Comparison of AI Search Algorithms
Author: VedFI

GAME

GOAL: Reach from the initial state to the target state using actions.

ACTIONS: Move Left, Move Right, Move Up, Move Down.
(The number of these actions may vary if the agent is on the frame.)

-START STATE-	**-GOAL STATE-**
[3][2][5]	[#][1][2]
[7][#][8]	[3][4][5]
[4][1][6]	[6][7][8]

USED SEARCH ALGORITHMS

- Breadth-First Search (BFS)
- Depth-First Search (DFS) (Randomised)
- Iterative Deepening Search (IDS)
- A* Search (h1 used)

HEURISTIC FUNCTIONS FOR A* SEARCH

- h1( ) : Calculates every block object's distance between current state and goal state. (w/ Manhattan's D.)
- h2( ) : Calculates the number of blocks that their position is not on goal position.
- h3( ) : Calculates number of neighbor blocks that their position is not on goal position.
- h4( ) : Calculates the distance between neighbor blocks with their goal positions. (w/ Manhattan's D.)

When heuristic functions are compared to each other, the most successful (dominating others) function is h1. Although the results are very close to each other at low difficulty levels, it is observed that as the difficulty level increases, the performance of h1 is much higher than other functions. (Second Page)

When heuristic functions are ordered according to their performance: h1 > h2 > h4 > h3

