**What is A* Search?**

A* is an intelligent search algorithm used to find the shortest path between a start state and a goal state.

It selects the best state based on the evaluation function:

f(n) = g(n) + h(n)

Where:
g(n): the cost from the start state to the current state (number of moves).
h(n): a heuristic function that estimates the remaining cost to reach the goal.
f(n): the total estimated cost used to choose the next state.

---

**Heuristic Function (h)**

The heuristic function is the most important part of A*.
It helps the algorithm reach the goal faster than uninformed search algorithms like BFS.

Common heuristics used in the 8-Puzzle problem:

**1. Misplaced Tiles**

This heuristic counts the number of tiles that are not in their correct position, excluding the empty tile.

Example:

1 2 3
4 5 6
_ 7 8

Tiles 7 and 8 are not in the correct position, so:
h = 2

---

**2. Manhattan Distance**

This heuristic calculates the sum of the distances of each tile from its goal position.

Formula:
|x_current − x_goal| + |y_current − y_goal|

This heuristic is more accurate than misplaced tiles, so A* performs better when using it.

---

**Steps of A* Algorithm in 8-Puzzle**

1. Start with the initial state
   Put it into the OPEN list
   Set g = 0
   Calculate h
   Calculate f = g + h
2. Choose the best state
   Select the state with the lowest f value
   Move it from OPEN list to CLOSED list
3. Generate new states
   Move the empty tile up, down, left, or right
   Each move produces a new state
4. Calculate values for each new state
   g = parent g + 1
   h = heuristic value
   f = g + h
5. Ignore repeated states
   If the state is already in CLOSED list, ignore it
   If it is in OPEN list with higher cost, update it
6. Repeat
   Continue until the goal state is reached
   Or no more states exist (failure)

---

**Problem Definition**

Initial State:

1 2 3
4 _ 6
7 5 8

Goal State:

1 2 3
4 5 6
7 8 _

The empty tile is represented by (_).

---

**A* Evaluation Function**

A* uses the formula:

f(n) = g(n) + h(n)

Where:
g(n) is the number of moves from start
h(n) is the estimated distance to the goal

The heuristic used here is Manhattan Distance.

---

**Step-by-Step Execution**

**Step 0: Start State**

1 2 3
4 _ 6
7 5 8

g = 0

Manhattan Distance:
Tile 5 is one move away
Tile 8 is one move away

h = 2
f = 2

This state is added to the OPEN list.

---

**Step 1: Expand the Best Node**

Possible moves of the empty tile are up, down, left, and right.

Move Up:

1 _ 3
4 2 6
7 5 8

g = 1
h = 3
f = 4

Move Down:

1 2 3
4 5 6
7 _ 8

g = 1
h = 1
f = 2

Move Left:

1 2 3
_ 4 6
7 5 8

g = 1
h = 3
f = 4

Move Right:

1 2 3
4 6 _
7 5 8

g = 1
h = 3
f = 4

---

**Step 2: Choose Best Node**

The state with the lowest f value is:

1 2 3
4 5 6
7 _ 8

f = 2

This state is moved to the CLOSED list.

---

**Step 3: Expand This State**

Possible moves are up, left, and right.

Move Right leads to the goal state:

1 2 3
4 5 6
7 8 _

g = 2
h = 0
f = 2

Goal state is reached.

---

**Solution Path**

Step 0:
1 2 3 / 4 _ 6 / 7 5 8

Step 1:
1 2 3 / 4 5 6 / 7 _ 8

Step 2:
1 2 3 / 4 5 6 / 7 8 _

The puzzle is solved in 2 moves.

---

**Why A* is Effective**

Uses heuristic information
Avoids unnecessary paths

Guarantees the optimal solution
Faster than BFS for this problem

---

**Summary**

Insert start state into OPEN list
Choose node with lowest f(n)
Expand the node and generate children
Calculate g, h, and f for each child
Ignore repeated states
Stop when the goal is reached