Part1:

h1 = h_distance= sum of distances of the tiles to their end positions.

h2 = h_misplaced= # of misplaced tiles.

Therefore, when running the program with <N> = 1, h_distance is used. And when running the program with <N> = 2, h_misplaced is used.

Let n be the current node, and n’ be a successor of n, and act be a move.

cost(n, act, n’) = 1 for this tile puzzle problem.

For h1:

h1(n’) can be h1(n) + 1 or h1(n) – 1 since a single move act can affect only one tile. And it can make this tile 1 position further or closer to its end position. Thus,

$h1(n') + cost(n, act, n') = h1(n)-1+1$ or $h1(n') + cost(n, act, n') \leqslant h1(n)+2$

So $h1(n) \leqslant h1(n') + cost(n, act, n')$. Therefore, h1 is consistent.

For h2:

h2(n’) can be h2(n) + 1 or h2(n) or h2(n) – 1 since a single move act can affect only one tile. And after this move, this tile can have one of the following three state changes:

corret_pos->wrong_pos

wrong_pos1->wrong_pos2

wrong_pos->correct_pos

So with similar proof as h1, $h2(n) \leqslant h2(n') + cost(n, act, n')$. Therefore, h2 is consistent.

h1 dominates h2:

If a tile t is in the correct position, it contributes 0 to both heuristics. But if t is in a wrong position, it contributes 1 to h2, while it can contribute more than 1 to h1. For example if tile of number 1 is in location board[1][2], it contributes 1 to h2’s calculation, while it contributes 3 to h1’s calculation. Therefore, for any node n, $h1(n) \geqslant h2(n)$. In other words, h1 dominates h2.

Part2:

A* search algorithm has to keep a large number of unexplored nodes in the priority queue. So, it can quickly fill up memory. For 15-puzzle, I need $O(b^d)$ memory space, where d is the depth of the solution, and b is the average number of successor per state, which is 3.

Part3:

I used IDA* search algorithm for this assignment. IDA* only keeps the nodes on the current path. Therefore, it requires an amount of memory that is only linear in the length of the solution that it constructs. Therefore, space complexity of IDA* is O(d), where d is the depth of the solution.

IDA* finds the shortest path from start node to goal node when the heuristic function h is admissible. So IDA* is both complete and optimal.

With higher h(n) value, time complexity is higher, since it will explore more states with higher h(n) value.

Part4:

Time t is in unit ms.

	A* with h1	A* with h2	IDA* with h1	IDA* with h2
8-puzzle-1	s=4 t=0.5 d=4	s=4 t≈0 d=4	s=83 t=2 d=4	s=6 t=0.5 d=4
8-puzzle-2	s=2 t≈0 d=2	s=2 t=0.5 d=2	s=21 t=0.5 d=2	s=4 t≈0 d=2
8-puzzle-3	s=1 t=0.5 d=1	s=1 t≈0 d=1	s=6 t=0.5 d=1	s=2 t≈0 d=1
8-puzzle-4	s=5 t=1 d=5	s=5 t=0.5 d=5	s=152 t=3.3 d=5	s=6 t≈0 d=5
8-puzzle-5	s=4 t=0.5 d=4	s=5 t=1.5 d=4	s=88 t=1.9 d=4	s=11 t=0.5 d=4
8-puzzle-6	s=29 t=5.5	s=60 t=13	s=4687 t=137.4	s=128 t=2

	d=11	d=11	d=11	d=11
8-puzzle-7	s=3 t≈0 d=3	s=3 t≈0 d=3	s=45 t=1 d=3	s=5 t=0.7 d=3
8-puzzle-8	s=3 t=0.5 d=3	s=3 t=0.5 d=3	s=60 t=2 d=3	s=6 t=0.5 d=3
8-puzzle-9	s=4 t=1 d=4	s=4 t=0.5 d=4	s=91 t=1.5 d=4	s=7 t=0.2 d=4
8-puzzle-10	s=78 t=17 d=20	s=2794 t=830.7 d=20	s=1038662 t=24090 d=20	s=19093 t=321.9 d=20
15-puzzle-1	s=3 t=0.5 d=3	s=3 t=0.5 d=3	s=71 t=1.5 d=3	s=5 t=0.5 d=3
15-puzzle-2	s=39 t=23.6 d=16	s=229 t=59.1 d=16	s=738077 t=23792.4 d=16	s=611 t=15.8 d=16
15-puzzle-3	s=3 t=0.5 d=3	s=3 t=1.5 d=3	s=63 t=1.5 d=3	s=6 t=0.5 d=3
15-puzzle-4	s=1 t≈0 d=1	s=1 t≈0 d=1	s=7 t≈0 d=1	s=2 t≈0 d=1
15-puzzle-5	s=2 t≈0 d=2	s=2 t=0.5 d=2	s=30 t=1 d=2	s=4 t=0.5 d=2
15-puzzle-6	s=12 t=3 d=8	s=13 t=1.5 d=8	s=3288 t=229.1 d=8	s=24 t=0.9 d=8
15-puzzle-7	s=52 t=19.6 d=13	s=134 t=38.1 d=13	s=193939 t=6646.4 d=13	s=592 t=23 d=13
15-puzzle-8	s=3 t=0.5 d=3	s=3 t=0.5 d=3	s=85 t=3.5 d=3	s=6 t≈0 d=3
15-puzzle-9	s=5 t=1.5 d=5	s=8 t=1.5 d=5	s=370 t=10 d=5	s=23 t=0.5 d=5
15-puzzle-10	s=8 t=2 d=8	s=8 t=1 d=8	s=4669 t=124.8 d=8	s=15 t≈0 d=8