Artificial intelligence Project 1 Report

Chifaa Bouzid

82723

CSC 4301 01

Dr. Tajjeddine Rachidi

February 13, 2022

Teammate: Youssef Gaimes

Table of Contents

Project Overview

This work is the first project of my Artificial Intelligence class. The main tasks are to reproduce the pathfinding project and experiment different search and heuristic strategies. The algorithms that are implemented in this pathfinding project are: A* (implemented by Sebastian Lague), A* with Manhattan Heuristics, A* with Euclidean Heuristics, Depth-First Search, Breath-First Search, and Uniform Cost Search. The time and memory allocated by each algorithm are also computed and all the paths corresponding to each algorithm are colored in the grid. This report serves as an analysis to the algorithms and their comparison.

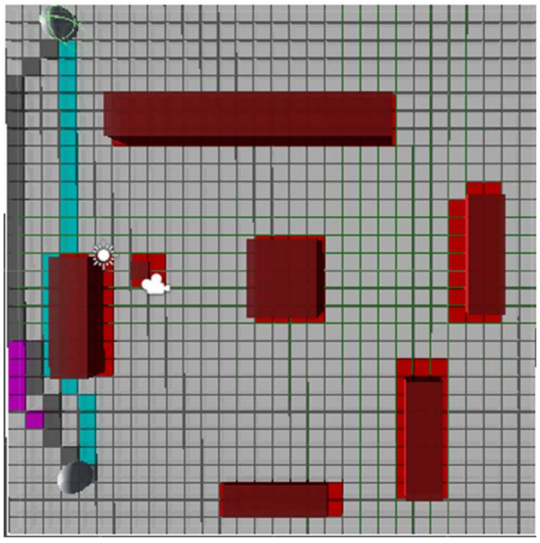
Algorithms Comparison

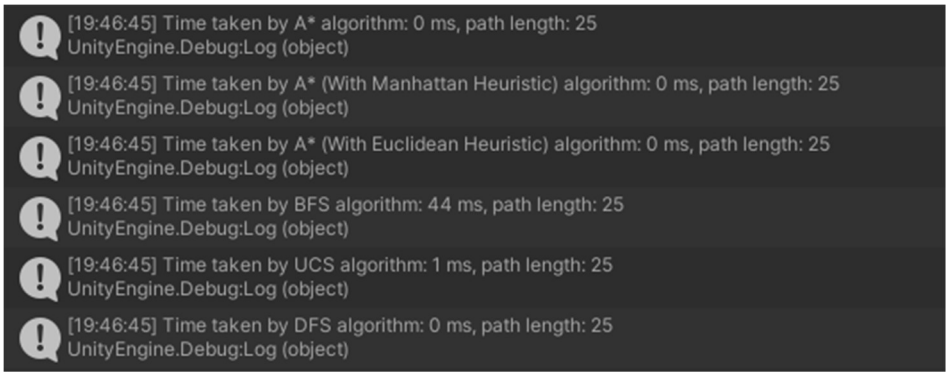
1. Snapshot of the Paths of all the Algorithms

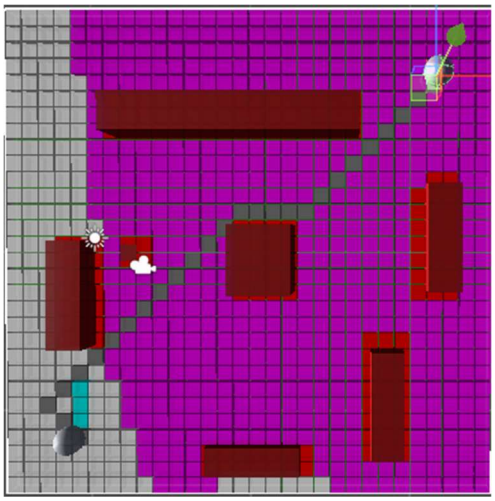
Here are the colors that are used for each algorithm:

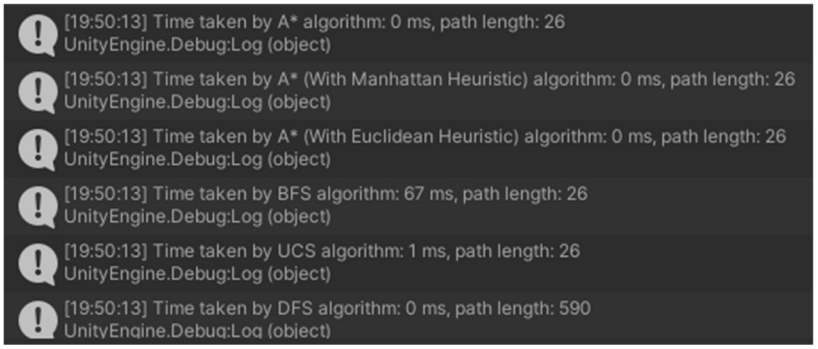
Color	Algorithms
Blue	A*
Black	BFS
Yellow	UCS
Magenta	DFS
Cyan	A* Manhattan
Grey	A* Euclidean

In order to see the difference between the algorithms according to different cases, I have changed the location of the seeker and target from one snapshot to another.





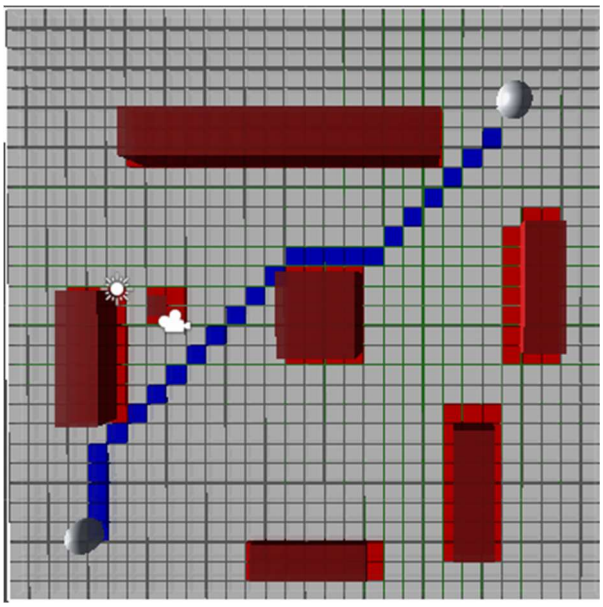




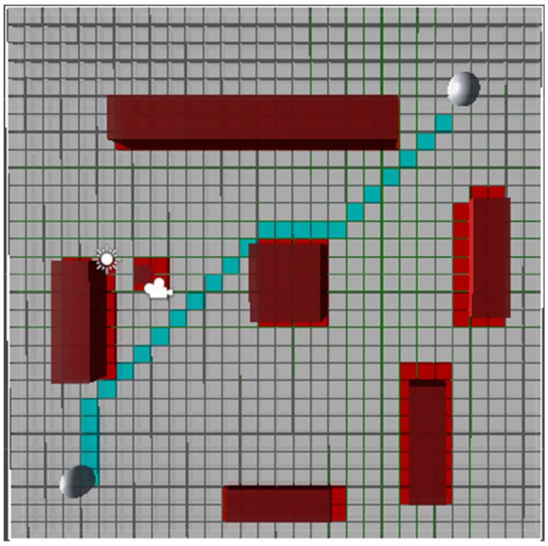
I am aware that colors overlap, thus, it is not very clear, especially in the second case, the difference between each algorithm. However, in the rest of the report, I will take a screenshot of each algorithm. Nevertheless, we can still make some conclusions. DFS performs the worst in the second case as it has a path length of 590 nodes unlike the others that have much less count of nodes (26). I think this explains the fact that DFS keeps seeking for the children of each current node, meaning it keeps following the depth, while the target might be closer than that.

2. A*, A* with Manhattan Heuristics, and A* with Euclidean Heuristics

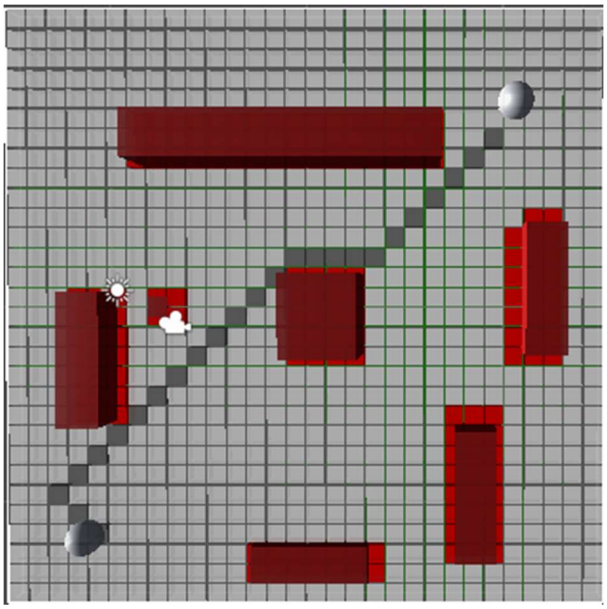
a. A* (Sebastian Lague Implementation)



