CSI4106 - Assignment 1

by Oliver Fei (7624001) and Jonathan Guillotte-Blouin (7900293)

1. Environment Properties

- **Completely observable**: the agent always has access to the position of each tile which consists of all the pertinent information of the environment.
- **Deterministic**: given a current state (arrangement of tiles) and an action (a tile swaps with the empty tile), the tile swap will always result in the same state.
- **Sequential**: the choice of action will effect the next action as different actions lead to different board states which will effect the next chosen action.
- **Static**: the positions of each tile do not change while the agent decides on its action.
- **Discrete**: there is a finite number of ways that tiles can be arranged (thus a finite number of states) and since for each state, there can be at most 4 different actions, there is a finite number of actions.
- **Mono-agent**: there is only a single agent trying to solve the eight puzzle problem.

2. Performance of Algorithms

Puzzle 1 (easy)	DFS	BFS	A* H1	A* H2
Depth of solution	29	1	1	1
Solution path cost	29	1	1	1
Time (seconds)	0.00282	0.000375	0.000240	0.000283
Number of visited nodes	29	2	1	1
Complete/optimal	Complete, not optimal	Complete, optimal	Complete, optimal	Complete, optimal

Puzzle 6 (medium)	DFS	BFS	A* H1	A* H2
Depth of solution	62452	14	14	14
Solution path cost	62452	14	14	14
Time (seconds)	5.4903	0.2648	0.0208	0.0211
Number of visited nodes	64792	3334	262	243
Complete/optimal	Complete, not optimal	Complete, optimal	Complete, optimal	Complete, optimal

Puzzle 3 (hard)	DFS	BFS	A* H1	A* H2
Depth of solution	110073	19	19	19
Solution path cost	110073	19	19	19
Time (seconds)	10.0378	2.8202	0.2040	0.1697
Number of visited nodes	122260	35960	2477	1895
Complete/optimal	Complete, not optimal	Complete, optimal	Complete, optimal	Complete, optimal

3. Heuristics

a. Description of the two heuristics used by A*

Heuristics 1 simply computes the number of misplaced numbers. Let's say we have:

2	3	0
1	4	5
6	7	8

"2", "3", "0", and "1" are misplaced; we get a cost of 4. However, we substract 1 to this in the end (unless we have a cost of 0, then we return 0); we'll discuss why in part b.

Heuristics 2 computes the sum of the Manhattan distance of all numbers to their goal position. The Manhattan distance itself is the sum of the distance horizontally and of the distance vertically between two points. Let's say we have:

2	3	8
1	4	5
6	7	0

- "2" is at distance 2 (2 horizontally)
- "3" is at distance 2 (1 horizontally - 1 vertically)
- "8" is at distance 2 (2 vertically)
- "1" is at distance 2 (1 horizontally - 1 vertically)
- "0" is at distance 4 (2 horizontally - 2 vertically)

The cost would therefore be 2 + 2 + 2 + 2 + 4 = 12. However, we divide this result by 2 in the end; we'll discuss why in part b.

b. Explanation of their admissibility

Both heuristics are admissible, as their estimated costs are always smaller or equal to the real cost.

The real cost in our case is a cost of 1 per move (3 swaps = cost of 3, etc.)

Heuristics 1: With a real cost of *x*, we know that we must do exactly *x* moves to get to the goal state. With 1 move, 2 numbers have moved from their original position (0 and a neighbour); with a second move, another number can be displaced (we don't include 0 as it is already displaced), and so on. So, with *x* moves (read "real cost of *x*"), we can displace **at least** *x+1* numbers. Knowing that, if we compute that *y* numbers are misplaced | y > 0, we know we can move these *y* numbers in **at least** *y-1* moves, so has a minimum real cost of *y-1*. Therefore, **Heuristics 1** is admissible as it is certain to never overcome the minimum real cost at one state.

Heuristics 2: The Manhattan distance of a number to its goal position is really the number of moves required to get to that position. However, the sum of all Manhattan distances will lead to some redundancy, because a single move can reduce the distance of 2 numbers at once. If we push this case to the extreme, it is possible that every move will reduce the distance of 2 nodes at once, therefore we know that the minimum number of moves (real cost) is half the sum of all Manhattan distances. Therefore, **Heuristics 2** is admissible because it divides the sum of all Manhattan distances by 2, which is certain to never overcome the minimum real cost at one state.

c. Does one heuristic dominate the other?

No, as none of the heuristics always have a higher score than the other. Here's a proof:

1 5 6
3 4 2
7 0 8

Cost of Heuristics 1: 5
Cost of Heuristics 2: 6

1 4 2
0 8 5
3 7 6

Cost of Heuristics 1: 5
Cost of Heuristics 2: 4

In the first case, **heuristics 2** dominates **heuristics 1**; in the second case, **heuristics 1** dominates **heuristics 2**. Therefore, none of them are dominant.

4. Conclusion

It is clear that A* is the best of the three, BFS is second; DFS is the worst, as it visits a huge number of nodes and almost always return a non-optimal solution. For example, for **puzzle #7**, it returns a solution of length 110 582, while the optimal solution is only of length 24.

The extra computational cost of having to calculate heuristics in A* is worthwhile; this drastically reduces our state space. Because our heuristics are not dominating each other, we compute both for a single state and return the maximum; this approach lead to faster results than **heuristics 1** or **heuristics 2** separately.

By using iterative search algorithms and relying on a set of "visited" states rather than the `isRepeated` method provided, we were able to avoid `maximum recursion depth` errors and — specifically related to `isRepeated` — speed up our algorithm (`O(1)` with a set of visited nodes rather than `O(d)` with `isRepeated`, where `d` is the maximum depth of the tree). We haven't had a case where we were not able to get *DFS* results, which was often the case before we made these changes.