MISPLACED HEURISTIC

EASY BOARD

```
Optimal Solution Nodes: 5
Total Nodes Expanded: 6
Total Time: 17 miliseconds
```

MEDIUM BOARD

```
Optimal Solution Nodes: 9
Total Nodes Expanded: 26
Total Time: 3 miliseconds
```

HARD BOARD

```
Optimal Solution Nodes: 12
Total Nodes Expanded: 87
Total Time: 2 miliseconds
```

WORST BOARD

```
Optimal Solution Nodes: 30
Total Nodes Expanded: 107819
Total Time: 670 miliseconds
```

MANHATTAN HEURISTIC

EASY BOARD

```
Optimal Solution Nodes: 5
Total Nodes Expanded: 5
Total Time: 0 miliseconds
```

MEDIUM BOARD

```
Optimal Solution Nodes: 9
Total Nodes Expanded: 15
Total Time: 0 miliseconds
```

HARD BOARD

```
Optimal Solution Nodes: 12
Total Nodes Expanded: 25
Total Time: 0 miliseconds
```

WORST BOARD

```
Optimal Solution Nodes: 30
Total Nodes Expanded: 1013
Total Time: 6 miliseconds
```

IDA*

EASY BOARD

```
Optimal Solution Nodes: 5
Total Nodes Expanded: 24
Total Time: 0 miliseconds
```

MEDIUM BOARD

```
Optimal Solution Nodes: 9
Total Nodes Expanded: 212
Total Time: 0 miliseconds
```

HARD BOARD

```
Optimal Solution Nodes: 12
Total Nodes Expanded: 981
Total Time: 0 miliseconds
```

WORST BOARD

```
Optimal Solution Nodes: 30
Total Nodes Expanded: 180940
Total Time: 1121 miliseconds
```

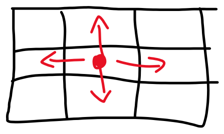
1. What is the number of possible states of the board?

The number of possible states of the board is 9!, I got this by doing the permutation nPr, there are 9 tiles, and 9 different ways to arrange them, so given the permutation formula we would get 9!/(9-9)! which would give us 9!.

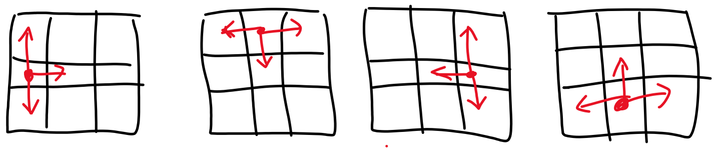
And the number of solvable boards is 9!/2

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

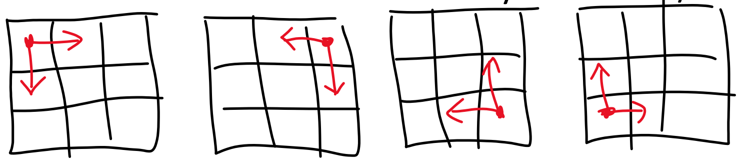
The most possible number of moves would be 4, which in the 8 puzzle board would be if the blank tile is in the middle, where it can move up, down, left and right. Probability of this is 1/9



There can also be 3 number of moves which would be if the blank tile is located middle-side of the board. Probability of this is 4/9



The other possibility is 2 number of moves, this can happen when the blank tile is on one of the four corners. Probability of this is 4/9



(4*(1/9)) + (3*(4/9)) + (2*(4/9))

= .44+1.33+.88

=2.66 possible number of moves

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).

If we use the manhattan distance and applied it to our root note, lets say initially the manhattan distance is 24, which is the case of our worst board, we can assume that each move that we make brings us closer one move closer to the goal, so each move we make we can assume is -1, with that logic the optimal path would be some what similar to that of the manhattan distance that we found for the root node.

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.

Since brute force breadth-first search is checking everything level by level, and we assumed that the average branching factor is 2.66, we can say that the number of nodes visited would be, 2.66(1) + 2.66(2) + 2.66(3) + … + 2.66(level).

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?

NO! if it keeps seeing duplicates then a blind search would keep going back and forth between the duplicates.

1000 * 60 * 60* 24 * 75 = 6.48*10^9

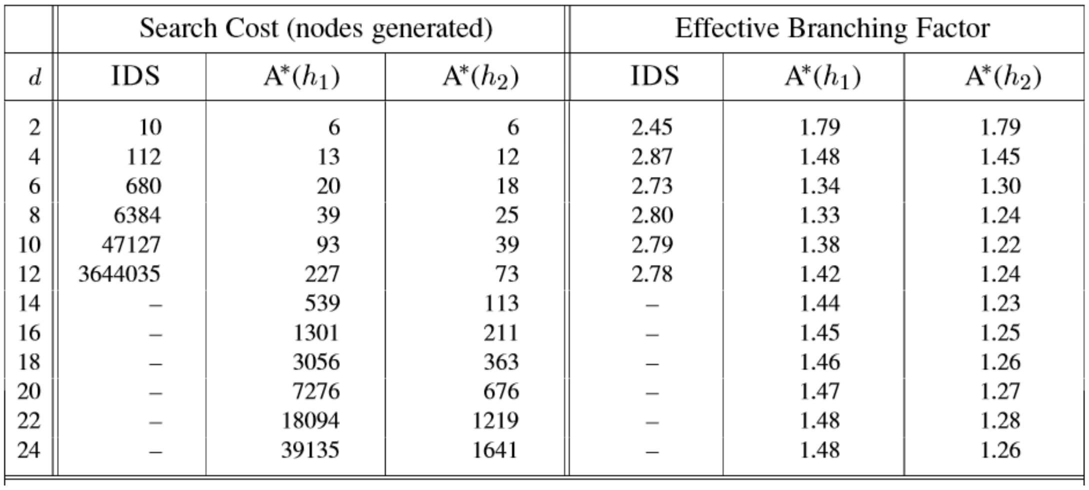
2.66^(24) > 6.48*10^9

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?

What makes this the worst case is the fact that all the tiles are in the opposite location as compared to the goal state. For a human it is easy to see that we just need move the board in a circle like motion. But for a computer all it see’s is the current board and the goal. And it is only able use distance(Manhattan) or misplaced to help it distinguish the best choices. So a computer isn’t able to see the “bigger” picture, where as a human can use logic and understand that the board is in placed in the opposite of the goal.

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

A* is has the least efficiency in space, A* keeps a large queue of unexplored nodes that can quickly grow in size, while IDA* is one of the most efficient in space that is because it does not remember any other nodes besides the ones on the current path, so memory usage is the same as the most optimal path. One of the disadvantages of IDA* is the average branching factor, but this is one of the advantages of A* which has a much lower branching factor. One of the advantages of DFBnB is it keeps track of the lowest path found so far, it is also good in memory management as it discards those nodes who have a higher costBound.



	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26