Carlos Vargas
June 21st, 2025

CS 4200 - Artificial Intelligence
Professor Daisy Tang
California State Polytechnic University - Pomona

Project 1 Report

Approach

The approach that I took for this assignment was to break it down by tasks. I decided to start with the task of converting the A* algorithm from pseudocode into python because I figured this would be the most difficult for myself to understand — I was sorely correct. I ended up finding a good pseudocode version of the program here: https://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf.

The biggest thing that i struggled with on the A* algorithm was understanding the different data structures. Everything changed when I thought about making a class for the nodes and sorting the frontier by priority queue and then exploring it as a hash — easily searchable.

In terms of comparing my approaches to the two heuristics, I don't have much to compare in terms of the implementation. Given that it was straightforward to find the Manhattan distance (and interesting to make it more efficient) and the misplaced tiles were obviously straight forward as well.

I encourage you to play around with inputs and edge cases because I spent a lot of time ensuring that the user can't screw it up — only get booted out.

I thought it was interesting that the algorithm was *insanely* faster when using queues and hash tables versus using plain sets.

⭐ it should be noted that I included outputs in the zip for depths of 8, 12, 20. This goes beyond the 6-12 option. Given that we don't have to generate the specific depths of the puzzles I used the puzzles provided to us.

TABLE OF RESULTS

trials	depth	h1 ave cost	h2 ave cost	h1 ave time	h2 ave time
10	4	12.56	0.88	0.21 ms	0.01 ms
9	8	89.57	1.72	1.14 ms	0.02 ms
10	12	460.73	3.42	4.70 ms	0.03 ms
10	16	2185.07	6.33	24.54 ms	0.05 ms
10	20	7077.51	23.68	87.12 ms	0.22 ms
100	random	57012.31	562.08	931.00 ms	5.72 ms

My results from testing the files that we were given, as well as 100 random generations.