ARTIFICIAL INTELLIGENCE LAB

ASSIGNMENT-1: DFS, BFS

(Read all the instructions carefully & adhere to them.)

Date: Feb 14, 2026

Total Credit: 20 (Implementation:10; Documentation & Explanation: 10)

Instructions:

1. Markings will be based on the correctness and soundness of the outputs.
2. Marks will be deducted in case of plagiarism.
3. Proper indentation and appropriate comments are mandatory.
4. You should zip all the required files and name the zip file as:
   roll_no .ipynb, eg. 2501cs11.ipynb.
5. Upload your assignment by downloading the .ipynb file from google colab on moodle, also all the explanation you can write in colab notebook only.

For any queries regarding this assignment, you can contact:
Deepak Kumar (deepakkumar1538@gmail.com)
Kshetrimayum Boynao Singh (boynfrancis@gmail.com)

Assignment: Solving the 8-Puzzle Problem using BFS and DFS

Problem Statement

You are given a 3×3 grid (the classic 8-puzzle) containing the numbers 1 to 8 and one blank space (‘B’).

The goal is to reach the target grid configuration by sliding the blank space up, down, left, or right.

Target grid (fixed):
1 2 3
4 5 6
7 8 B

A sample initial grid (randomly generated) could be:

3 2 1
4 5 6
8 7 B

Tasks

Task 1: Random Grid Generation

Write a function to generate a random initial 3×3 grid that contains all numbers 1–8 and a blank space B.

Ensure that the generated state is valid (no duplicates, no missing tiles).

Task 2: BFS Implementation

Implement Breadth First Search (BFS) to solve the puzzle:

Treat each grid state as a node.

The blank space (‘B’) can move up, down, left, or right (if valid).

Keep track of visited states to avoid infinite loops.

Output:

Number of steps (depth) taken to reach the solution.

The sequence of moves (Up, Down, Left, Right).

Task 3: DFS Implementation

Implement Depth First Search (DFS) to attempt solving the puzzle.

Output:

Whether a solution was found.

Number of steps explored before finding a solution (or reaching a limit).

The sequence of moves if successful.

To prevent infinite recursion, add a depth limit (e.g., 30).

Task 4: Compare BFS vs DFS

Run BFS and DFS on the same random initial grid.

Compare:

Number of steps to reach the solution.

Nodes explored.

Time taken (use Python’s time module).

Write a short explanation:

When BFS is faster/better.

When DFS is faster/better (e.g., shallow vs deep solutions).