : https://www.allinformativepost.com/uninformed-search-algorithms/ .[Accessed Mar 18, 2025]

[6] G. Tejashree, “Depth First Search (DFS) for Artificial Intelligence” *geeksforgeeks.org*, May 16, 2024. [Online]. Available:https://www.geeksforgeeks.org/depth-first-search-dfs-for-artificial-intelligence/ . [Accessed Mar. 18, 2025]

[7] S. Amarel, “On the Presentation of Problems of Reasoning about Actions,” *RCA Laboratories Princenton ,N.J.*, Mar. 19, 2025. [Online]. Available:https://cse-robotics.engr.tamu.edu/dshell/cs625/MI3-Ch.10-Amarel.pdf . [Accessed Marc. 18, 2025]