**Comparison of Search Algorithms for the 8-Puzzle Problem**

**22l-7497**

**Introduction**

This document presents a comparative analysis of four search algorithms used to solve the 8-puzzle problem: **Breadth-First Search (BFS), Depth-First Search (DFS), Iterative Deepening Search (IDS), and Uniform Cost Search (UCS).** The comparison is based on four key performance metrics:

1. **Number of Nodes Visited** - The total number of nodes explored before reaching the goal state.
2. **Path Cost** - The cost associated with the solution path.
3. **Memory Consumed** - The maximum number of nodes stored in memory at any time.
4. **Time Taken** - The total time required to find the solution.

**Comparison Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Nodes Visited** | **Path Cost** | **Memory Consumed** | **Time Taken (seconds)** |
| BFS | 7 | 2 | 8 | 0.000222 |
| DFS | 3 | 2 | 3 | 0.000070 |
| IDS | 8 | 2 | 2 | 0.000060 |
| UCS | 4 | 2 | (4 | 0.000074 |

**Observations & Analysis**

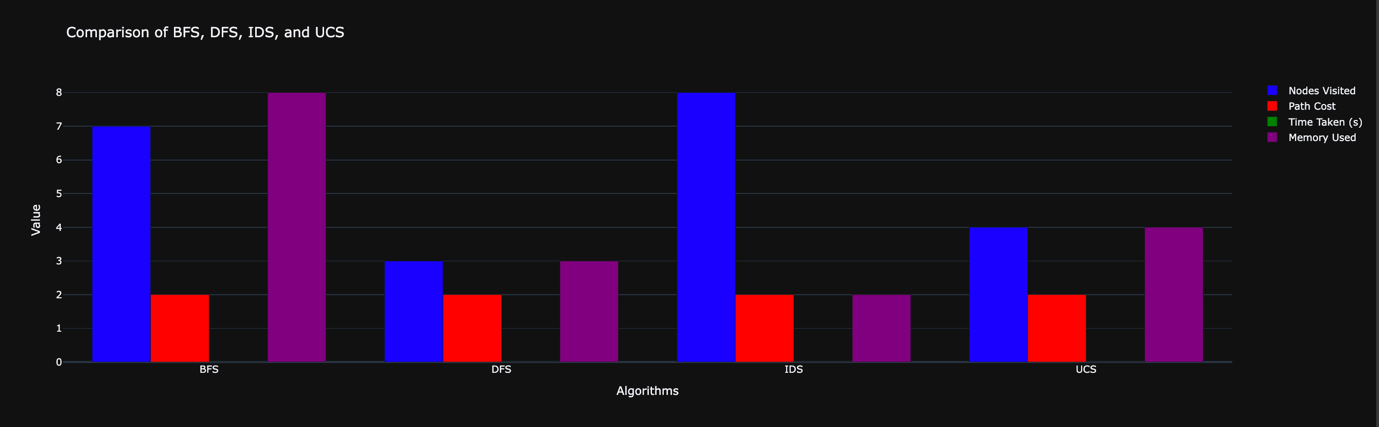
1. **BFS** ensures the shortest path but may consume more memory due to storing all nodes at the same depth.
2. **DFS** explores deep paths first but may get stuck in deep or infinite paths, requiring backtracking.
3. **IDS** combines the benefits of BFS and DFS, though it repeats searches at different depths, increasing computation time.
4. **UCS** prioritizes the lowest-cost path, making it efficient for weighted problems but potentially slower if all moves have equal cost.

**Conclusion**

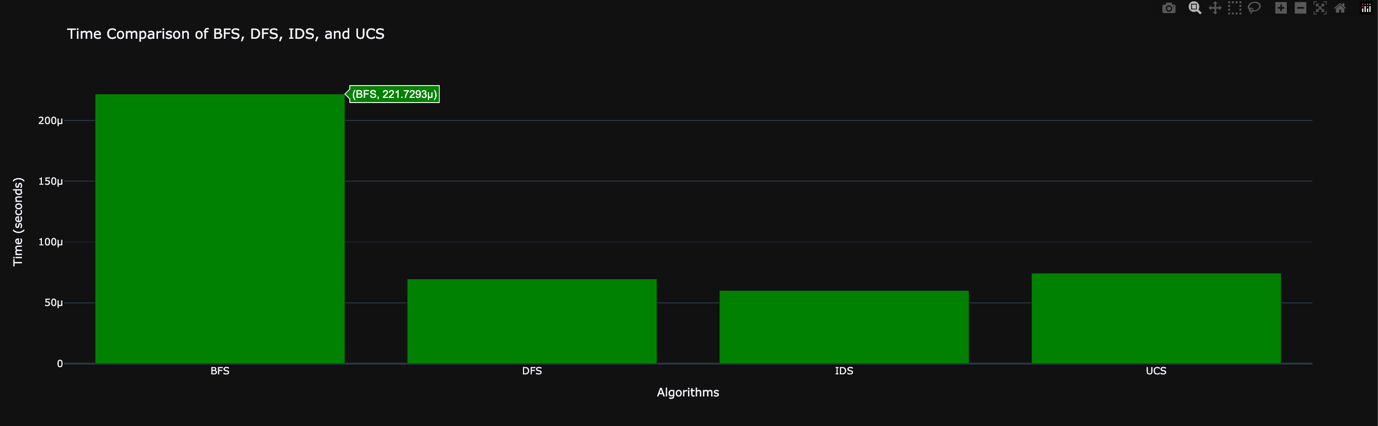
Each algorithm has its strengths and weaknesses depending on the problem constraints. For optimality, BFS and UCS perform well, whereas DFS and IDS are better suited for problems requiring memory efficiency. The choice of algorithm depends on the trade-off between speed, memory, and solution accuracy.

**Graphs for Algorithm Comparison** The performance of all four algorithms is visualized through bar graphs generated using Plotly. These graphs highlight differences in execution time, node visits, path cost, and memory consumption.

1. **Graph 1**: Nodes Visited, Path Cost, and Memory Used Comparison



1. **Graph 2**: Time Taken by Each Algorithm



**Reference and GitHub Link:**