

CS 441: Program 1

Josue Lopez

1 Introduction

Results for solving the 8-puzzle using Best-First Search and A* Search algorithms with three different heuristics on 5 initial states:

- **Heuristic 1 (h1):** Misplaced tiles
- **Heuristic 2 (h2):** Manhattan distance (L1 norm)
- **Heuristic 3 (h3):** $\max(h1, h2)$

Initial States Used

1.
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ b & 7 & 8 \end{bmatrix}$$
2.
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & b & 5 \\ 6 & 7 & 8 \end{bmatrix}$$
3.
$$\begin{bmatrix} 1 & 2 & 3 \\ b & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$
4.
$$\begin{bmatrix} 3 & 1 & 2 \\ 6 & 4 & 5 \\ b & 7 & 8 \end{bmatrix}$$
5.
$$\begin{bmatrix} 8 & 6 & 7 \\ 2 & 5 & 4 \\ 3 & b & 1 \end{bmatrix}$$

4 Conclusion

The results clearly demonstrate that A* Search significantly outperforms Best-First Search in terms of efficiency, consistently requiring fewer steps to reach the goal state across all heuristics and initial states. While Best-First Search with heuristics h2 and h3 improved performance compared to h1, A* won by incorporating the cost to reach a node ($g(n)$) in addition to the heuristic estimate ($h(n)$). This allowed A* to explore more promising paths, being faster and leading to more reliable solutions. Interestingly, heuristics h2 and h3 produced nearly identical performance under both search strategies, suggesting that combining heuristics did not degrade A*'s efficiency. Overall, A* not only yielded lower step counts but also demonstrated greater consistency and scalability across the different initial states. While the textbook, and class lectures, said this was the case, it's interesting to see it happen with a real example. This highlights how significantly an algorithm can impact complexity and why it's important to choose the right one, especially in AI applications.