COMPARISON CRITERIAS

- Runtime (ms)
- Used Memory (mb)
- Number of expanded nodes

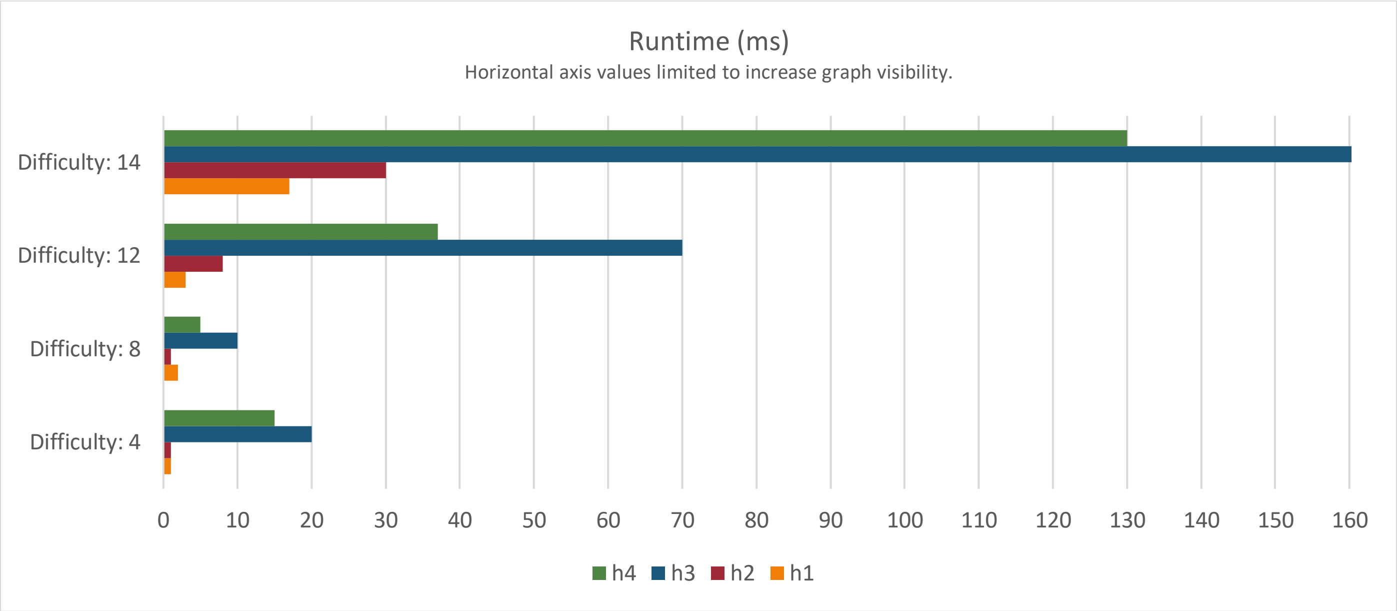
OBTAINED RESULTS

- A * is by far the most successful search algorithm. (All criteria are considered.)
- IDS is the most successful algorithm among uninformed search algorithms.
- DFS is not complete because there are an infinite number of actions in the game. It needs to be randomized. (The way in which the nodes open is determined at random)
- DFS is not optimal and consumes much memory.
- BFS and IDS algorithms are optimal because all moves in the game have equal costs.
- Among the search algorithms examined, BFS is the most memory consuming algorithm.
- BFS and DFS are the slowest algorithms when run times are analyzed. (Depending on the chance factor, DFS may be faster or slower.)
- IDS can be preferred if A * is not available.

NOTES

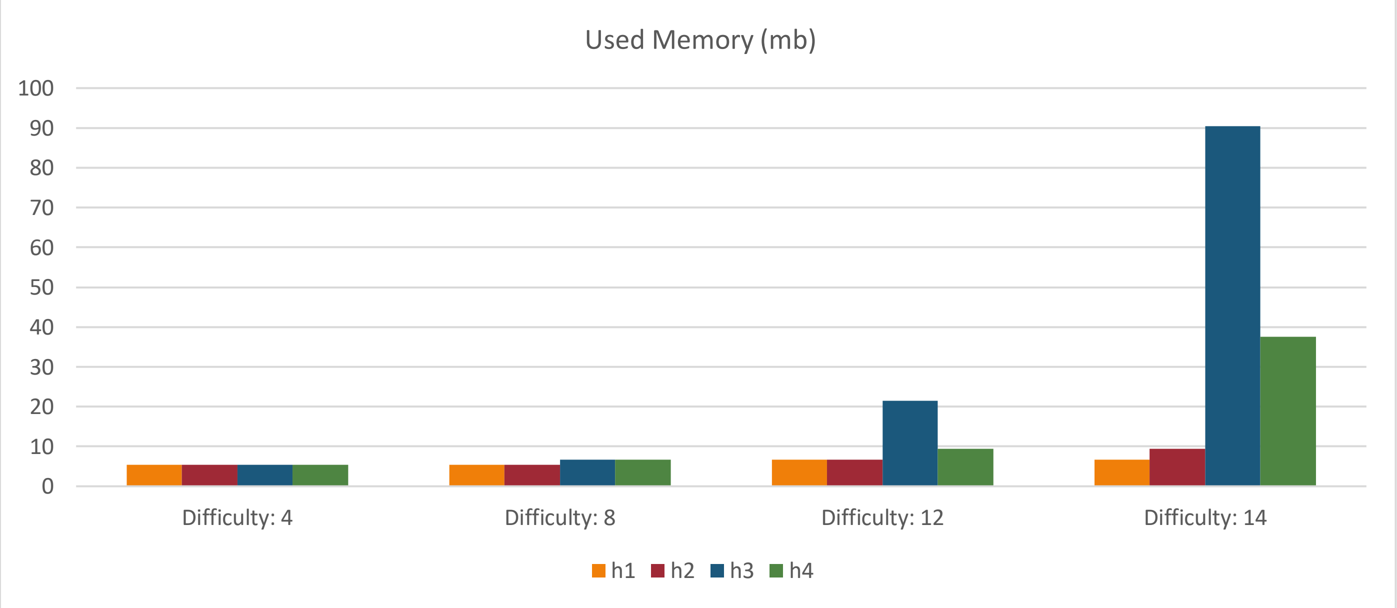
1. Source files are written by me.
2. Runtime was measured using the System.currentTimeMillis() method.
3. The amount of memory used was determined by the ManagementFactory .getMemoryMXBean().getHeapMemoryUsage().getUsed() method.
4. The amount of expanded node is calculated using a counter in the search algorithms.
5. The technical report contains the initial conditions and measurements in which comparisons are made.

