P: Performance Measure

The agent's performance is measured by the quality of the solution it finds. The primary metrics are:

- **Goal Achievement:** Did the agent successfully reach the goal state?
- **Path Cost:** What was the total cost of the path from the initial state to the goal state? For the 8-puzzle, the cost is the number of moves. An optimal agent finds the solution with the minimum possible path cost. Each move has a uniform cost of 1.

E: Environment

The 8-puzzle exists in a well-defined environment with the following properties:

- **Fully Observable:** The agent has access to the complete state of the board at all times. It knows the location of every tile.
- **Deterministic:** The outcome of any action is guaranteed. Moving a tile into the blank space results in one specific, predictable new state.
- **Sequential:** The current action has consequences for all future actions. The solution is a sequence of moves, and each move affects the choices available for the next.
- **Static:** The environment does not change on its own. The puzzle's state only changes as a direct result of the agent's actions.
- **Discrete:** The problem has a finite number of states (distinct board configurations) and a finite set of actions (Up, Down, Left, Right).
- **Single-Agent:** The agent is the sole actor in the environment. There are no other agents to cooperate or compete with.

A: Actuators

The agent's actuators are the mechanisms through which it can alter the state of the environment.

- The agent can move the blank space **Up, Down, Left,** or **Right**. This is functionally equivalent to swapping an adjacent numbered tile into the blank space's position. The set of available actions is determined by the blank space's location (e.g., if the blank is in a corner, only two actions are possible).

S: Sensors

The agent's sensors are what it uses to perceive the state of the environment.

- The agent "senses" the entire board configuration at once. This is typically represented by a data structure, such as a 2D array or a list, which contains the current position of all 8 numbered tiles and the blank space.

State-Space Model

The 8-puzzle problem is formally defined by the following components:

- **States:** A state represents a unique configuration of the 8 tiles and the blank space on the 3x3 grid. The state space is the set of all possible configurations.
- **InitialState():** This function returns the starting configuration of the puzzle. For the experiments, a set of 30 unique, solvable initial states was used.
- **Actions(state):** This function returns the set of legal moves that can be taken from a given `state`. It determines the location of the blank tile and returns the possible directions (Up, Down, Left, Right) it can move. A corner position yields 2 possible actions, an edge position yields 3, and the center position yields 4.
- **Transition(state, action):** This function applies an `action` to a `state` and returns the resulting new state. For example, if the action is 'UP', the function swaps the blank tile with the tile directly above it.
- **GoalTest(state):** This function returns `true` if the given `state` matches the predefined goal configuration `[[1, 2, 3], [4, 5, 6], [7, 8, 0]]`, and `false` otherwise.
- **StepCost(state, action, next_state):** This function returns the cost of a single action. For the 8-puzzle, every move has a uniform cost of **1**.
- **Heuristic(state) (Pluggable):** This is a pluggable function that provides an estimated cost from the current `state` to the goal. Multiple variants were implemented to compare their performance:
    - h_0 **(Zero Heuristic):** Always returns 0, effectively turning A* into Uniform Cost Search.
    - h_1 **(Misplaced Tiles):** An admissible and consistent heuristic that simply counts the number of tiles that are not in their correct goal position.
    - h_1 **(Manhattan Distance):** A stronger, but still admissible and consistent, heuristic that sums the vertical and horizontal distances of each tile from its goal position.
    - h_2 **(Weighted Manhattan Distance):** An inadmissible heuristic that multiplies the Manhattan Distance by a constant (1.5) to act more greedily, prioritizing speed over finding the optimal path.

Heuristic Design & Proofs

This section provides the arguments for why the chosen h_1 heuristics are guaranteed to be admissible and consistent, a requirement for A* to find the optimal solution.

h_1 - Misplaced Tiles

- **Argument for Admissibility (Never Overestimates):**
    - **Definition:** An admissible heuristic never overestimates the true cost to reach the goal.

