**Assignment 1: Implement DFS, BFS for 8-Puzzle Problem**

**Problem Statement:**  
Implement Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms to solve the 8-puzzle problem, where the objective is to arrange tiles in a 3x3 grid in numerical order by sliding tiles into an empty space.

**Objectives:**

* To solve the 8-puzzle problem using DFS and BFS algorithms.
* To implement efficient state representation and valid move detection.
* To compare DFS and BFS performance in terms of time, space, and optimality.
* To handle both solvable and unsolvable puzzle configurations effectively.

**Theory:**

* **Methodology:**  
  Depth-First Search (DFS): The DFS algorithm explores the 8-puzzle by going deep into one possible path before backtracking when a dead end is reached. It uses a stack (LIFO) to track states and explores child nodes until the goal state is found or all possible states are visited. DFS is memory efficient but may not find the shortest solution path.

Breadth-First Search (BFS): The BFS algorithm explores all possible moves from the initial state level by level, using a queue (FIFO) to track states. It checks each neighboring state before moving deeper, ensuring that the shortest solution path is found. BFS consumes more memory but guarantees the shortest path to the goal state.

* **Working Principle / Algorithm:**
  + **DFS Algorithm:**
    1. Start with the root (initial configuration).
    2. Explore each branch as deep as possible.
    3. Use backtracking when no further moves are possible.
    4. Repeat until the goal configuration is reached or all configurations are explored.
  + **BFS Algorithm:**
    1. Start with the root (initial configuration).
    2. Explore all nodes at the present depth.
    3. Move to the next depth level and repeat until the goal configuration is reached.
* **Advantages:**

DFS**:**

* Memory efficient, stores only the current path.
* Can find deep solutions without exploring the entire space.
* Simple to implement for large branching problems.
* Useful when any solution is acceptable, not necessarily the shortest.

BFS:

* Guarantees the shortest solution path.
* Systematically explores all possible states.
* Ideal for problems where the shortest solution is required.
* Easily parallelizable for distributed computing.
* **Disadvantages / Limitations:**

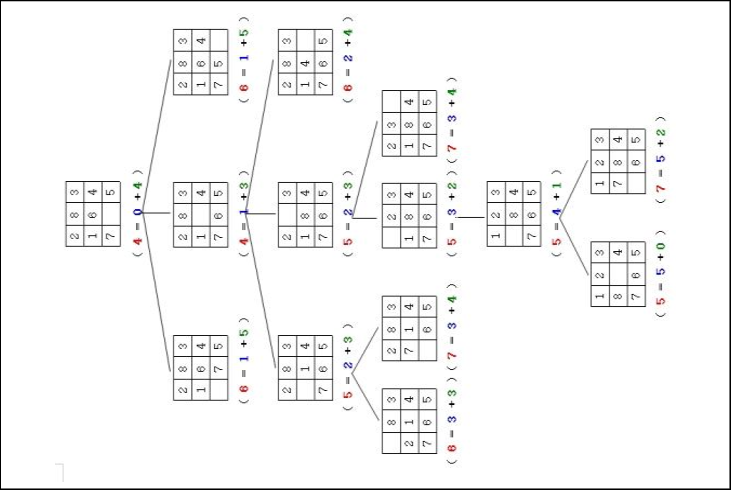
DFS:

* May get stuck in deep, irrelevant paths, missing shorter solutions.
* Does not guarantee the shortest path to the solution.
* Can lead to infinite loops if cycles in the search space are not handled.

BFS:

* Consumes a lot of memory, as it stores all explored nodes.
* Slower for deep solutions due to exploring all levels equally.
* Inefficient for problems with large state spaces or high branching factors.

**Diagram:**



**Conclusion:**  
DFS and BFS are effective search algorithms for the 8-puzzle problem. However, BFS ensures the shortest path, while DFS may consume less memory for deep searches.