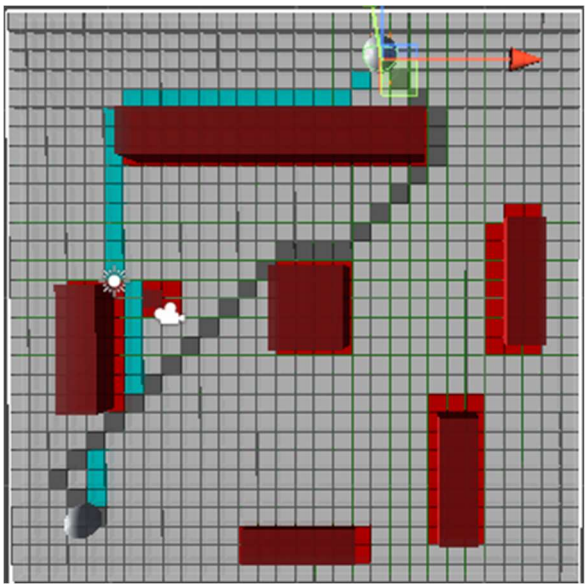
b. A* with Manhattan Heuristics

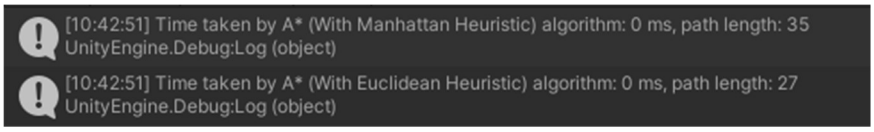


c. A* with Euclidean Heuristics



d. A* with Manhattan VS with Euclidean Heuristics





As it can be seen from the snapshots, A* with Manhattan Heuristics has taken a longer path to reach the goal (37 vs 25). The reason why is that Euclidean heuristics always underestimates the distance left to reach the goal. As we said in a class, it is better to have a heuristic whose estimate is less than the actual distance (optimistic) which is the case for Euclidean heuristics.

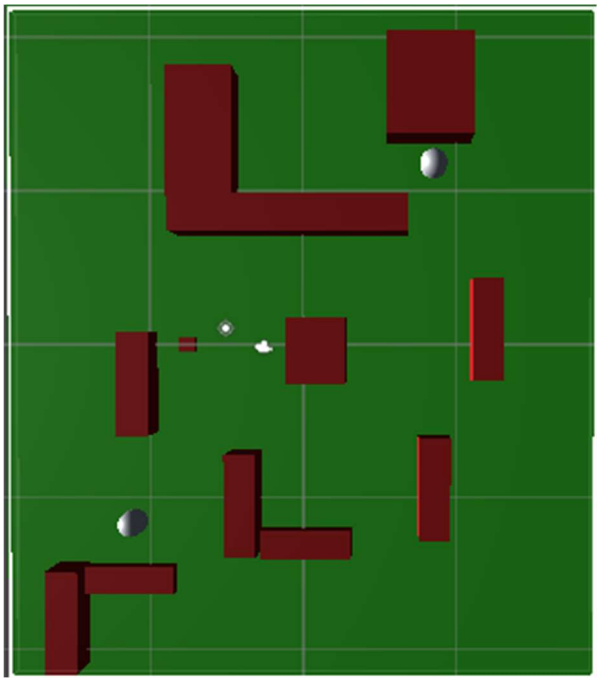
In order to compare between the two heuristics, I printed in the console the distance that each one returns. I have conducted multiple test cases (all along the grid with different target and seeker positions) and in all of them Manhattan heuristics always returns a greater or equal distance to Euclidean. A snapshot of the result can be found below.

So, since $h(s)_{Manhattan} >= h(s)_{Euclidean}$ we can conclude that Euclidean heuristics returns a more optimal path than Manhattan.