COMPARISON OF HEURISTIC FUNCTIONS

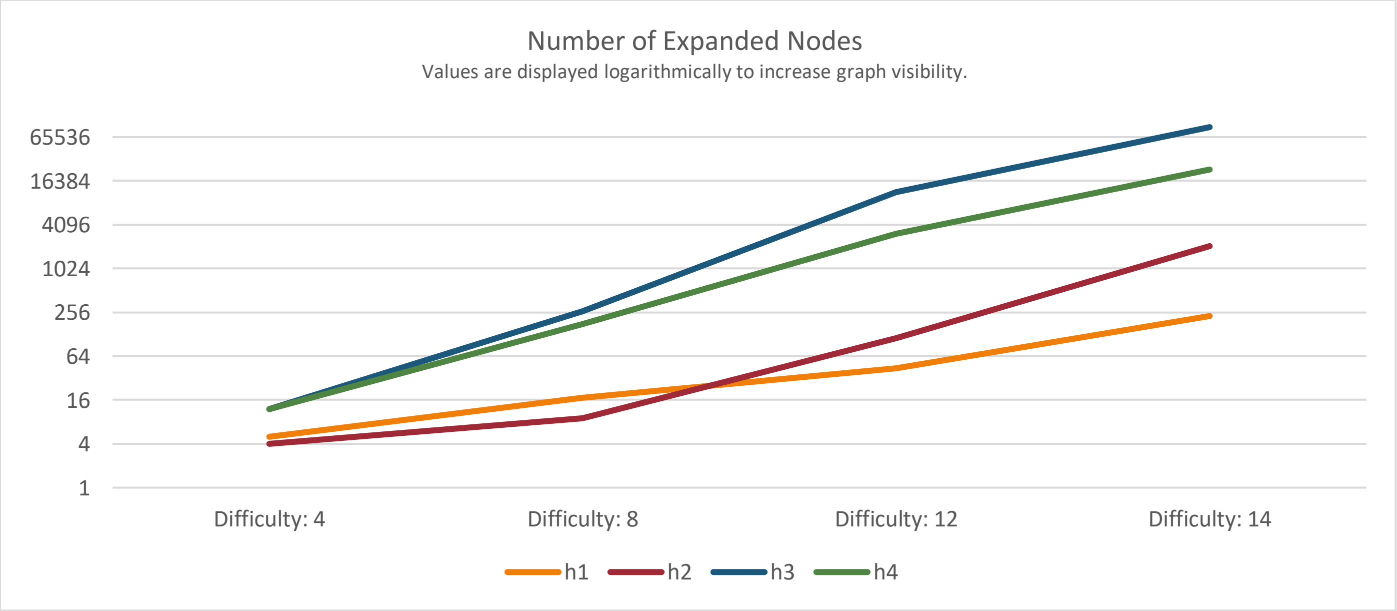


Runtime (ms)

Horizontal axis values limited to increase graph visibility.

