To begin let’s examine the data comparing all four types of searches. These problems were chosen to be relatively simple, though the last problem ran was just one step down in complexity than the next data set. From this data I conclude that, even though it would take more time in some problems, Breadth-First Search will achieve the same path as A\*. However, A\* is the more efficient algorithm, in half of the cases it had a significantly better time than BFS and in all them A\* expands far fewer nodes than BFS. Depth-First Search is another matter entirely. DFS typically has a path cost equal to the number of nodes that were examined however, as you can see from the table in two of the problems this was not the case. This is because when DFS is generating the successors it will not add a Node that is already a part of the path, thus if the only successors of a Node are nodes that are already in the path than it will have expanded that node, but does not add to the path cost and begins to look at neighboring Nodes for successors that will get us closer to the Goal State. DFS is consistent that it never has a path cost that is better than BFS or A\*, this mainly comes from the way in which the EightPuzzle problem picks its moves. Even though its paths are longer, it does only keep the Nodes in memory that are closer to the Goal State, whereas BFS keeps all Nodes in memory. This is the fundamental difference between these two searches, though the Data cannot show this since we are only looking at the path cost and the Nodes expanded to get there. Based on this evidence if you are looking for a minimal path you will likely want to use BFS rather than DFS, if you are worried about space you would choose DFS over BFS. A\* would be the best pick for solving any problem, but the objective of this portion was too compare BFS and DFS. Next we will look at the last four problems, which I considered to be far more difficult than the problems that were run with BFS and DFS.

For comparing the two A\* heuristics four problems were chosen based on the number of moves (i.e path cost) needed to solve them. The easiest problem in this set required a minimum of 21 moves to complete, with the hardest requiring 25 moves. These particular problems were chosen as during testing it appeared that any more than 25 moves and Tiles Out of Place (TOP) to a far more significant time to complete than Manhattan Distance. Even though these problems do not require such a lengthy time for TOP they still show the difference between the two heuristics. In every problem both heuristics were able to give a solution of the same quality (same path cost). However, they were not able to do it in the same amount of time or with expanding the same amount of Nodes. Thus, based on the data gathered and the analysis of said data the opinion of this author is that only a single conclusion can be drawn, and that is that A\* with Manhattan a superior combination than A\* with TOP. Admittedly, the implementation of this A\* search and the heuristics is likely not the optimal implementation. Therefore, given a better implementation the conclusion drawn here might change, yet, based on the data in the table provided the only conclusion the author can draw is the above.