Alex Pham

CS420

**CS420 Project : 8-Puzzle**

I began the project by trying to use an array list to run the board, and use Manhattan Distance or the misplaced tile heuristics to sum the paths. These are used with the A\* algorithm and I was not too familiar with either of them. I had to specifically search for the ways to implement these algorithms in Java. Using a hash map, I find the distance and apply it to the function. The A\* algorithm is the total path costs with the node and total sum (f(n) = g(n) + h(n)). In theory, it was fairly easy, but finding out which method to use was difficult. A key point that I had to take into account was to check if the initial state was even solvable. This was done by a method I have and finds the inversions. The puzzle must have an even number of inversions, otherwise it will not be solvable. This is done by a modulus of 2 and returns true. Also, the user must input an empty tile, the “0,” and we make sure that there are no repeats between 0 and 8.

After running a hundred cases of depths from two to twenty, I found that the Manhattan Distance is more efficient in the long run because it finishes the puzzle in a shorter time and uses less nodes. The Misplaced tiles function has some better numbers at lower depths, but with a higher depth it is exponentially worse.

On the assignment, I found it rather hard to figure out how to use the heuristics to influence the empty tiles. It was a struggle to find out how to print the puzzle board each time while being efficient. After finishing what I had, I realized I had to test 100 cases and so I had to go back and find a way to run multiple instances in the format that was given to us. Also, I was having an oddly hard time trying to get the time to show a number other than zero. Then, I realized I was doing the time in milliseconds and the puzzle was being solved too fast to time it in milliseconds. Thus, I had to change it to nanoseconds and I got some results. That is why the numbers on my tables are so large. Actually, on just the regular sample output, the time is in milliseconds still because the 100 test cases were tested on my PC which did have a problem with milliseconds. However, the laptop could do it in milliseconds. Therefore, the times are in different units. Ultimately, I learned a lot from this assignment and it has given me a great lesson on different ways to implement something along the lines of a list.

H1 = misplaced tiles h2 = Manhattan Distance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Search Cost | | Average Run Time (ns) | | Number of Cases |
| d | A\*(h1) | A\*(h2) | A\*(h1) | A\*(h2) |  |
| 2 | 6 | 6 | 287716 | 199577 | 100 |
| 4 | 9 | 11 | 340235 | 383504 | 100 |
| 6 | 16 | 17 | 342975 | 391669 | 100 |
| 8 | 31 | 25 | 363836 | 395767 | 100 |
| 10 | 83 | 42 | 366454 | 395428 | 100 |
| 12 | 168 | 75 | 573824 | 451502 | 100 |
| 14 | 439 | 162 | 755191 | 591980 | 100 |
| 16 | 1185 | 363 | 1313011 | 824449 | 100 |
| 18 | 3188 | 708 | 3738513 | 987855 | 100 |
| 20 | 8244 | 1265 | 25317669 | 1494790 | 100 |