Stephen Wong

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Puzzles | H1 ( Misplaced Tiles) | | H2 (Manhattan Distance) | | Both | |
|  | Average Steps | Average Run Time (seconds) | Average Steps | Average Run Time (seconds) | Average Steps | Average Run Time (seconds) |
| 10 | 298 | 6 | 89 | 2 | 178 | 4 |
| 100 | 304 | 10 | 132 | 8 | 151 | 10 |
| 1 000 | 420 | 14 | 162 | 11 | 234 | 13 |
| 10 000 | 473 | 19 | 174 | 13 | 302 | 18 |
| 100 000 | 462 | 26 | 290 | 18 | 356 | 21 |

The data suggests that as more puzzles come in, the more steps the program will take and a slow increase in run time. As expected, the second heuristic, the Manhattan Distance, performs better than the first heuristic. Additionally, we can see that the second heuristic performs much better than both heuristics combined as well. Ideally, running both heuristics at once would generate a tight middle-point result, but we can see it is almost there, but still not quite.

Admittedly, the number of steps that is taken is rather erratic. In fact, I’m not even positive if that is the correct value. In the output, I noticed that some nodes that were being outputted were repeated. It would be safe to say that the step count is quite inflated, but unfortunately I have no way to discover that by how much at this moment. I also noticed from the table that the time taken is quite low, but this may be due to my own machine power.

My approach to this project was pretty straight-forward. At first, I started with figuring out methods to calculate both heuristic values individually, and that did take up some time. I then proceeded to start up on A\*, and, unfortunately, the A\* itself took much, much more than I’d imagined.