I used the A* algorithm for the project. To make this work the nodes of the graph were individual states of the puzzle and the algorithm searched the next state based on which possible moves had the best heuristics. I implemented my algorithm with a TreeSet instead of a priority Queue which automatically polled the state with the best score. The algorithm solves grids from 3x3 up to 6x6 with increasing time lengths required. It should theoretically be able to solve 7x7 grids given enough time but it is very memory intensive as the memory required for A* grows exponentially. Heuristics calculated with the Manhattan distance. While simple it ultimately was the best solution I found. I tested many heuristics involving prioritizing the states that found the next number to be solved. For example, if 1,2,3 were in the correct spot the heuristic would value moves that moved 4 closer to their destination over all else. While this implementation worked exceptionally in some cases such as board1.txt which it solved perfectly with 14 moves, it fell short of Manhattan's consistency. The Hardest part was definitely just starting the project and thinking about how to conceptualize a graph made with nodes that are graphs. And figuring out how to implement the A* algorithm for my needs. The biggest problem I ran into was with hashing adjacent vertices because I kept getting the hashcode for the original graph so no new nodes would be added to trees because it though they were duplicates. Overall it was probably the most fun CS project I have ever done and I have a huge sense of pride and accomplishment for my work.