8-Puzzle Solver using Hill Climbing Algorithm

1. Objective

To implement a Python program that solves the 8-Puzzle problem using the **Hill Climbing algorithm** with the **Manhattan distance heuristic**, demonstrating a simple greedy local search strategy.

2. Introduction

The **8-Puzzle** is a sliding puzzle consisting of eight numbered tiles and a single blank space on a 3x3 grid. The objective is to rearrange the tiles from a given initial configuration into the goal state, typically ordered from 1 to 8 with the blank at the bottom right corner.
This problem is widely used in artificial intelligence and heuristic search studies.
In this project, we explore the **Hill Climbing algorithm**, a basic local search method that continually moves toward the state that appears to be closest to the goal.

3. Algorithm Used

Hill Climbing is a **greedy local search algorithm** that:

1. Starts with an initial solution (in this case, the initial puzzle configuration).
2. Generates all neighboring states by moving the blank tile up, down, left, or right.
3. Evaluates these states using a **heuristic function** (Manhattan distance).
4. Moves to the neighbor with the **lowest heuristic cost**.

The process repeats until:

1. The goal is reached (heuristic is 0), or
2. No better neighbor exists (algorithm gets stuck in a **local minimum**).

4. Working Principle

Initialization:
Start with the given puzzle configuration.

Neighbor Generation:
Generate all possible states by moving the blank tile in four possible directions.

Heuristic Calculation:
For each neighbor, compute the **Manhattan distance**, which is the sum of the distances of tiles from their goal positions.

Move Decision:
Select the neighbor with the lowest heuristic value.

1. If it's better than the current state, move to this neighbor.
2. Otherwise, stop, as the algorithm has reached a local minimum.

Termination:
Continue the above steps until the goal is reached or no improvement is possible.

5. Python Code Features

1. Computes the **Manhattan distance** for each puzzle state.
2. Efficiently generates all valid neighboring configurations by sliding the blank tile.
3. Continuously moves to the best neighbor based on heuristic.
4. Prints the board state and heuristic at each step.
5. Stops if no better neighbor exists (demonstrating the local minimum problem).

6. Advantages and Disadvantages

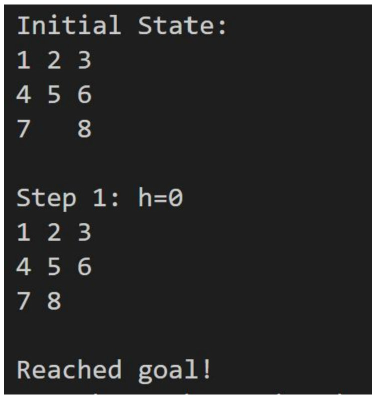
Advantages:

1. Very simple and easy to implement.
2. Uses very little memory (tracks only current and neighbor states).

Disadvantages:

1. Can get stuck in **local minima**, unable to reach the actual goal.
2. Does **not guarantee** finding a solution even if the puzzle is solvable.
3. Cannot backtrack or explore alternative paths.

7. Output

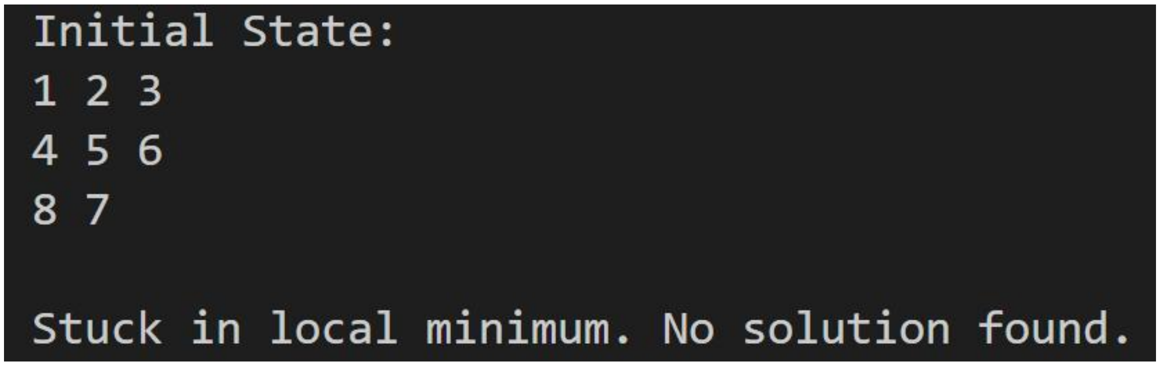


```
Initial State:
1 2 3
4 5 6
7   8

Step 1: h=0
1 2 3
4 5 6
7 8

Reached goal!
```

Where it Stuck



```
Initial State:
1 2 3
4 5 6
8 7

Stuck in local minimum. No solution found.
```

8. Conclusion

This project successfully demonstrates how the **Hill Climbing algorithm** can be applied to solve the 8-Puzzle problem using a heuristic approach. While it effectively illustrates greedy local search principles, it also underscores the importance of choosing search strategies wisely, especially given the risk of local minima in problems like the 8-Puzzle.