CSE537 Artificial Intelligence
Assignment-1

Furqan Baig (109971799)

October 9, 2016

1 Heuristics

I have used following heuristics in this assignment for A$*$ and IDA$*$ algorithms

1.1 Hamming Distance

A simple heuristic function h = *No. of misplaced tiles in current state*. The number of moves required to reach goal state from given a given state would require each misplaced tiles to be moved atleast once. Therefore, the number of moves to reach goal state would always be more than or equal to the number of misplaced tiles. Therefore Hamming distance as cost is never an over estimate and thus is an admissible heuristic.

1.2 Manhattan Distance

This cost function is defined as h = *Sum of Manhattan distances of all tiles from their goal state position.* Since the only moves that we can make in our particular problem are up, down, right and left, the manhattan distance of a tile to its goal state is always consistent and thus is admissible.

2 Memory Issue with A$*$

A$*$ although optimal is not memory efficient since it has to keep track of visited states to avoid re-evaluation. However, the number of nodes to track needs to be kept in memory and can exponentially increase with the depth of the solution in the search tree. In addition, A$*$ in essence relies on the same principles of Breath First Search. Since BFS is not memory efficient therefore A$*$ is not as well. The memory of A$*$ is dependent on the branching factor and the depth of the solution. In particluar the memory required for A$*$ is bounded by $O(b^d)$

3 Memory Bounded Algorithm

In my solution I used IDA$*$ as the memory bounded algorithm for 15-tile puzzle. IDA$*$ is a variant of iterative deepening search. In iterative deepening search, all branches are explored in a Depth First Search style for a given depth. If solution is not found, the depth is increased iteratively and the search is started again. IDA$*$ uses the same principle but the depth is determined using the heuristic function, Manhattan Distance in our case. In particular the depth is determined using the

f function similar to A$*$ as $f(n) = g(n) + h(n)$. During a recursion, minimum f value is maintained and returned as the new depth for the next iteration. No visited states are kept from previous recursion to avoid the memory pressure.

3.1 Optimality

IDA$*$ is optimal based on similar arguments as of A$*$ given admissible heuristic. IDA$*$ traverses all paths till calculated depth threshold and stops when expanding the goal node. At the same time, admissibility ensures that all other paths with lower costs have already been examined. Thus the path selected by IDA$*$ is the lowest cost path and therefore is optimal.

3.2 Completeness

IDA$*$ is essentially a Depth First Search which filters away nodes having f value greater than a given threshold. Since f value is derived directly from the heuristic value, given an admissible heuristic, nodes that lead to goal will never be filtered and IDA$*$ is guaranteed to find a solution if it exists.

4 A$*$ Performance

<table>
<tr><th colspan="3"></th><th colspan="3">Hamming Distance</th><th colspan="3">Manhattan Distance</th></tr>
<tr><th colspan="3">Start State</th><th>Explored States</th><th>Time</th><th>Depth</th><th>Explored States</th><th>Time</th><th>Depth</th></tr>
<tr><td>4</td><td>1</td><td>5</td><td rowspan="3">34</td><td rowspan="3">0.059</td><td rowspan="3">8</td><td rowspan="3">20</td><td rowspan="3">0.053</td><td rowspan="3">8</td></tr>
<tr><td>2</td><td>5</td><td>6</td></tr>
<tr><td></td><td>7</td><td>8</td></tr>
<tr><td></td><td>1</td><td>3</td><td rowspan="3">9</td><td rowspan="3">0.048</td><td rowspan="3">4</td><td rowspan="3">9</td><td rowspan="3">0.047</td><td rowspan="3">4</td></tr>
<tr><td>4</td><td>2</td><td>6</td></tr>
<tr><td>7</td><td>5</td><td>8</td></tr>
<tr><td>5</td><td>1</td><td>3</td><td rowspan="3">32</td><td rowspan="3">0.54</td><td rowspan="3">8</td><td rowspan="3">20</td><td rowspan="3">0.051</td><td rowspan="3">8</td></tr>
<tr><td>2</td><td></td><td>6</td></tr>
<tr><td>4</td><td>7</td><td>8</td></tr>
<tr><td>1</td><td>8</td><td>3</td><td rowspan="3">1018</td><td rowspan="3">0.312</td><td rowspan="3">16</td><td rowspan="3">318</td><td rowspan="3">0.097</td><td rowspan="3">16</td></tr>
<tr><td>4</td><td>5</td><td>2</td></tr>
<tr><td></td><td>7</td><td>6</td></tr>
<tr><td>4</td><td>6</td><td>1</td><td rowspan="3">1087</td><td rowspan="3">0.353</td><td rowspan="3">16</td><td rowspan="3">157</td><td rowspan="3">0.068</td><td rowspan="3">16</td></tr>
<tr><td>3</td><td></td><td>5</td></tr>
<tr><td>7</td><td>2</td><td>8</td></tr>
<tr><td>3</td><td>6</td><td>2</td><td rowspan="3">975</td><td rowspan="3">0.306</td><td rowspan="3">16</td><td rowspan="3">195</td><td rowspan="3">0.070</td><td rowspan="3">16</td></tr>
<tr><td>4</td><td></td><td>8</td></tr>
<tr><td>1</td><td>7</td><td>5</td></tr>
<tr><td>1</td><td>3</td><td>8</td><td rowspan="3">382</td><td rowspan="3">0.099</td><td rowspan="3">14</td><td rowspan="3">70</td><td rowspan="3">0.054</td><td rowspan="3">14</td></tr>
<tr><td>7</td><td>2</td><td>4</td></tr>
<tr><td>5</td><td>6</td><td></td></tr>
<tr><td>4</td><td>6</td><td></td><td rowspan="3">2097</td><td rowspan="3">1.088</td><td rowspan="3">18</td><td rowspan="3">405</td><td rowspan="3">0.122</td><td rowspan="3">18</td></tr>
<tr><td>7</td><td>3</td><td>1</td></tr>
<tr><td>8</td><td>5</td><td>2</td></tr>
</table>

1 6 4	2 3 7	 5 8	443	0.117	14	150	0.068	14
1 7 2	3 5	8 6 4	4943	5.354	20	635	0.193	20
 3 2	4 1 6	8 5 7	30601	295.88	24	1807	0.928	24
2 5 6	7 1 8	4 3 	13370	53.00	22	878	0.288	22
8 6 1	5 2	4 7 3	72049	2199.25	26	2604	1.783	26
4 6 	1 3 5	7 8 2	12687	48.056	22	623	0.185	22
7 2 3	6 1 8	0 4 5	29506	327.67	24	1247	0.516	24
1 3 2	7 0 8	4 5 6	35526	491.89	24	2324	1.427	24
 8 6	7 2 1	3 5 4	12569	40.81	22	546	0.163	22
3 8 	5 7 4	2 6 1	12856	47.66	22	1402	0.590	22
4 3 7	5 2	1 8 6	4910	5.802	20	567	0.171	20
7 1 5	4 8	2 6 3	989	0.317	16	242	0.081	16