Artificial Intelligence Project 2

8-Puzzle

Code and Report by Caleb Pudvar

SUMMARY:

The goal of this project was to create a program which was capable of solving a 3-by-3 sliding puzzle. This program is written to compare two Best-First search heuristics: misplaced tiles and Manhattan distance. The following data proceeds to examine the results in more detail. Sample output is also provided.

METHODS:

This program is built upon a Best-First search algorithm. This is known as a "greedy" algorithm, and is designed to choose the next best path without any ability of looking ahead. The way the algorithm is able to designate the strength of a board is by its heuristic. I have implemented two in an effort to compare them. They are:

1. Sum of Misplaced Tiles: This simply compares a board to the goal state, and tallies up each tile that is in the correct location.
2. Manhattan Distance: For each tile, the distance away from its goal position is recorded. The sum of each tile's distance results in this score.

Using a sample size of 100 executions of each heuristic, I will be able to better discuss the relationship between them.

SAMPLE OUTPUT: The following table shows the sample output of the "medium" board provided.

Best-First Search **HEURISTIC: Misplaced Tiles**	**Best-First Search** **HEURISTIC: Manhattan Distance**
FINAL PATH Starting board: --- | 3 | 2 | 5 | --- | 4 | 1 | 8 | --- | 6 | 0 | 7 | --- Move # 1 --- | 3 | 2 | 5 | --- | 4 | 1 | 8 | --- | 6 | 7 | 0 | --- Move # 2 --- | 3 | 2 | 5 | --- | 4 | 1 | 0 | --- | 6 | 7 | 8 | --- Move # 3 --- | 3 | 2 | 0 | --- | 4 | 1 | 5 | --- | 6 | 7 | 8 | ---	FINAL PATH Starting board: --- | 3 | 2 | 5 | --- | 4 | 1 | 8 | --- | 6 | 0 | 7 | --- Move # 1 --- | 3 | 2 | 5 | --- | 4 | 1 | 8 | --- | 6 | 7 | 0 | --- Move # 2 --- | 3 | 2 | 5 | --- | 4 | 1 | 0 | --- | 6 | 7 | 8 | --- Move # 3 --- | 3 | 2 | 0 | --- | 4 | 1 | 5 | --- | 6 | 7 | 8 | ---

Move # 4 --- | 3 | 0 | 2 | --- | 4 | 1 | 5 | --- | 6 | 7 | 8 | --- Move # 5 --- | 3 | 1 | 2 | --- | 4 | 0 | 5 | --- | 6 | 7 | 8 | --- Move # 6 --- | 3 | 1 | 2 | --- | 0 | 4 | 5 | --- | 6 | 7 | 8 | --- Move # 7 --- | 0 | 1 | 2 | --- | 3 | 4 | 5 | --- | 6 | 7 | 8 | --- # of moves to goal: 7 Max size of queue: 8 Total runtime: 0.00500011444092 seconds.	Move # 4 --- | 3 | 0 | 2 | --- | 4 | 1 | 5 | --- | 6 | 7 | 8 | --- Move # 5 --- | 3 | 1 | 2 | --- | 4 | 0 | 5 | --- | 6 | 7 | 8 | --- Move # 6 --- | 3 | 1 | 2 | --- | 0 | 4 | 5 | --- | 6 | 7 | 8 | --- Move # 7 --- | 0 | 1 | 2 | --- | 3 | 4 | 5 | --- | 6 | 7 | 8 | --- # of moves to goal: 7 Max size of queue: 21 Total runtime: 0.00999999046326 seconds.

Table 1: Comparison of both heuristics solving the same trivial board. Note that the max size of the queue in Manhattan Distance is much larger than that of the misplaced tiles. In this instance, Manhattan Distance took twice as long to execute as well.

RESULTS:

After tirelessly testing my heuristics, a few noteworthy things leap out at me. As shown in Table 2, the Manhattan heuristic has the better average at ~62 moves per solution. Upon looking up some traits about the 8-puzzle, I have found that all puzzles can be solved in 31 or fewer moves. This tells me that while Best-First Search is quite efficient, it does not find the optimal path to the goal state. One would need to instead implement an A* approach which tracks the depth through the tree in addition to the heuristic in order to find such an optimal path. After viewing the results in Table 2 below, I am astonished that these two heuristics perform *so* differently in terms of memory and speed. The misplaced tiles heuristic uses ~2.5 times more memory than Manhattan distance. I was unaware that a slight tweak of a heuristic would have such a beneficial result. It turns out, that although Manhattan distance has more computational overhead for determining each board's heuristic value, it still speeds up the solution rate by over 6 times on average as opposed to the misplaced tiles heuristic.

Algorithm	Average Number of Moves	Average Max Queue Size	Average Runtime (sec)	Total runtime (sec)
Best-First Search h(n) = Sum of Misplaced Tiles	84.49	466.46	1.44821	144.821
Best-First Search h(n) = Manhattan Distance	62.62	186.02	.22167	22.167

Table 2: Comparison of various factors based on heuristic used. Sample size for averages: 100. Manhattan distance completes in fewer moves than misplaced tiles on average, and completes *much* faster than misplaced tiles while maintaining a much shorter queue.