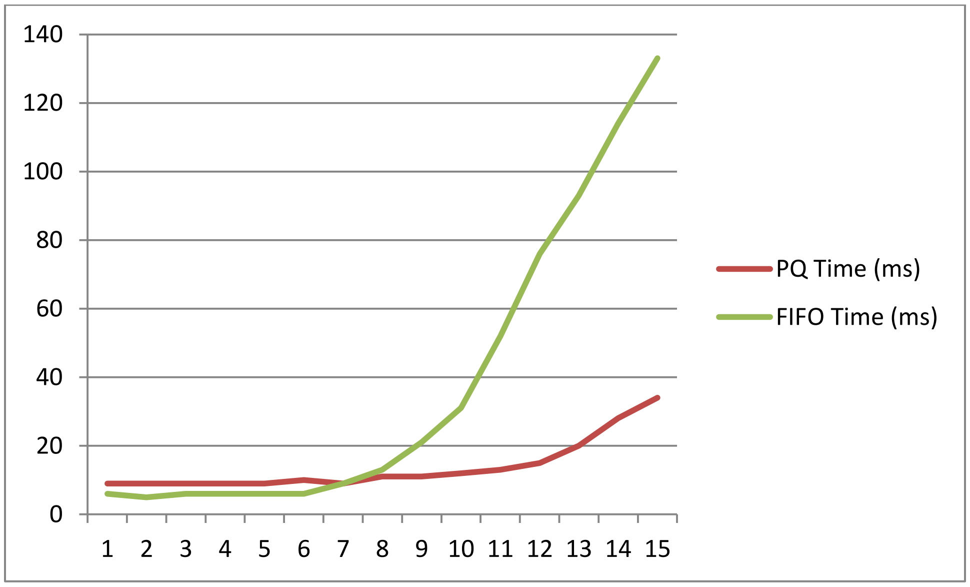
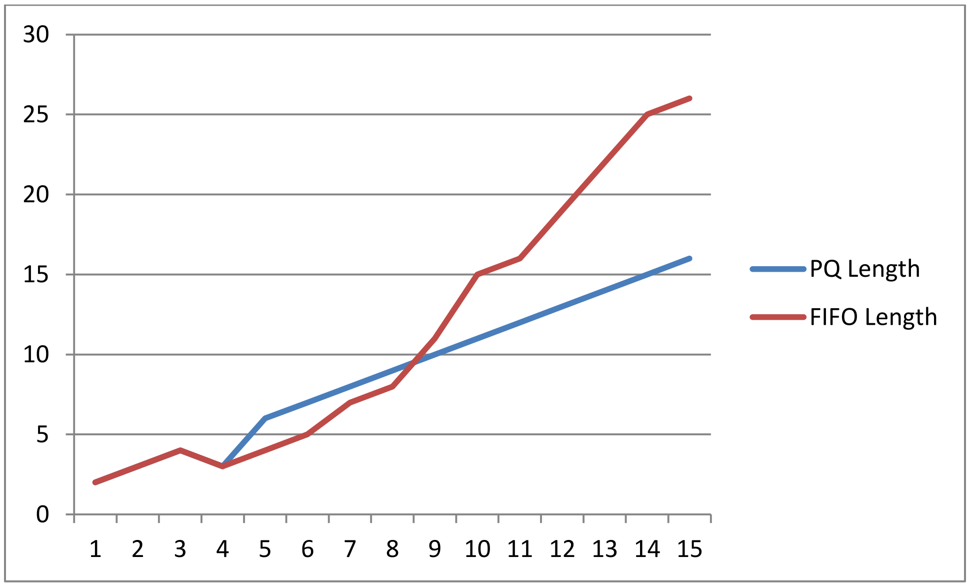
Jack Cobb
905659469

Project 3 Report

1. I found that Priority Queue was much faster in producing a correct solution path vs the FIFO implementation, though FIFO produced the shortest sequence of moves to get to the solution. This is because FIFO uses a true breadth first implementation for checking whereas a Priority Queue is a "smart" interpretation of BFS. It uses the manhattan distance to determine which direction is "closer" to the goal state. It doesn't produce the shortest solution path most of the time because the manhattan distance isn't the best heuristic for determining the best moves to the solution.





2. My overall design for this project was that I wanted my class representing the puzzle (PuzzleState) to handle most of the methods dealing with changing the board, possible moves, setting the board, and creating various strings of the current board layout. For example, to reduce

Jack Cobb
905659469

code complexity in later use, my PuzzleState class returned a list of valid moves of the empty slot, then my move method took one of those numbers and moved in that specific direction, returning a new board that was in the desired state.

For my BFS algorithm, I used a while loop instead of recursion because it made more sense to me for a more exhaustive search of all the neighbors until a goal state was found. If it were a DFS, I would have used recursion to check all of a state's children before moving to the next neighbor PuzzleState.