**** **Bansilal Ramnath Agarwal Charitable Trust’s**

**Vishwakarma Institute of Information Technology, Pune-48**

**(An Autonomous Institute affiliated to Savitribai Phule Pune University)**

**Department of Computer Science and Engineering (Artificial Intelligence)**

**LAB SUBMISSION**

**Artificial Intelligence**

**CAUA31201**

*Submitted by:*

*Avdhoot Nilesh Pimparkar,*

*PRN: 22210178,*

*Roll Number: 381054*

*Third Year*

*Semester I Academic Year 2024-25*

**Assignment: 1**

Aim:

1. To understand how search algorithms like DFS and BFS can be applied to solve constraint-based problems such as the 8-puzzle.
2. To learn how to formulate a search problem, including defining the initial state, goal state, and successor functions.
3. To understand the differences in how DFS and BFS traverse the state space in search of a solution.
4. To learn how to analyze the time and space complexity of DFS and BFS for solving the 8-puzzle.
5. To understand the practical implementation of search algorithms and compare their performance in terms of efficiency and completeness.

Theory:

1. **8-Puzzle Problem Overview:**

The 8-puzzle problem consists of a 3x3 grid with tiles numbered 1 to 8 and one empty space (represented by 0). The goal is to rearrange the tiles by sliding them into the empty space until they match a specified goal configuration, usually ordered from 1 to 8 with the blank space in the bottom-right corner.

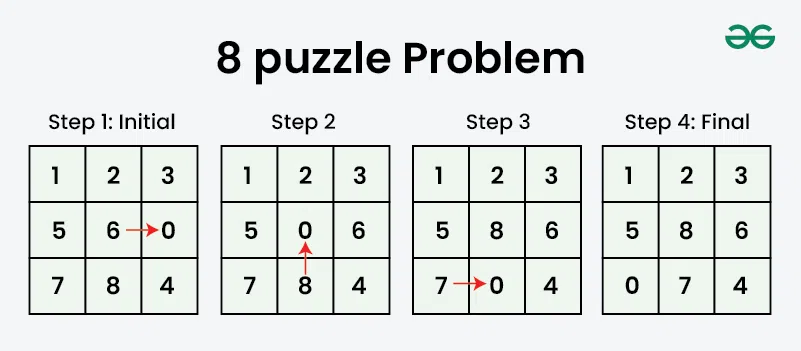


Figure 1. 8-Puzzle Problem

1. **State Representation:**

Each state represents a configuration of tiles in the grid. The puzzle’s state can be represented as a list of numbers where 0 indicates the blank space. The challenge is to move the tiles into the goal state using a sequence of valid moves.

1. **Search Problem:** The 8-puzzle problem is a search problem where:

* **Initial State**: The current configuration of tiles.
* **Goal State**: The desired arrangement.
* **Successor Function**: Generates new states by moving adjacent tiles into the blank space.

1. **Search Algorithms:**
   * **Breadth-First Search (BFS)**: Explores all possible moves level by level, ensuring the shortest path is found. It’s complete and optimal but uses significant memory.
   * **Depth-First Search (DFS)**: Explores as deeply as possible along each path, backtracking when necessary. DFS is memory-efficient but may not find the shortest solution.

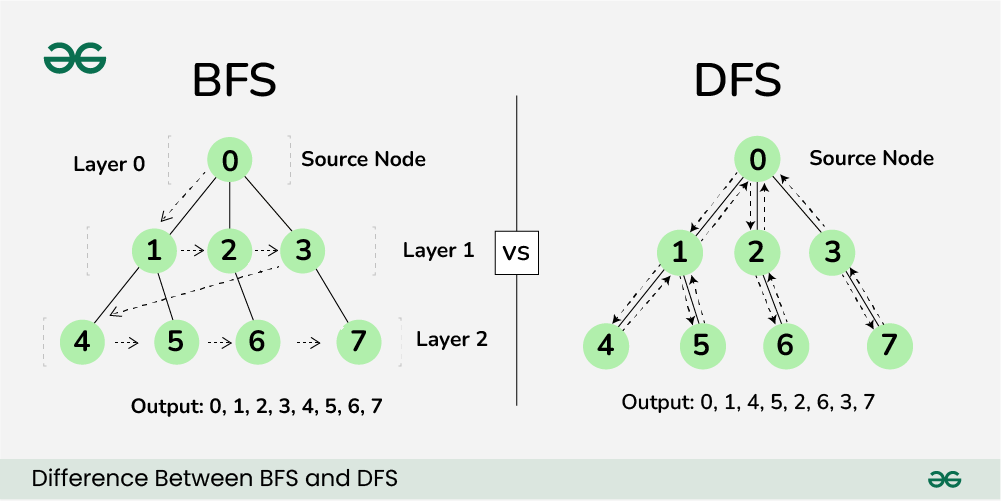


Figure 2. BFS and DFS

1. **Comparison:**

* **BFS** is suitable for finding the shortest solution but consumes more memory.
* **DFS** is faster in terms of memory but may not always find the optimal solution and is incomplete if it gets stuck in loops or deep paths.

Both algorithms are foundational for understanding problem-solving strategies in AI.

Conclusion:

By solving the 8-puzzle problem using DFS and BFS, we get to see how these two algorithms work differently. BFS finds the shortest solution but can take up a lot of memory, while DFS uses less memory but might not find the best solution. This exercise helps us understand how these search methods work, and teaches us about the trade-offs between speed, memory use, and finding the best answer in problem-solving.