Step 1: Understand the Problem

The 8-Puzzle consists of a 3×3 grid with 8 numbered tiles and one empty space (0).

- Initial State: The starting arrangement of tiles
- Goal State: The desired final arrangement

The empty space (0) can move:

- Up
- Down
- Left
- Right

by swapping with the adjacent tile.

Initial State (from your example):

(

2 8 3

1 0 4

7 6 5

)

Goal State:

(

1 2 3

8 0 4

7 6 5

)

Step 2: Understand BFS

Breadth-First Search (BFS) explores states level by level:

1. Start from the initial state.
2. Generate all possible states at depth 1.
3. Then all states at depth 2, and so on.
4. Stop as soon as the goal state is found.

Advantages of BFS:

- Guarantees the shortest path
- Simple to implement
- Complete (will find a solution if it exists)

Step 3: Implementing BFS for 8-Puzzle

1. Start from the Initial State.
2. Insert it into a queue.

3. Generate all possible moves of the empty tile (0).

4. Add each new state to the queue if it was not visited before.

5. Repeat until the Goal State is reached.

Step 4: Avoiding Repetition

• Use a visited set to store already explored states.

• This prevents infinite loops and repeated work.

Step 5: Track the Solution Path

• Each state keeps track of the path that led to it.

• When the goal is reached, the stored path represents the solution.

Step-by-Step Moves:

Step 1: Move UP

Action: Swap the blank space with number 6.

Result: Blank moves to the Center position.

Step 2: Move UP

Action: Swap the blank space with number 8.

Result: Blank moves to the Top-Middle position.

Step 3: Move LEFT

Action: Swap the blank space with number 2.

Result: Blank moves to the Top-Left position.

Step 4: Move DOWN

Action: Swap the blank space with number 1.

Result: Blank moves to the Middle-Left position.

Step 5: Move RIGHT

Action: Swap the blank space with number 8.

Result: Blank moves to the Center position (Goal State reached).

Step 6: Summary

1. Use a queue to apply BFS.
2. Explore states level by level.
3. Generate valid moves of the empty tile.
4. Avoid repeated states using a visited set.
5. Stop when the goal state is found.
6. BFS guarantees the shortest solution.