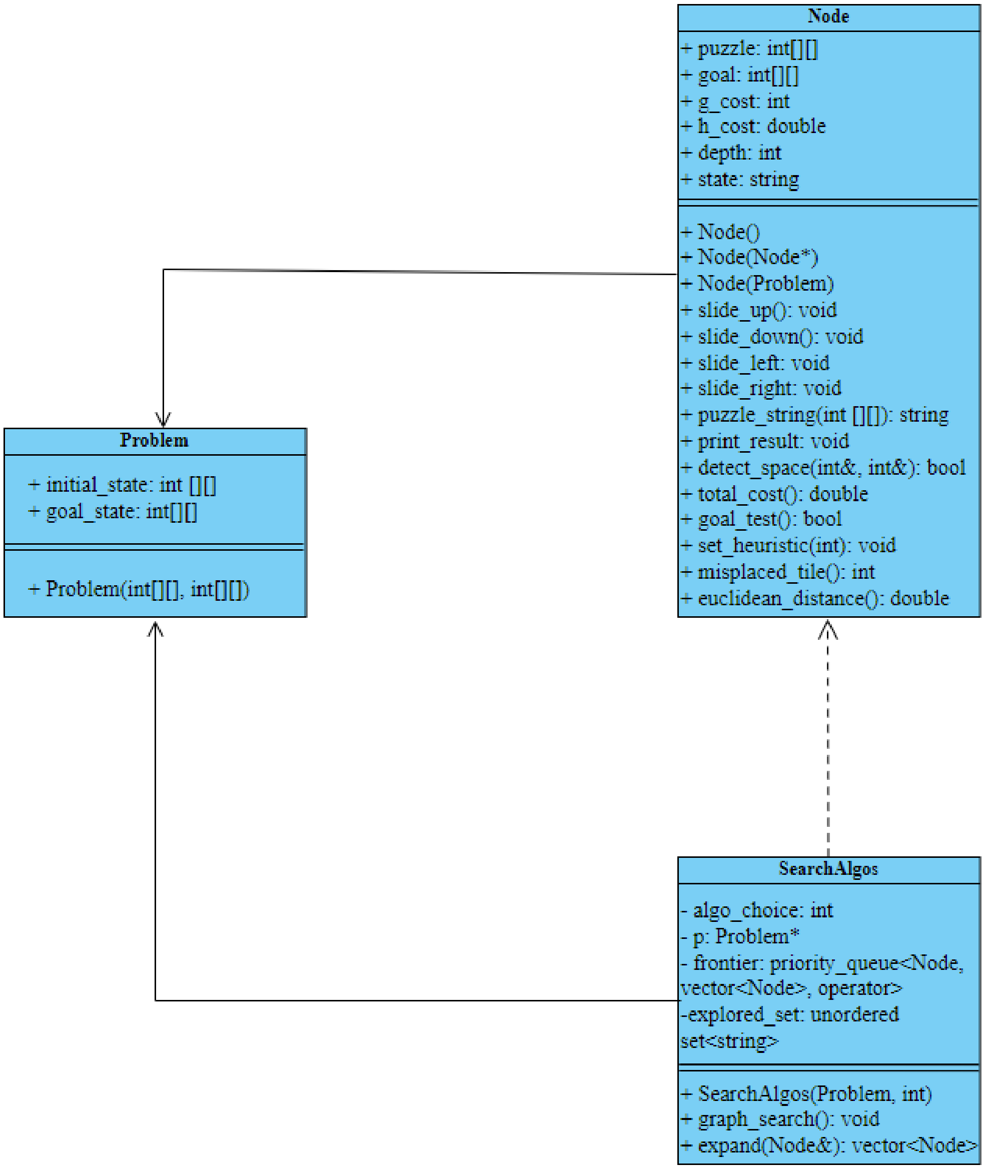
Project#1 8 Puzzle

CS-170 Section 001

Sidharth Ramkumar
862129657

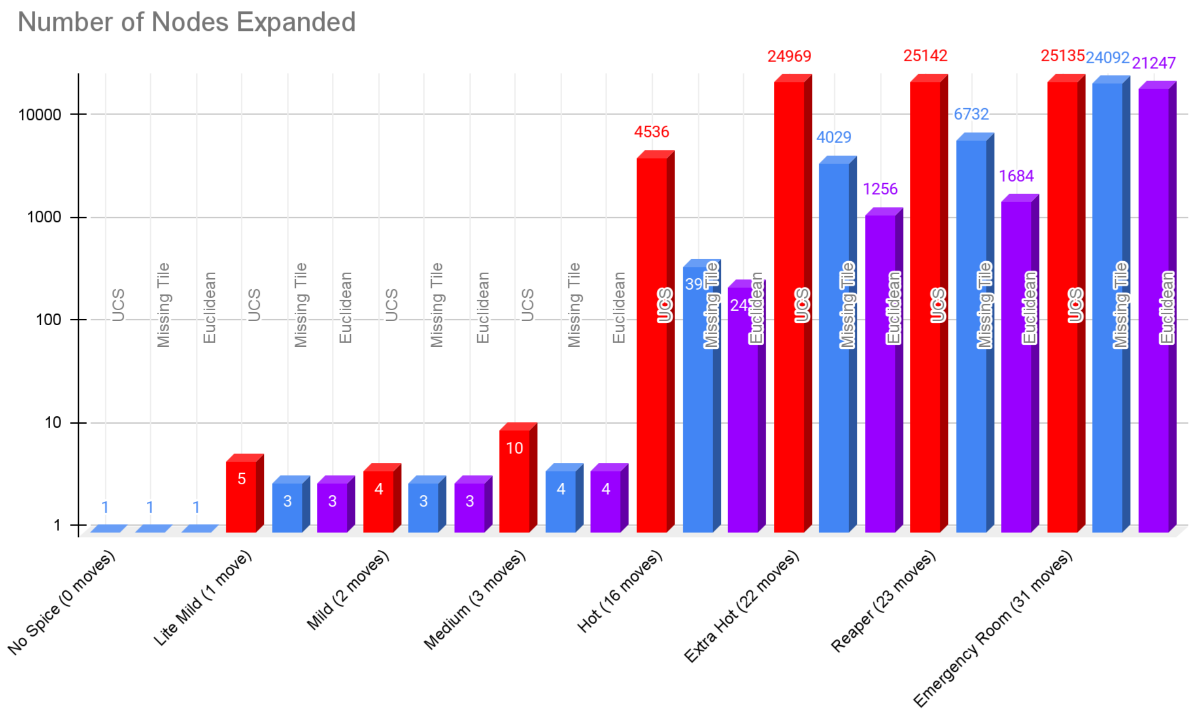
Project Design

I had designed this project to fit the graph-search function and perform uniform-cost search and A* search with very minimal changes. I had used graph search to reduce the amount of time it takes to solve larger puzzles while keeping track of the relevant information for each step.

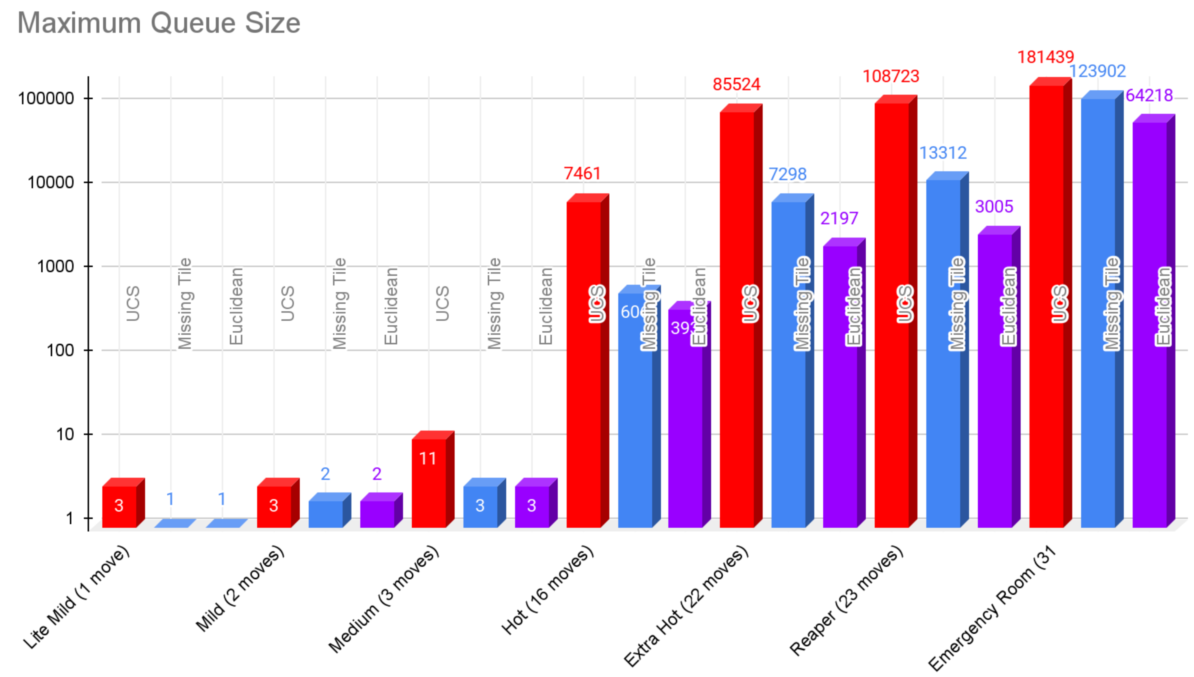


Some of the more significant data structures include a priority queue and an unordered set. The priority queue was the frontier from my graph search, allowing for very quick ordering of Nodes pushed to the queue and existing Nodes in the frontier. The unordered set was used to keep track of explored Nodes and includes hash values to retrieve elements in O(1) runtime.

Comparing Heuristics



Number of Nodes Expanded



Maximum Queue Size

Findings

To measure the time and space for heuristic search functions I utilized seven different 8-puzzles. The number of nodes expanded grew exponentially given the steady increase in the number of moves to solve the puzzle or node depth for the solution. Comparing each function, it is apparent that uniform-cost search has a steep growth curve in terms of time complexity w.r.t difficulty. Whereas the misplaced tile has a slightly slower growth with euclidean distance being almost linear. The same can be said for the space complexity, with the queue size for each node rapidly growing as more optimal moves are required to get the solution. As the problems increase in difficulty, the euclidean distance is the most helpful, with the misplaced tile heuristic being a bit behind. Therefore it is clear that a good heuristic function is very relevant to get optimal results with minimal space and time usage.

Challenges

One major challenge was keeping the code very general across different search algorithms. Although it was more efficient to keep track of code and switch between different search algorithms, creating one large search function would have been a lot easier. Also, I had an issue with the priority queue (frontier), where the Nodes were not sorted in the correct order. This would produce very weird errors that took me a long time to debug. I was able to reach the goal of the 8-puzzle, but it took well over 1000+ Nodes to reach it for simple puzzles while the program was bugged.

Project Link

https://github.com/sidrk01/CS170-The_Eight_Puzzle

Resources

https://cplusplus.com/forum/general/263317/
https://stackoverflow.com/questions/19535644/how-to-use-the-priority-queue-stl-for-objects
https://stackoverflow.com/questions/5374311/convert-arrayliststring-to-string-array