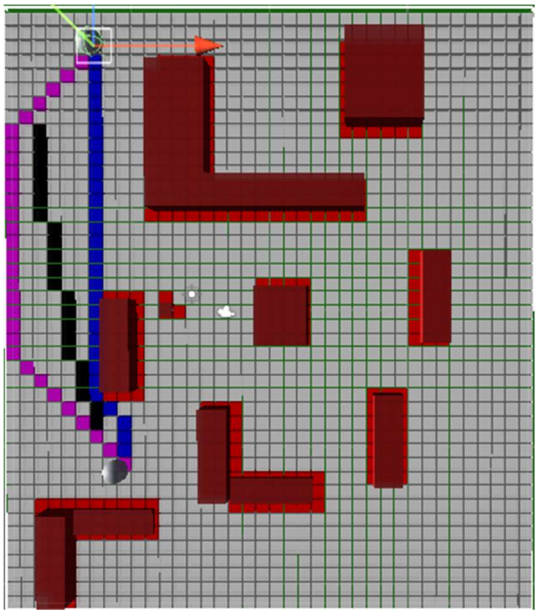
I think the main issue with Manhattan Heuristics is that in this project we are allowing diagonal movements. So, if we are willing to move northwest for example, Manhattan should move West than North instead of diagonally moving.

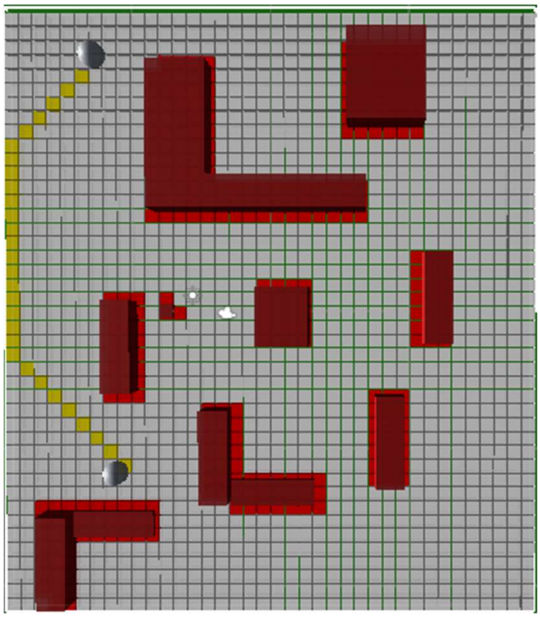
3. Comparison between A*, BFS, UCS, and DFS Based on Time

In order to compare between A*, BFS, UCS, and DFS and actually notice the difference, I have made the plane bigger and added more obstacles as it could be seen below:



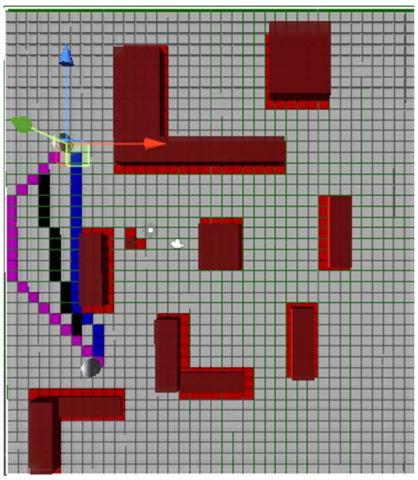
Below are the paths that are done by each of the algorithms:

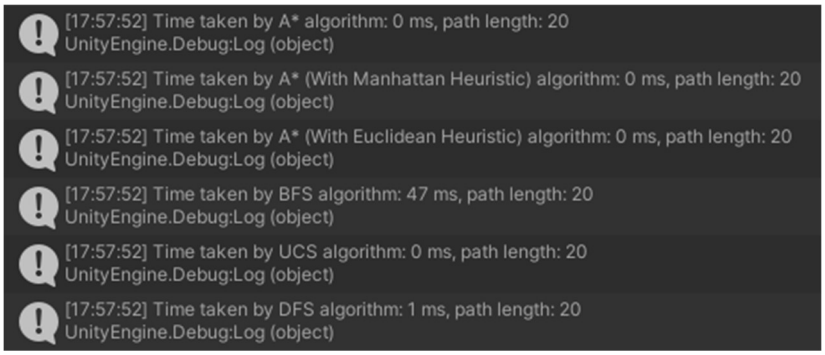


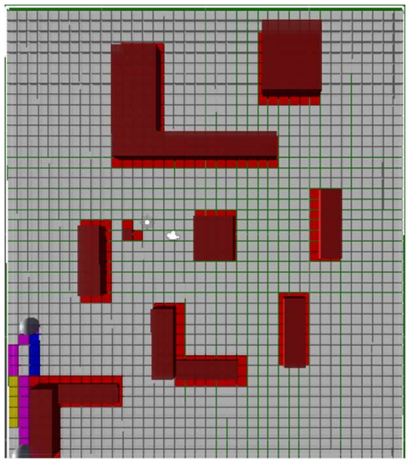


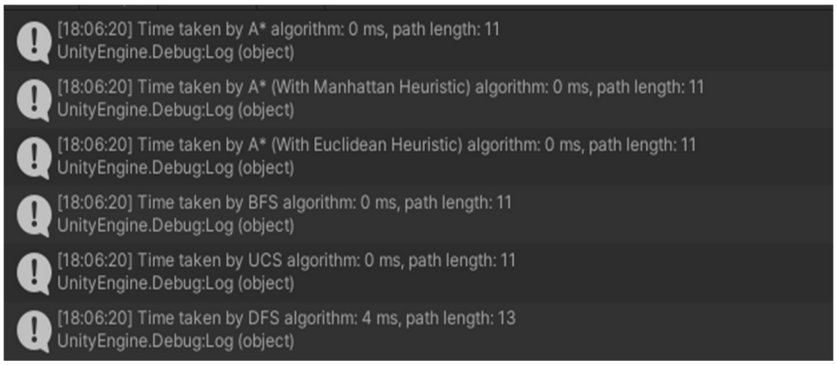
As it could be shown in the right snapshot, UCS has the same path as DFS in this case. Thus, that is why it was hidden in the left snapshot since the two paths overlap.

Now, let's compare the time that each algorithm takes to find the path:



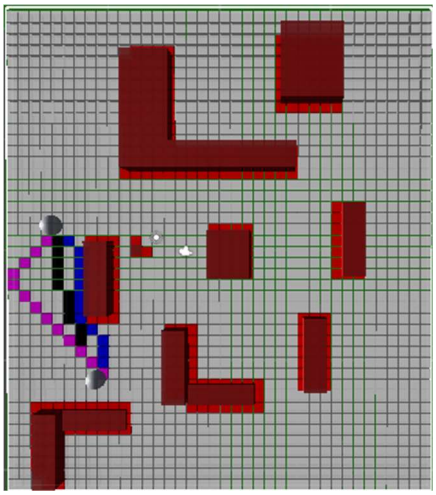






As it can be seen above, all the algorithms have different time consumption depending on the case. For the first case, BFS took longer than all other algorithms (47 ms vs 1 ms for DFS and negligible amount of time for the other ones). However, for the second case DFS took the longer time. It is noticeable also that A* is usually the fastest among all of them (in most cases). Another detail to notice is how BFS and UCS can be very close in terms of the time taken. This is the case because the cost is the same for all directions.

4. Comparison Between A*, UCS, BFS, and DFS Based on Memory



[18:35:22] Time taken by A* algorithm: 0 ms, path length: 13, Fringe Memory: 44
UnityEngine.Debug:Log (object)

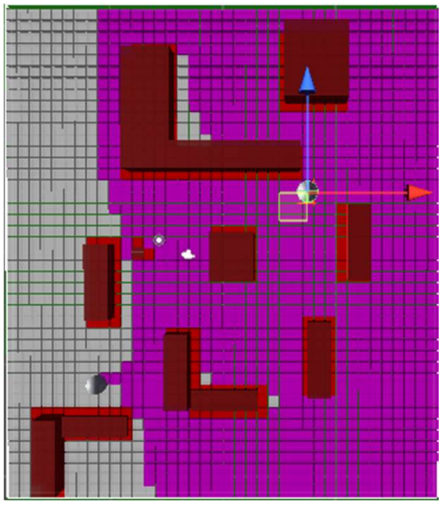
[18:35:22] Time taken by A* (With Manhattan Heuristic) algorithm: 0 ms, path length: 13, Fringe Memory: 44
UnityEngine.Debug:Log (object)

[18:35:22] Time taken by A* (With Euclidean Heuristic) algorithm: 0 ms, path length: 13, Fringe Memory: 50
UnityEngine.Debug:Log (object)

[18:35:22] Time taken by BFS algorithm: 15 ms, path length: 13, Fringe Memory: 364
UnityEngine.Debug:Log (object)