    - **Proof:** Every tile that is not in its correct final position must be moved at least once to solve the puzzle. The Misplaced Tiles heuristic calculates the number of tiles in this state. Therefore, it provides a count of the absolute minimum number of moves required. Since any real solution may involve moving a tile multiple times or moving other tiles out of the way, the actual cost will always be greater than or equal to this count. It is thus impossible for this heuristic to overestimate the true cost, making it **admissible**.
- **Argument for Consistency (Monotonic):**
    - **Definition:** A heuristic is consistent if, for any child node, its heuristic value is not more than 1 less than its parent's value (`h(parent) - h(child) ≤ 1`).
    - **Proof:** When a tile is moved, the Misplaced Tiles count can only change in one of three ways:
        1. A tile is moved *into* its correct goal position: The count decreases by exactly 1. (`h(parent) - h(child) = 1`)
        2. A tile is moved *out of* its correct goal position: The count increases by exactly 1. (`h(parent) - h(child) = -1`)
        3. A tile is moved from one incorrect position to another: The count remains unchanged. (`h(parent) - h(child) = 0`)
    - In all possible cases, the change satisfies the condition `h(parent) - h(child) ≤ 1`. The heuristic is therefore **consistent**.

h_1 - Manhattan Distance

- **Argument for Admissibility (Never Overestimates):**
    - **Proof:** The Manhattan distance for a single tile is the minimum number of moves required to move it to its goal position if no other tiles were on the board. Since other tiles can only ever obstruct a tile's path (requiring more moves to go around them), the true cost to get a tile home is always greater than or equal to its individual Manhattan distance.
    - The total Manhattan Distance heuristic is the sum of these individual minimums. Since each component of the sum is an underestimate of its true cost, the total sum must also be less than or equal to the total true cost of solving the puzzle. Therefore, the Manhattan Distance heuristic is **admissible**.
- **Argument for Consistency (Monotonic):**
    - **Proof:** When a single tile is moved one square:
        1. If the move is *towards* its goal position (horizontally or vertically), its individual Manhattan distance decreases by exactly 1, and the total heuristic value decreases by 1. (`h(parent) - h(child) = 1`)
        2. If the move is *away from* its goal position, its individual Manhattan distance increases by exactly 1, and the total heuristic value increases by 1. (`h(parent) - h(child) = -1`)

- No single move can change the total Manhattan distance by more than 1. The condition `h(parent) - h(child) ≤ 1` is always satisfied. The heuristic is therefore **consistent**.

Experiments & Results

The A* algorithm was run on 30 unique, solvable puzzle instances using each of the defined heuristics. The following tables report the key performance metrics for each run.

h_0 - Zero Heuristic (Uniform Cost Search Baseline)

Board	Difficulty	Cost/Depth	Nodes Expanded	Nodes Generated	Max Frontier
1	Easy	1	3	10	5
2	Easy	3	15	41	12
3	Easy	3	11	31	10
4	Easy	4	21	57	16
5	Easy	1	3	10	5
6	Easy	7	119	327	83
7	Easy	4	24	65	18
8	Easy	3	12	33	10

9	Medium	7	117	319	80
10	Medium	8	176	479	116
11	Medium	4	18	51	16
12	Medium	22	76,781	206,617	23,952
13	Medium	22	76,782	206,621	23,953
14	Medium	14	4,173	11,231	2,368
15	Medium	21	61,569	165,701	21,721
16	Medium	14	4,157	11,195	2,368
17	Medium	20	41,231	111,105	16,993
18	Medium	10	491	1,313	290
19	Medium	18	23,587	63,495	11,132
20	Medium	26	164,811	441,121	25,015

21	Hard	31	181,438	483,837	25,156
22	Hard	20	48,389	130,219	18,612
23	Hard	31	181,438	483,837	25,205
24	Hard	25	146,064	391,757	25,132
25	Hard	26	164,774	441,047	24,983
26	Hard	22	86,824	233,581	24,973
27	Hard	16	10,218	27,517	5,482
28	Hard	25	146,051	391,725	25,173
29	Hard	25	146,062	391,751	25,139
30	Hard	22	86,843	233,615	24,970
Total	--	**519**	**1,542,411**	**4,136,892**	--

Brandon Lewis

h_1 - Manhattan Distance Heuristic

Board	Difficulty	Cost/Depth	Nodes Expanded	Nodes Generated	Max Frontier
1	Easy	1	1	4	3
2	Easy	3	3	11	6
3	Easy	3	3	11	6
4	Easy	4	4	13	6
5	Easy	1	1	4	3
6	Easy	7	7	19	6
7	Easy	4	4	13	6
8	Easy	3	3	11	6
9	Medium	7	7	21	8
10	Medium	8	10	30	11
11	Medium	4	4	13	6

