**Name: Aditya Parade  
Roll No: 381047  
PRN: 22311577**

**Assignment 1**

**Problem Statement:** Implementation of DFS and BFS for 8-Puzzle Problem

**1. Introduction**

The **8-puzzle problem** is a classic example of **Artificial Intelligence search problems**, often used to illustrate how search algorithms work.  
It consists of a 3×3 grid with 8 numbered tiles and one empty space. The goal is to rearrange the tiles from a given **initial state** to the **goal state** by sliding tiles into the empty position.

For example:

**Initial State:**  
1 2 3  
4 \_ 6  
7 5 8

**Goal State:**  
1 2 3  
4 5 6  
7 8 \_

To solve this problem, search algorithms such as **Depth-First Search (DFS)** and **Breadth-First Search (BFS)** are commonly used. These algorithms explore possible states by moving the blank space (up, down, left, right) until the goal configuration is achieved.

**2. Objective**

The main objectives of this experiment are:

* To implement and understand the working of **DFS** and **BFS** algorithms.
* To apply these algorithms to solve the **8-puzzle problem**.
* To compare the efficiency of DFS and BFS in terms of search depth, time, and memory usage.
* To understand how uninformed search strategies explore state spaces differently.

**3. Theory**

**3.1 The 8-Puzzle Problem**

The 8-puzzle is represented as a **state-space problem**, where:

* Each **state** represents a particular configuration of tiles.
* Each **action** corresponds to moving the blank tile in one of the four possible directions (up, down, left, right).
* The **goal state** is achieved when all tiles are arranged in the correct order.

Each move generates a **new state** by swapping the blank with an adjacent tile. The total possible configurations are 9! (362,880), but only half are solvable.

**3.2 Depth-First Search (DFS)**

DFS is a **blind search algorithm** that explores as far as possible along each branch before backtracking.  
It uses a **stack** (LIFO) data structure, either explicitly or through recursion.

**Characteristics:**

* May go deep into one branch before exploring others.
* Memory efficient but can get stuck in infinite loops if not managed properly.
* Not guaranteed to find the shortest solution.

**Steps:**

1. Start from the initial state.
2. Push it onto a stack.
3. Pop a state, generate its successors.
4. Push unvisited states onto the stack.
5. Repeat until the goal is found or stack becomes empty.

**3.3 Breadth-First Search (BFS)**

BFS is an **uninformed search algorithm** that explores all nodes at the current depth before moving to the next level.  
It uses a **queue** (FIFO) data structure.

**Characteristics:**

* Explores level by level.
* Guaranteed to find the shortest path (if one exists).
* Requires more memory compared to DFS.

**Steps:**

1. Start from the initial state.
2. Insert it into a queue.
3. Remove the front state and generate successors.
4. Add unvisited successors to the queue.
5. Continue until the goal is found or queue is empty.

**3.4 Comparison Table**

| **Feature** | **DFS** | **BFS** |
| --- | --- | --- |
| Data Structure | Stack | Queue |
| Completeness | May fail (if infinite) | Always complete |
| Optimality | Not guaranteed | Guaranteed |
| Time Complexity | O(b^m) | O(b^d) |
| Space Complexity | O(bm) | O(b^d) |
| Suitable For | Deep but narrow searches | Shallow but wide searches |

*(b = branching factor, d = depth of goal node, m = maximum depth)*

**4. Conclusion**

The implementation of **DFS and BFS** for the 8-puzzle problem demonstrates how uninformed search algorithms explore the problem space differently.

* **BFS** ensures that the solution found is **optimal**, but it consumes more memory.
* **DFS**, on the other hand, is **memory efficient** and may find a solution faster for shallow goals, but it might also get stuck in deep or infinite paths.

Through this experiment, we learned how to represent the 8-puzzle as a **state-space problem** and apply **systematic search strategies** to reach the goal configuration.  
Understanding these basic algorithms provides a strong foundation for more advanced search techniques like **A\*** and **heuristic-based algorithms** used in modern AI systems.