CSE-AI TY A div

Student Name: Prachee Prasad

Roll No.: 381060

Assignment 1

Implement DFS and BFS for 8-Puzzle Problem

Problem Statement:

To implement and compare the performance of **Depth First Search (DFS)** and **Breadth First Search (BFS)** algorithms in solving the **8-puzzle problem**, where the goal is to reach a predefined state by moving tiles using search strategies.

Objective:

To understand and apply the concepts of uninformed search strategies (DFS and BFS) to solve the 8-puzzle problem and analyze their efficiency in finding solutions.

Requirements:

- Programming Language: **C++ / Python**
- Concepts: Graph Search, State Representation, Search Tree
- Input: Initial state of the 8-puzzle
- Output: Sequence of moves leading to the goal state

Operating System:

Windows / Linux / macOS

Libraries and Packages Used:

- **C++ STL** (Standard Template Library)

    - `queue` for BFS
    - `stack` for DFS
    - `map` or `set` for visited states tracking

Theory:

Definition:

The **8-puzzle problem** consists of a 3×3 grid with eight numbered tiles and one empty space. The objective is to move the tiles until the goal configuration is achieved.

Structure:

- **State Space:** All possible tile arrangements.
- **Operators:** Move blank tile (Up, Down, Left, Right).
- **Goal Test:** Check if current configuration matches goal state.
- **Path Cost:** Number of moves taken to reach the goal.

Methodology:

1. **Represent** the 8-puzzle as a node with current state and blank position.
2. **DFS Algorithm:**
    - Use a stack to explore nodes.
    - Go deep along one branch before backtracking.
    - May find solution faster but not always optimal.
3. **BFS Algorithm:**
    - Use a queue to explore nodes level by level.
    - Guarantees the shortest path to goal.

4. **Compare** both algorithms in terms of number of nodes expanded and time taken.

Advantages:

- **DFS:**
    - Requires less memory.
    - Can reach a solution quickly for deep goals.
- **BFS:**
    - Always finds the shortest path.
    - Systematic and complete.

Limitations:

- **DFS:**
    - May go into infinite loops without proper checks.
    - Not optimal.
- **BFS:**
    - High memory usage.
    - Slow for large state spaces.

Working / Algorithm:

Algorithm for BFS:

1. Initialize a queue with the start state.
2. While the queue is not empty:
   a. Dequeue a node.
   b. If goal is reached → return path.
   c. Generate all valid next states and enqueue them.

    d. Mark visited states.

Algorithm for DFS:

1. Initialize a stack with the start state.

2. While the stack is not empty:
    a. Pop a node.
    b. If goal is reached → return path.
    c. Generate successors and push them onto the stack.
    d. Mark visited states.

Conclusion:

DFS and BFS are fundamental search algorithms for solving state-space problems like the 8-puzzle.

- **BFS** ensures finding the shortest solution but uses more memory.

- **DFS** explores deeper states quickly but can be inefficient for large spaces.
  The choice depends on whether the goal is **efficiency or optimality**.