12	Medium	22	1,060	2,833	606
13	Medium	22	1,489	3,994	849
14	Medium	14	101	276	71
15	Medium	21	788	2,105	439
16	Medium	14	76	207	52
17	Medium	20	445	1,198	270
18	Medium	10	12	35	12
19	Medium	18	527	1,418	315
20	Medium	26	2,648	6,993	1,393
21	Hard	31	21,197	55,926	8,862
22	Hard	20	282	748	170
23	Hard	31	21,197	55,926	8,861

24	Hard	25	1,445	3,832	784
25	Hard	26	3,688	9,774	1,875
26	Hard	22	741	1,971	406
27	Hard	16	64	170	43
28	Hard	25	2,994	7,972	1,596
29	Hard	25	2,893	7,713	1,559
30	Hard	22	2,013	5,410	1,119
Total	--	**519**	**60,717**	**161,159**	--

Brandon Lewis

h_2 - Weighted Manhattan Distance Heuristic (w=1.5)

Board	Difficulty	Cost/Depth	Nodes Expanded	Nodes Generated	Max Frontier
1	Easy	1	1	4	3
2	Easy	3	3	11	6
3	Easy	3	3	11	6
4	Easy	4	4	13	6
5	Easy	1	1	4	3
6	Easy	7	7	19	6
7	Easy	4	4	13	6
8	Easy	3	3	11	6
9	Medium	7	7	21	8
10	Medium	8	8	23	8
11	Medium	4	4	13	6

12	Medium	24	500	1,328	303
13	Medium	22	526	1,434	345
14	Medium	14	116	321	88
15	Medium	21	524	1,411	325
16	Medium	14	36	100	26
17	Medium	20	189	519	132
18	Medium	10	10	28	9
19	Medium	18	292	793	184
20	Medium	28	1,065	2,838	636
21	Hard	31	1,017	2,724	614
22	Hard	20	85	231	61
23	Hard	31	1,014	2,717	613

24	Hard	25	85	232	60
25	Hard	26	894	2,409	543
26	Hard	24	183	491	119
27	Hard	16	114	307	80
28	Hard	25	672	1,807	423
29	Hard	25	406	1,095	250
30	Hard	22	1,058	2,854	629
Total	--	**527**	**8,831**	**23,800**	--

Results Analysis

The experimental data provides a clear picture of the trade-offs between different heuristic strategies.

1. **Optimality vs. Speed:**
    - Both the **Zero Heuristic (h_0)** and the **Manhattan Distance (h_1)** are admissible, and as predicted, they always found the optimal (shortest) solution path, resulting in an identical total path cost of **519**.
    - The **Weighted Manhattan (h_2)** is inadmissible. I noticed that it ended up producing a slightly higher total path length in its solutions, costing **527** in total. This is because it prioritizes paths that appear closer to the goal, even if they aren't truly the cheapest, finding suboptimal solutions for harder puzzles like Board #12, #20, and #26.
2. **Search Efficiency (Nodes Expanded):**

- The power of a good heuristic is immediately obvious here. The uninformed **h_0** search expanded over **1.5 million** nodes.
- By providing a simple estimate, the **h_1** heuristic dramatically cut down the search space, expanding only **~60,000** nodes—a **96% reduction** in search effort.
- While the **Weighted Manhattan (**h_2**)** heuristic was by far the most efficient, expanding fewer than **9,000** nodes, the trade-off with its inadmissibility is that you may end up not finding the shortest path to the goal state.

3. **Memory Usage (Max Frontier Size):**
   - The maximum size of the frontier directly correlates with the number of nodes expanded. The uninformed h_0 search required storing tens of thousands of nodes in memory for the hardest problems. h_1 and h_2 required significantly less memory, showing that a better heuristic not only saves time but also reduces the memory footprint of the search.

Conclusion:

Based on the results, I noticed that the h_2 weighted manhattan heuristic ended up producing a slightly higher total path length in its solutions. While this heuristic was by far the most efficient in terms of search effort—expanding fewer than 9,000 nodes compared to over 1.5 million for the baseline—the trade-off with its inadmissibility is that you may end up not finding the shortest path.

For this reason, the **Manhattan Distance (**h_1**) is the superior heuristic for general use**. It provides a massive improvement in performance over an uninformed search while still guaranteeing that the solution found is the best one possible. The Weighted Manhattan (h_2) is a viable alternative only when finding *any* solution as quickly as possible is the primary goal, and optimality can be sacrificed.