Joyce Tan
CSCI 363 – Artificial Intelligence
Programming Assignment #1

Heuristic Search - Problem Analysis

Algo: A* search where h(n) = the number of misplaced tiles

Start State	**# of expanded nodes**	**Total time (in milliseconds)**	**Sequence of moves**
Easy	7	3	up right up left down
Medium	94	12	up right right down left left up right down
Hard	738	333	right up left up right right down left left up right down
Worst	Takes too long to solve		

Algo: A* search using the Manhattan heuristic function

Start State	**# of expanded nodes**	**Total time (in milliseconds)**	**Sequence of moves**
Easy	6	1	up right up left down
Medium	19	0	up right right down left left up right down
Hard	85	8	right up left up right right down left left up right down
Worst	12178	60040	right down left left up up right right down down left left up up right right down down left left up up right right down down left left up right

Algo: Iterative deepening A* with the Manhattan heuristic function

Start State	**# of expanded nodes**	**Total time (in milliseconds)**	**Sequence of moves**
Easy	6	8	up right up left down
Medium	55	228	up right right down left left up right down
Hard	181	888	right up left up right right down left left up right down
Worst	Out of memory error		

Algo: Depth-first Branch and Bound with the Manhattan heuristic function

Start State	# of expanded nodes	Total time (in milliseconds)	Sequence of moves
Easy	15	0	up right up left down
Medium	Out of memory error		
Hard	Out of memory error		
Worst	Out of memory error		

1. What is the number of possible states of the board?

   <u>Answer:</u> 9! possible states

2. What is the average number of possible moves from a given position of the board?

   <u>Answer</u>: From the middle tile, the number of possible moves = 4
   From the four corners tiles, the number of possible moves = 2
   From the remaining four tiles on the middle edges, the number of possible moves = 3
   Average number of possible moves = 24/9 = 2.67

3. Estimate how many moves might be required for an optimal (minimum number of moves) solution to a "worst-case" problem (maximum distance between starting and goal states). Explain how you made your estimate (Note this is an open-ended question; any logical answer may suffice).

   <u>Answer:</u> Minimum # of moves = 30 moves
   First you can move the empty tile in any direction left, down, right, or up
   = +1 move.
   Then you move the empty tile along the edges of the board in one direction either keep moving left or keep moving right until all the tiles are in the right place other than the middle tile that was moved during the first move. Each tile is 4 moves away from their goal state.
   = 7 tiles * 4 moves = +28 moves.
   Finally you swap the empty tile with the middle tile that was swapped with the empty tile during the first move.
   = +1 move

4. Assuming the answer to question #2 is the "average branching factor" and a depth as in the answer to question #3, estimate the number of nodes that would have to be examined to find an answer by the brute force breadth-first search.

Answer: Average branching factor = b = 2.67
depth = d = 30
estimate # of nodes = summation of b^d = summation of 2.67^{30} = $9.98 * 10^{12}$

5. Assuming that your computer can examine one move per millisecond, would such a blind-search solution to the problem terminate before the end of the semester?

Answer: There is about 75 days left in the semester, which is equal to $6.48 * 10^9$ milliseconds. Therefore this blind solution to the problem will not terminate before the end of the semester because we need $9.98 * 10^{12}$ milliseconds to examine all the nodes.

6. The "worst" example problem given above is actually one of the easiest for humans to solve. Why do you think that is the case? What lessons for AI programs are suggested by this difference between human performance and performance of your search program?

Answer: In the "worst" example, a human can observe the problem and notice patterns. Humans may be able to notice the optimal solution without going through every possible path to find a solution. The AI programs uses heuristic functions to try to mimic human performance, therefore these programs depend heavily on how accurate the heuristic functions are.

7. Compare A*, DFBnB, and IDA* and discuss their advantages and disadvantages.

Answer: A*'s advantage is that it is optimally efficient because this algorithm expands the fewest amount of nodes to get the solution. A*'s disadvantage is that A*'s complexity is dependent on how good/accurate the heuristic function for calculating h(n). DFBnB's advantage is its space complexity is linear. DFBnB's disadvantage is that you will have to expand a large amount nodes if you have not reached the optimal solution with a low upper bound early on in the search. IDA*'s advantage is that its space complexity is linear. IDA's disadvantage is that it looks repeats large parts of the search several times to find a solution.