Marlon Aquino

Professor Dominick Atanasio

CS 4200.01

23 February 2020

Project 1: 8-Puzzle Problem

My Approach:

At a high level, the 8-puzzle problem implementation covered during class was fairly simple. However, it was difficult to find out how to implement it in Java. I also was not familiar with the Manhattan distance and Hamming distance heuristics that need to be used with the A* algorithm. In order to implement these, I had to look them up, one of which would be using a HashMap. Using a HashMap, I was able to find the distance and apply it for the A* algorithm. With the Tree-Search and Graph-Search algorithms, it was pretty straightforward implementing the two.

In addition to that, I also had to consider if the puzzle was even solvable. This was done by checking to see if the number of inversions were even. I created a method called isSolvable() that counts the number of inversions on the board and returns true if the number of inversions % 2 would equal 0. Otherwise, the puzzle isn't solvable.

I also had to take into account a few edge cases. I had to consider if the user entered more than nine numbers, any numbers other than 0 - 8, repeated numbers, and making sure that the user must input a 0 for an empty space. If the entered puzzle didn't pass any of these tests, then I made sure to return an exception that lets the user know why the puzzle is not valid / is not able to be solved.

Comparison of the Two Algorithms

I ran a hundred cases from a variety of solution depths (two to twenty) and I found that the Manhattan Distance along with Graph Search is more efficient since it finishes the puzzle in a shorter amount of time. I think the reason why the Manhattan

Distance was more efficient was due to how it takes into account how far the tile is from the goal position. In comparison to the Hamming Distance, it only cares about whether or not the tile position is in the correct place or not. The Manhattan Distance takes more into consideration such as how much it would cost to go through a certain node and if it takes it, how much closer it is to the goal. Additionally, the advantage of graph search is that it has the explored set so when a node has been seen already, it won't search it again. However, Graph Search also uses more memory so there is a trade-off between space and time. The Hamming Distance (aka the number of misplaced tiles) had better numbers at lower depths, but at higher depths it became worse exponentially.

Table on A* with Graph Search and A* with Tree Search

	Search Cost (Graph Search)					**Search Cost (Tree Search)**				
d	**A* using Hamming Distance**	**A* using Manhattan Distance**	**Average running time (ns)** **H1**	**H2**	**Number of Test Cases**	**A* using Hamming Distance**	**A* using Manhattan Distance**	**Average running time (ns)** **H1**	**H2**	**Number of Test Cases**
2	9	4	282500 ns	116400 ns	100	5	10	193900 ns	232000 ns	100
4	17	13	443200 ns	346200 ns	100	24	28	249100 ns	375700 ns	100
6	37	33	11381100 ns	621300 ns	100	84	72	695300 ns	351700 ns	100
8	80	48	10241100	772700 ns	100	91	505	668700 ns	1790500	100

			ns						ns	
10	90	50	759100 ns	778000 ns	100	105	228	733800 ns	926000 ns	100
12	231	133	1585300 ns	965300 ns	100	1386	7658	3336600 ns	6524400 ns	100
14	668	284	3087900 ns	2687000 ns	100	16958	103253	15789300 ns	45920700 ns	100
16	1361	389	4329700 ns	4181500 ns	100	146079	332407	69440800 ns	110351800 ns	100
18	3487	446	4506900 ns	6901600 ns	100	25135	748770	25289400 ns	250651300 ns	100
20	7524	534	3169600 ns	70351300 ns	100	7492	3602215	8383300 ns	359642510 0 ns	100

What I Learned

Through this project I was able to learn how the A* algorithm, Tree Search algorithm, as well as the Graph Search algorithm works with different heuristics. It was interesting to see how each test case differed depending on the solution depth. This project was difficult to implement, but after tons of research and watching YouTube tutorial videos, I was able to get it to work. Overall, I really enjoyed this project and it showed me that you can implement different algorithms to achieve the same result. There is just the trade-off in regards to space and time efficiency that you need to take into account.