|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Average Depth** | **Average Nodes** | **Average Time** |
| **Average BFS** | |  | **27** | **155199.5** | **2.529177656** |
| **Average DFS** | |  | **66968.2** | **156101** | **2.079816747** |
| **Average A\* Misplaced Tiles** | |  | **26.9** | **104704.7** | **2.171307519** |
| **Average A\* Manhattan Distance** | |  | **26.4** | **92367.6** | **2.466042536** |
| **The Averages of these searches include valid puzzles as well as invalid puzzles.** \* The breadth first search (BFS) has a similar depth to the A\* searches, but the node count is considerably higher which shows that the breadth first search is defiantly not memory friendly.  \* The DFS has an extremely high depth due to the search expanding as far as possible before traversing down the next possible path. This version shows the maximum depth reached when attempting to find the solution. The node count is also high. This result could be different with other random puzzles. The average search time for depth first was slightly better than the average time for the breadth first search. \* The A\* misplaced tiles had a very similar depth to the BFS, which is expected. The average nodes were a good bit lower. This average would have been even lower if the invalid puzzles would not have been a factor. The average time was slightly faster, and again this is due to the invalid solutions being calculated in the average. \* The A\* Manhattan distance had a slightly better average depth than the breadth first search and A\* misplaced tiles search. The average nodes expanded was the least of all the search types, which is definitely expected. The average search time for the A\* Manhattan was slightly higher than the misplaced tiles search which came as a surprise, but then I considered that this was again due to the invalid solutions being a factor in the averages.  I actually tested the paths of valid solutions with my phone. I downloaded a slider 8 puzzle app, and I would start a random puzzle on my phone. I would then input the location of each tile into my program to simulate the same puzzle that was displaying on the app. When the solution was given by my program, I would follow the path that was given and find the solution to the puzzle on my phone. It was a fun little way to test my programs solutions.  The A\* searches were incredibly fast for valid puzzles. Given the heuristics, the agent was able to find the optimal solution. The misplaced tiles algorithm didn’t have as wide of a heuristic range for placement in the priority queue as the Manhattan distance did. The Manhattan distance allowed for greater accuracy without over estimating which in turn provided the optimal solution in the fastest search time when the puzzle was valid. | | | | | |
|
|
|
|
|
|
|
|  |  |  | **Average Depth for Valid Puzzles** | **Average Node Count for Valid Puzzles** | **Average Time for Valid Puzzles** |
| **Average BFS** | |  | **23.2** | **128959** | **1.898357255** |
| **Average DFS** | |  | **51035.6** | **130762** | **1.556707195** |
| **Average A\* Misplaced Tiles** | |  | **23.2** | **27969.4** | **0.437628652** |
| **Average A\* Manhattan Distance** | |  | **23.2** | **3295.2** | **0.06207284** |
| **The Averages of these searches includes only valid puzzle solutions. The invalid solutions were not** **added into the calculations.**  \*The breadth first search A\* misplaced and A\* Manhattan searches all have the same average depth which is expected. The node count for the depth first search is considerably higher than all the other searches, which again is due to the stack data structure causing the search to traverse all the way down one branch before moving to the next branch.  \*The average node count for the breadth first search shows that this type search is definitely not memory friendly. Even though only valid solutions were factors in these averages, breadth first search still expanded A LOT of nodes to find the solution. This is due to the queue data structure.  \*The average node count for the A\* misplaced tiles is quite a bit lower than the node count for BFS or DFS. This is due to the heuristic being used in correlation with the depth of each node. This value is used to add nodes to the priority queue. The lower the heuristic value, the better chance at finding a solution. The search time is considerably less and again this is due to the heuristic being used.  \* The average node count for the A\* Manhattan distance is much, much lower than all the other searches. This is because the A\* Manhattan search finds the optimal solution with the depth of the node added to the heuristic value. The heuristic calculates how many positions the current tile is out of place and then adds that number to the total heuristic for that state. This total heuristic value for the state is then added with the depth at that state and this value is used to determine where the node should be placed in the priority queue. This ensures that the node with the lowest heuristic will be chosen every time. The search time for the A\* Manhattan distance is incredibly fast. The optimal solution is found faster, due to the heuristic being more precise than it was for the A\* misplaced tiles. | | | | | |
|
|
|
|
|
|
|
|