[18:35:22] Time taken by UCS algorithm: 0 ms, path length: 13, Fringe Memory: 364
UnityEngine.Debug:Log (object)

[18:35:22] Time taken by DFS algorithm: 2 ms, path length: 13, Fringe Memory: 3128
UnityEngine.Debug:Log (object)



[18:41:38] Time taken by A* algorithm: 0 ms, path length: 21, Fringe Memory: 112
UnityEngine.Debug:Log (object)

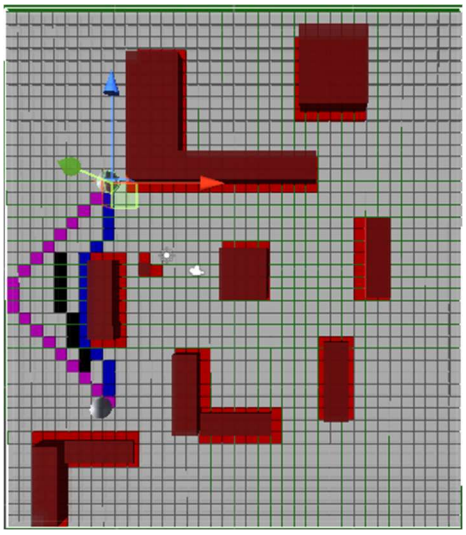
[18:41:38] Time taken by A* (With Manhattan Heuristic) algorithm: 0 ms, path length: 21, Fringe Memory: 93
UnityEngine.Debug:Log (object)

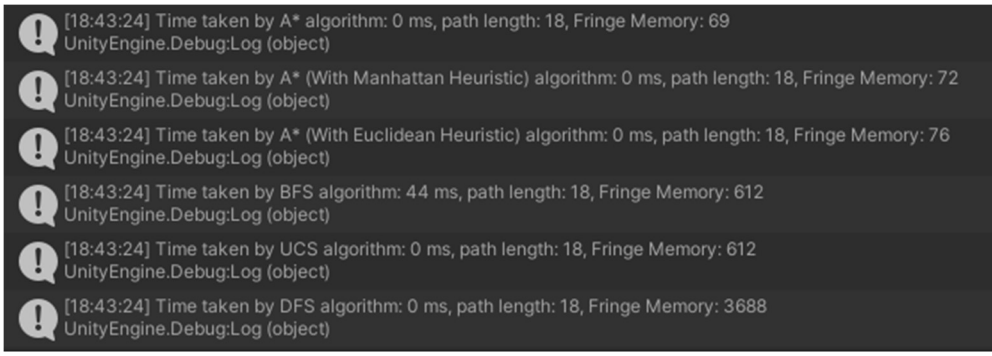
[18:41:38] Time taken by A* (With Euclidean Heuristic) algorithm: 0 ms, path length: 21, Fringe Memory: 87
UnityEngine.Debug:Log (object)

[18:41:39] Time taken by BFS algorithm: 93 ms, path length: 21, Fringe Memory: 878
UnityEngine.Debug:Log (object)

[18:41:39] Time taken by UCS algorithm: 1 ms, path length: 21, Fringe Memory: 878
UnityEngine.Debug:Log (object)

[18:41:39] Time taken by DFS algorithm: 0 ms, path length: 960, Fringe Memory: 3688
UnityEngine.Debug:Log (object)





Again, depending on each case and target's and seeker's position, and ultimately the obstacles positions, the memory (based on the fringe to have a precise idea about the memory of each algorithm separately) exploited by each algorithm differs. However, some general conclusions can be made. A* with the different heuristics always exploits a lesser amount of memory compared to the other algorithms. I think that is mainly due to the fact that A* make good use of heuristics to have a futuristic view on the optimal path and thus not a lot of memory is wasted for the nodes to be expanded yet. Another observation that can be made is that UCS and BFS often make use of the same memory size. I think this is explained by the fact that we have equal cost in all directions and thus UCS is similar to BFS in this project. Another detail to notice is the difference between Manhattan and Euclidean heuristics in the first and second cases. If there is huge "diagonal" distance between the seeker and the target, the fringe memory is bigger in Manhattan and vice versa. Finally, DFS often had the biggest fringe memory.

References

Youtube.com. 2022. A* Pathfinding. [online] Available at: <https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW> [Accessed 4 February 2022].

Theory.stanford.edu. 2022. Heuristics. [online] Available at: <http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html> [Accessed 11 February 2022].