CSCI 561 Joshua Villbrandt
A1 Sept. 19, 2011

Question 1

The heuristic I used is the sum of the Manhattan distances for each tile. The Manhattan distance is effectively a measure of how out of place a given tile is. This number is calculated as the sum of the difference of the components between a tile's current position and a tile's goal position. In other words, a tile in position (x, y) with goal position (x_{goal}, y_{goal}) would have a Manhattan distance of $abs(x_{goal} - x) + abs(y_{goal} - y)$.

The Manhattan distance is an admissible heuristic, because it never over-estimates the cost from one state to the goal state. This is easiest to understand on the tile-level. The distance for a single tile to that tile's goal position is never over-estimated with the Manhattan distance. It is, in fact, the best-case distance to its goal position. It is likely that, in the solution for a given initial puzzle, a particular tile might have to take an indirect route in the most efficient sequence for the puzzle as a whole. With this in mind, it is easy to see how, since this heuristic is the sum of the Manhattan distance for all of the tiles on the board, that the cost for a state as a whole is never over-estimated.

Question 2

The 8 Puzzle becomes unsolvable when pieces are swapped without moving the empty space. Actually, puzzles with an even number of swaps are solvable, because two swaps given the empty space a change to move a number without getting trapped. Catching unsolvable puzzles could come in two flavors: detecting odd permutations from the solution or giving up when the maximum number of moves is exceeded.

To discover if a puzzle is an odd permutation, a second A* search could be implemented that plays the game by swapping tiles instead of moving the empty space. When swapping tiles, the cost to the goal state should never increase after a move. By iterating through this tree, the search would error out when the solution was not found and the cost to the goal state started increasing. This would prevent having to search through the much wider search tree. There is no solution to the puzzle if this search errors out. Otherwise, the puzzle is solvable.

The other method to determine if a puzzle is unsolvable would be to error out when the search depth surpasses the maximum number of moves needed in an optimum solution for any puzzle. This number is 31 for the 8 Puzzle (Wikipedia – 15 puzzle.)

Question 3

This program was tested towards three goals. The first goal was simply to verify that the program would run given a very easy input case (testInput1.) Taking the solution state and moving the empty tile two spaces yielded a suitable test case for this goal.

The second goal was to make sure that the program was operating efficiently. Specific areas of concern were long execution times. The test case (testInput2) was generated in the same manner as before, but with many more moves of the empty space.

The final goal was to make sure that the program was computing the optimum solution. Making valid moves from the solution state generated a puzzle. More moves were taken than is needed for an optimum solution by any 8 puzzle (testInput3.) This loosely proves that the program is making an optimum solution is it stays under 32 moves.