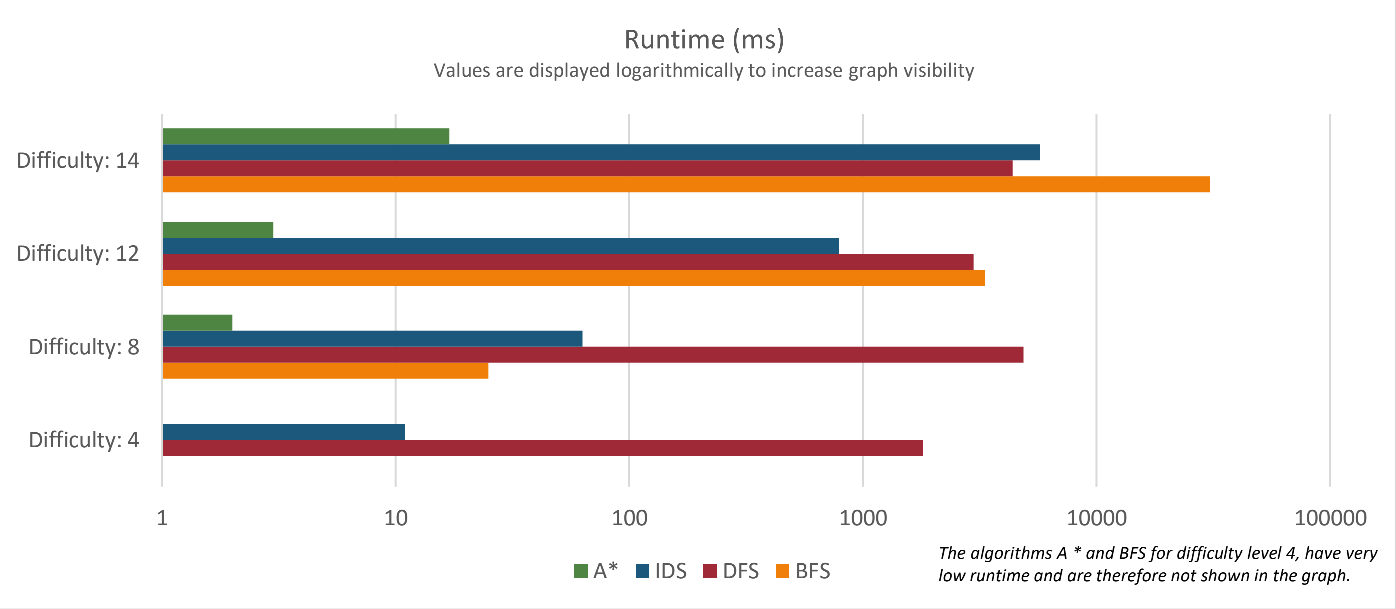


Used Memory (mb)

