CSE537 HW1

Sandeep Kumta Vishnu (111482809)

1. Heuristics

The two Heuristics used in the implementation are Misplaced Tiles Heuristics and Manhattan Distance Heuristics

1.1 Misplaced Tiles Heuristics

The misplaced tiles heuristic basically gives the number of misplaced tiles in a particular state compared to the goal state. Here there is no strict bound that the space need to be empty to move a tile or whether it is how far from the goal state. The tiles are allowed to move to any space.

In the best case when the tile is misplaced by a single empty adjacent space, this heuristic estimates the actual distance exactly. Other than the best case, in all other scenarios, the estimation is much lesser than the actually number of moves required to move the tiles to the goals state.
This heuristic underestimates the moves need to solve the puzzle. Since it is always gives less moves than the actual optimal solution, it is proved to be consistent

1.2 Manhattan Distance Heuristics

This heuristic basically gives the total number of moves needed to place all the elements in the grid to its goal state in a strictly horizontal and/or vertical path as opposed to diagonal path since diagonal path is not allowed.

Even This heuristic relaxes the strictness by allowing move the tile to adjacent position no matter whether it is empty or occupied. Because of this, a series of moves needed to actually solve the real problem without any relaxations are not considered here.

So, In the best case, i.e when the adjacent tile is empty and exactly the correct position and there exists only one misplace tile, this heuristic estimates the actual distace needed exactly. Other than the best case, in all other scenarios, the estimation is much lesser than the actually number of moves required to move the tiles to the goals state.
This heuristic underestimates the moves need to solve the puzzle. Since it is always gives less moves than the actual optimal solution, it is proved to be consistent.

1.3 Why h1 dominates h2?

Since h(n) is the estimation of minimum cost from current state to the goal state, as h(n) decreses its effect on f(n) which f(n)=g(n)+h(n). i.e in that case f(n) is less dependent on h(n).

Therefore as h(n) decreases, more nodes need to be expanded in A* algorithm.

So, if we prove h1(n) is always greater than h2(n) then, it implies h1 dominates h2. ==> Equation (1)

Now,
If a tile is misplaced, it will add value of 1 to the total heuristic evaluation, while its Manhattan distance is at least 1. If it's not misplaced, both are 0. Also, because of very relaxed strictness factor in Misplaced Heuristic as explained above the cost of placing it to the goal state is very less compared to the manhattan distance heuristic where on vertical or horizontal adjacent moves are possible. This makes the number of moves needed to move a tile to its correc position increase because of the restriction.

Therefore h1(n) >= h2(n).

Equating the equation (1), we proved that h1 always dominates h2.

2. Memory Issues with A*

In A* seach for each node, we find all possible successors and add it to the queue. So, for each best node we need to keep 3 extra useless nodes in the memory.

The frontier will be in memory and the nodes are not tested for goal till they are considered for expansion.

The upper bound on the memory needed in 4 X 4 puzzle is exponential interms of depth of the optimal solution. i.e 4^d because each state
has 4 possible successor nodes.

The instance took 7 GB of memory while running in my Dell XPS machine.

3. Iterative Deepening A*

IDA* is a uninformated depth first search which eliminates the successors when the current node possesses f-cost value greater than the current node's depth limit. The ancestor node will maintain this f-cost value of the eliminated node and the recursion continues with all the other successors.

It finds the smallest f-cost which is more than the depth-limit. If there is no node which has f-cost value more than the limit then there is no solution exists with that successor. The smallest f-cost which is more than the depth-limit will now become the limit and so on. If there is admissible heuristic function like manhattan distance heuristic or the misplaced tiles heuristic then IDA* algorithhm is guaranteed to give the optimal solution.

The space complexity of the IDA* is O(d) where d is the depth of the optimal solution. Because there will be atmost d nodes in the path and this algorithm keeps single path in memory at each time.

So, memorywise we get a lot of performance improvements in IDA* compared to A* algorithm.

4. A table describing the performance of your A* and memory-bounded implementations

4.1 Manhattan Distance Heuristics

Initial State	States Explored	Time (ms)	Depth
0 1 2 4 5 3 7 8 6	5	0.0	4
1 2 3 4 5 6 7 0 8	2	0.0	1
1 2 3 4 5 6 0 7 8	3	0.0	2
4 1 3 0 2 6 7 5 8	6	0.0	5
4 1 3 7 2 6 0 5 8	7	0.0	6
4 1 2 5 8 3 7 0 6	8	0.0	7
4 1 2 5 0 3 7 8 6	7	0.0	6
1 2 3 4 6 8 7 0 5	6	0.0	5
2 4 3 1 0 6 7 5 8	9	0.0	6
1 0 3 4 2 6 7 5 8	4	0.0	3
1 2 3 4 5 6 7 8 0 13 10 11 14 9 15 12	12	0.0	8
5 1 8 3 0 2 6 4 9 10 7 11 13 14 15 12	15	0.0	11
1 2 3 4 5 0 11 8 10 7 6 12 9 13 14 15	16	0.0	10
1 2 3 4 5 6 7 8 10 11 12 0 9 13 14 15	8	0.0	7
5 1 3 4 2 0 6 8 9 10 7 11 13 14 15 12	13	0.0	8
1 3 4 7 5 2 6 8 13 9 10 0 14 15 12 11	56	1.3	17
1 2 3 4 5 10 6 7 9 14 11 8 13 0 15 12	7	0.0	6
1 2 7 3 5 6 4 8 9 10 11 0 13 14 15 12	11	0.0	7
1 2 3 4 6 7 11 8 0 5 14 12 13 9 10 15	21	0.0	12
5 1 2 3 0 6 8 4 9 10 7 12 13 14 11 15	10	0.0	9

4.2 Misplaced Tiles Heuristics

Initial State	States Explored	Time (ms)	Depth
0 1 2 4 5 3 7 8 6	5	0.0	4
1 2 3 4 5 6 7 0 8	2	0.0	1
1 2 3 4 5 6 0 7 8	3	0.0	2
4 1 3 0 2 6 7 5 8	6	0.0	5
4 1 3 7 2 6 0 5 8	7	0.0	6
4 1 2 5 8 3 7 0 6	8	0.0	7
4 1 2 5 0 3 7 8 6	7	0.0	6
1 2 3 4 6 8 7 0 5	9	0.0	5
2 4 3 1 0 6 7 5 8	10	0.0	6
1 0 3 4 2 6 7 5 8	4	0.0	3
1 2 3 4 5 6 7 8 0 13 10 11 14 9 15 12	15	0.0	8
5 1 8 3 0 2 6 4 9 10 7 11 13 14 15 12	25	0.0	11
1 2 3 4 5 0 11 8 10 7 6 12 9 13 14 15	26	0.0	10
1 2 3 4 5 6 7 8 10 11 12 0 9 13 14 15	8	0.0	7
5 1 3 4 2 0 6 8 9 10 7 11 13 14 15 12	14	0.0	8
1 3 4 7 5 2 6 8 13 9 10 0 14 15 12 11	290	15.625	17
1 2 3 4 5 10 6 7 9 14 11 8 13 0 15 12	7	0.0	6
1 2 7 3 5 6 4 8 9 10 11 0 13 14 15 12	15	0.0	7
1 2 3 4 6 7 11 8 0 5 14 12 13 9 10 15	68	0.0	12
5 1 2 3 0 6 8 4 9 10 7 12 13 14 11 15	10	0.0	9