**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.

**Objectives:**

* Understand the functioning of DFS and BFS.
* Implement and compare these algorithms for solving the 8-puzzle problem.

**Theory** **:**

The 8-puzzle problem can be represented as a graph search problem where each state of the puzzle is a node and each valid move (up, down, left, or right of the empty space) represents an edge between nodes. To solve the problem, we will use DFS and BFS, which are both classic graph traversal algorithms but differ in their search strategies.

DFS explores as far as possible along each branch before backtracking, focusing on depth, while BFS explores all neighbors at the present depth level before moving to the next depth, focusing on breadth. Both methods explore the possible moves of tiles from the initial configuration to the goal state, but their working principles differ.

**Methodology:**

* DFS (Depth First Search): DFS explores as deep as possible into the state space of the puzzle, moving down a path until it can go no further before backtracking to explore alternative paths.
* BFS (Breadth First Search): BFS systematically explores all possible moves level by level, ensuring that all nodes at one depth are visited before moving on to the next level. BFS guarantees that if a solution exists, it will find the shortest one in terms of the number of moves.

**Working Principle / Algorithm:**

**DFS Algorithm:**

1. Start with the root node, which represents the initial configuration of the puzzle.
2. Mark the node as visited and explore its neighbors (possible moves).
3. Move to the next unvisited neighbor and repeat the process, going as deep as possible.
4. If no more moves are possible from a node, backtrack to the previous node and explore other unvisited neighbors.
5. Repeat this process until the goal state is reached or all configurations have been explored.

**BFS Algorithm:**

1. Start with the root node representing the initial configuration of the puzzle.
2. Explore all possible moves (neighbors) from the root node and add them to a queue.
3. Remove a node from the front of the queue and explore its neighbors.
4. Add all unvisited neighbors to the queue and continue this process until the goal configuration is found or all nodes are explored.

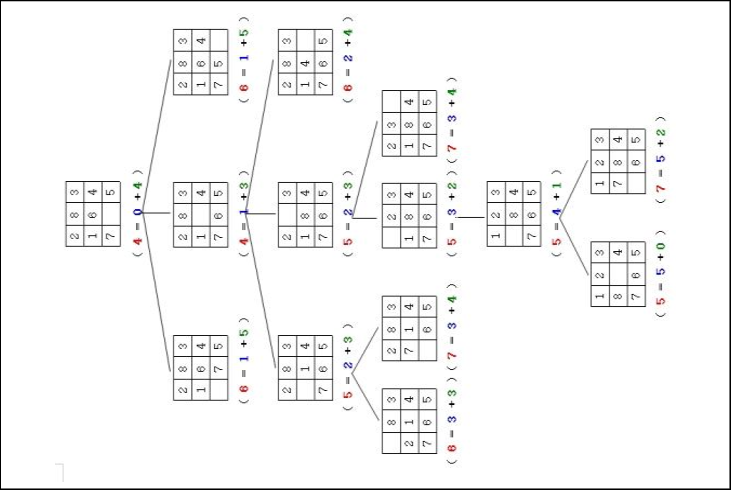
**Advantages:**

* DFS:
  + Memory efficient, especially for deep search spaces, as it doesn't store all nodes in memory.
  + Useful when the solution is deep in the tree.
* BFS:
  + Guarantees the shortest path (in terms of number of moves) to the goal state, provided the problem graph is unweighted.
  + Good for finding solutions when the goal is close to the root node.

**Disadvantages / Limitations:**

* DFS:
  + May not find the shortest path to the solution since it goes deep into the search space without considering other, potentially shorter, paths.
  + Can get stuck in infinite loops if cycles exist in the search space.
* BFS:
  + Memory intensive, especially for large search spaces, because it needs to store all nodes at the current depth before moving on to the next level.

**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.