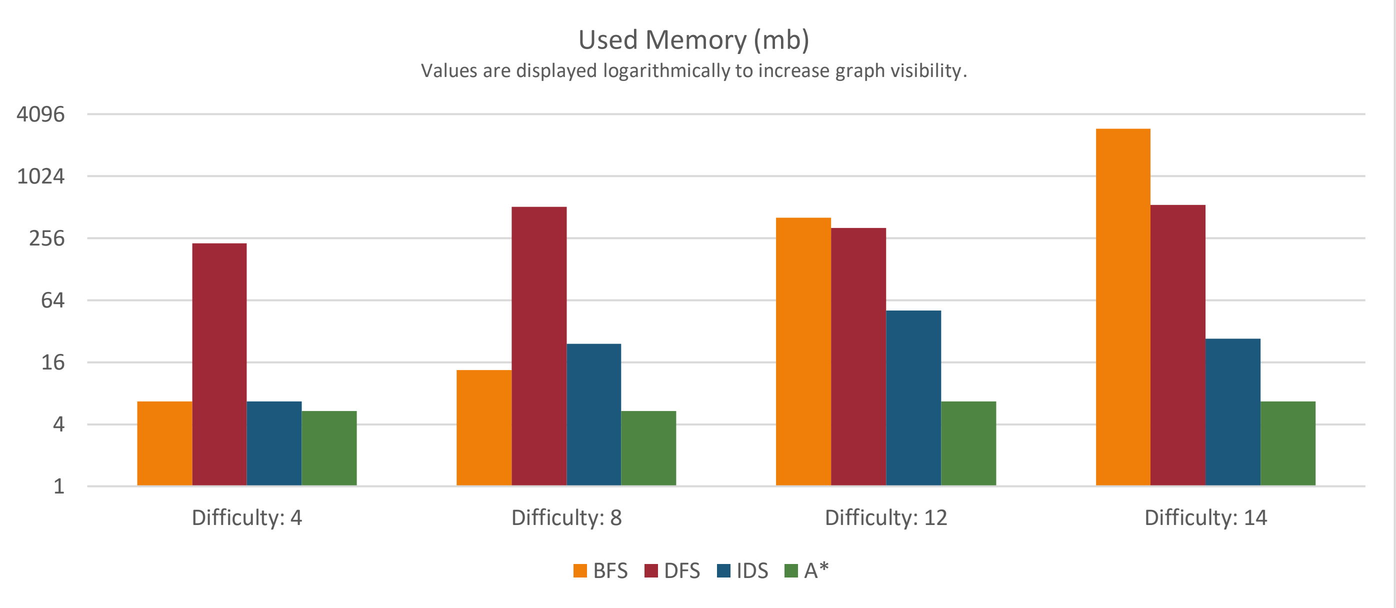


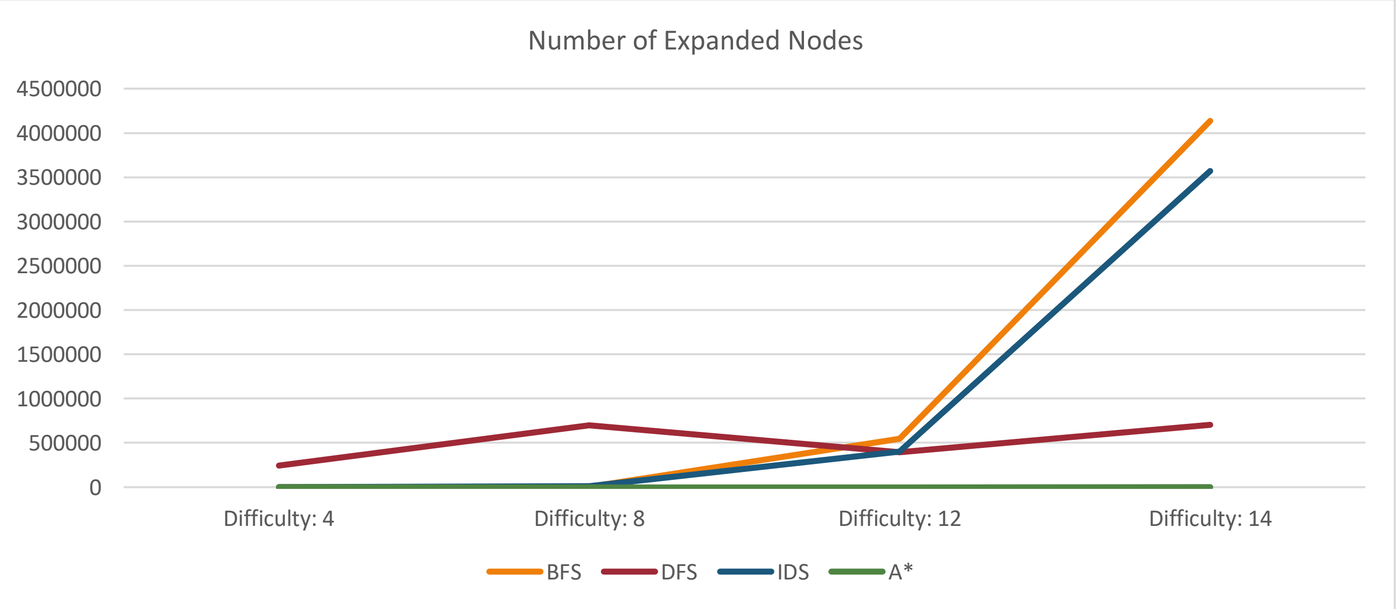
Number of Expanded Nodes

Values are displayed logarithmically to increase graph visibility.



*The algorithms A * and BFS for difficulty level 4, have very low runtime and are therefore not shown in the graph.*


Used Memory (mb)
Values are displayed logarithmically to increase graph visibility.
Difficulty: 4
Difficulty: 8
Difficulty: 12
Difficulty: 14
BFS
DFS
IDS
A*


Number of Expanded Nodes
Difficulty: 4
Difficulty: 8
Difficulty: 12
Difficulty: 14
BFS
DFS
IDS
A*