|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Average number of nodes visited** (you need repeat each algorithm several times with different initial configuration) | **Average run time in your experiments** | **Your comment on these algorithms** |
| **DFS** | 24773.0 | 32.653711 | DFS explores the deepest branch of the search tree first. In the context of the 8-puzzle problem, this means it may go down a long path of moves before backtracking. As a result, it often explores a large number of nodes, which can lead to inefficient search times. |
| **UCS** | 19.0 | 0.000494 | UCS aims to find the solution with the lowest cost. In the case of the 8-puzzle, this means it explores nodes in increasing order of the total cost incurred to reach them. UCS performs well when the solution is close to the initial state and doesn't involve a high branching factor. |
| **BFS** | 29.0 | 0.000500 | BFS explores all nodes at a given depth level before moving to the next level. It is guaranteed to find the optimal solution with the fewest moves in the 8-puzzle problem. However, it may require more memory and time compared to UCS, especially for puzzles with a high branching factor. |
| **A\*** | 181214.9 | 3.7253645181655886 | A\* combines the advantages of both UCS and heuristic information. It uses a heuristic to estimate the cost from the current state to the goal, guiding the search towards more promising paths. However, the quality of the heuristic can significantly impact its performance. In your experiments, it visited a large number of nodes, which may suggest that the heuristic used was not very informative or that the initial configurations were particularly challengingW |