**ASSIGNMENT HELP**

**MANUAL**



SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR THE SKILL AND COMPETENCY EVALUATION OF

ARTIFICIAL INTELLIGENCE [CAUA31201]

IN

**CSE AI DEPARTMENT**

BY

**Vedant Rakesh Mukhekar**

**Class: T.Y. BTech Division: A Batch: A2**

**Batch Teacher**

**Dr. ANURADHA YENKIKAR.**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **CONTENTS** | **PAGE NO.** |
| **1** | **PROBLEM STATEMENT** | **4-5** |
| **2** | **LIBRARY USED** | **5-6** |
| **3** | **THEORY** | **6-7** |
| **4** | **METHODOLOGY** | **7-14** |
| **5** | **ADVANTAGES & DISADVANTAGES** | **14-15** |
| **6** | **WORKING** | **16-17** |
| **7** | **DIAGRAM** | **17-18** |
| **8** | **CONCLUSION** | **18-19** |

### ****Problem Statement****

The 8-puzzle is a classic search problem where you are given a 3x3 board containing eight numbered tiles (1 to 8) and one blank tile. The goal is to rearrange the tiles to match a desired goal configuration by sliding the tiles horizontally or vertically into the blank space. Implement two search algorithms, DFS and BFS, to solve the puzzle, and analyze the performance of both.

### ****Library Used****

* **Python Libraries**:
  + queue: For implementing BFS (Queue structure).
  + copy: To handle deep copying of puzzle states.
  + time: To measure the performance (execution time).
  + **Optional**: Libraries such as numpy or matplotlib can be used for better representation and performance analysis.

### ****Theory****

* **8-Puzzle Problem**: A sliding puzzle problem where the board consists of 8 numbered tiles and an empty space. The objective is to move the tiles using the empty space to match a goal configuration.
* **Breadth-First Search (BFS)**: An uninformed search algorithm that explores all possible configurations level by level. It guarantees finding the shortest path to the solution but consumes a large amount of memory.
* **Depth-First Search (DFS)**: An uninformed search algorithm that explores one branch of the puzzle deeply before backtracking. It can be more memory efficient than BFS but does not guarantee finding the shortest path.

### ****Methodology****

1. **State Representation**: Represent the puzzle as a list of lists or a 2D array, where each element corresponds to a tile position.
2. **Goal State**: The target configuration is:

Copy code

1 2 3

4 5 6

7 8 \_

1. **Allowed Moves**: The empty tile can be swapped with neighboring tiles (left, right, up, down), provided the move remains within bounds.
2. **DFS and BFS Implementation**:
   * **BFS**:
     + Initialize a queue with the starting configuration.
     + Explore all neighboring states of the current state.
     + Add new states to the queue until the goal state is found.
   * **DFS**:
     + Initialize a stack with the starting configuration.
     + Explore deeply by picking one state at a time from the stack.
     + Backtrack when a branch leads to no solution and continue with another branch.
3. **Termination**: The algorithm terminates when the goal state is reached, or if no solution exists for a given configuration.

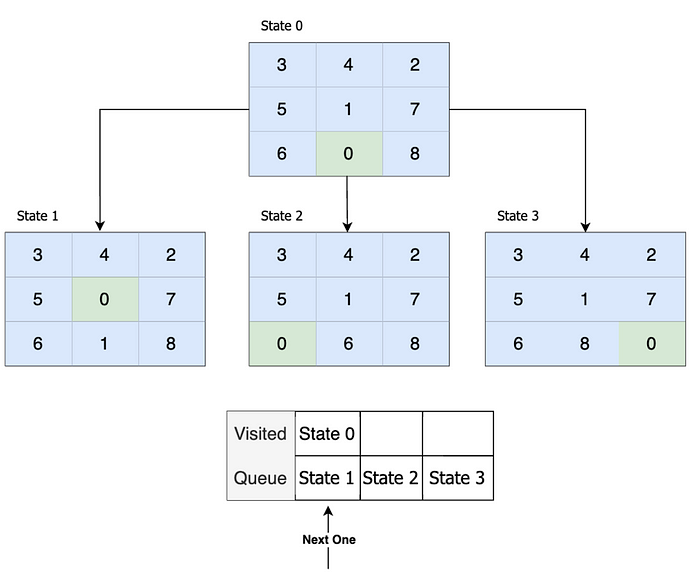
### ****Advantages & Disadvantages****

* **BFS**:
  + **Advantages**:
    - Guarantees finding the optimal solution (shortest path).
  + **Disadvantages**:
    - High memory usage.
    - Slow performance for large search spaces.
* **DFS**:
  + **Advantages**:
    - Low memory usage.
    - Can be faster in finding a solution in some cases.
  + **Disadvantages**:
    - Might explore deeper unnecessary branches.
    - Does not guarantee finding the shortest solution.

### ****Working****

* Start with an initial puzzle configuration.
* Choose DFS or BFS to solve the puzzle.
* Expand states by sliding the blank tile.
* Keep track of visited states to avoid cycles.
* The search algorithm continues until it reaches the goal state.

### ****Diagram:****



### ****Conclusion****

Both DFS and BFS are useful for solving the 8-puzzle problem, but they have trade-offs in terms of time and space complexity. BFS guarantees finding the optimal solution but at the cost of high memory consumption, while DFS is more memory-efficient but may not always yield the shortest solution. The choice between the two depends on the specific constraints of the problem.