Lab 10 and 11: BFS and DFS

1. **Connected Components**
   Using DFS and BFS determine the number of connected components in an undirected graph.

2. **Bipartite Graph Check**
   Using BFS, determine whether a given graph is a bipartite graph or not.

3. **Cycle Detection (Undirected Graph)**
   Use DFS to detect whether a given undirected graph contains a cycle.

4. **Topological Sorting**
   Implement Topological Sort for a Directed Acyclic Graph (DAG) using DFS.
   Input:

   ```
   6 6
   5 2
   5 0
   4 0
   4 1
   2 3
   3 1
   ```

   Output: One valid topological order: 5 4 2 3 1 0

5. **Strongly Connected** Given a directed graph G = (V, E) in its adjacency-list representation. Check whether this graph is strongly connected or not? Desired Time complexity is $O(m + n)$.

6. **Strongly Connected Components** Given a directed graph. Find all the strongly connected components of this graph using DFS in $O(m + n)$.

7. Given a connected undirected graph. Find the diameter of the given graph in $O(m + n)n$.

   The diameter of a graph is a measure of its "width" or "largest distance" and is defined as the largest shortest path distance between any two nodes in the graph.

   Formal definition: Let $G = (V, E)$ be a graph with vertex set V and edge set E. The *diameter* of G, denoted by D, is defined as:

   $$D = \max_{u,v \in V} \text{dist}(u, v)$$

   where $\text{dist}(u, v)$ represents the shortest path distance between vertices u and v.

8. **Articulation Points and Bridges**
   Using DFS, find all articulation points and bridges in an undirected graph.

More Applications of BFS and DFS

Problem 1: Minimum Steps in a Grid

You are given an $N \times M$ grid with obstacles. Each cell is either empty (.) or blocked (#). Find the minimum number of steps required to reach from the top-left corner $(0, 0)$ to the bottom-right corner $(N-1, M-1)$ using BFS. You are allowed moving up, down, left, or right (no diagonals). If not possible, print `-1`.

Input:

```
5 5
.....
..#..
..#..
.....
.....
```

Output:

```
8
```

Example 2:

```
Input:
3 3
.##
#.#
##.

Output:
-1
```

Hint: Use a BFS starting from $(0, 0)$ to compute the minimum distance to all reachable cells. Maintain a 2D `dist` array initialized to -1 and update neighbors $(x + dx, y + dy)$ only if they are inside the grid, not blocked, and not yet visited. The answer will be `dist[N-1][M-1]`, or `-1` if it remains unreachable.

Problem 2: Knight Moves on a Chessboard

You are given the coordinates of a knight and a target position on an $N \times N$ chessboard. Find the minimum number of moves required for the knight to reach the target. Use BFS on an implicit graph where each vertex represents a board position.

Input:

```
N=8
0 0
7 7
```

Output:

```
6
```

Problem 3: Number of Enclaves

You are given a binary grid of size $N \times M$, where:

- `1` represents land, and
- `0` represents water.

A land cell is said to be *connected to the boundary* if it can reach any boundary cell (first or last row, or first or last column) by moving through adjacent land cells in the four cardinal directions (up, down, left, right).

Your task is to count the number of land cells that **cannot reach the boundary** of the grid.

Example:

```
Input:
4 5
0 0 0 0 0
1 0 1 0 0
0 1 1 0 0
0 0 0 0 0

Output:
3
```

Explanation: The boundary land cell at position $(1, 0)$ can reach the boundary directly. The remaining three land cells form an enclosed region in the center and cannot reach any edge.

Hint: Think from the *outside-in*. All land cells that are reachable from the boundary are not enclaves. Start a **multi-source BFS (or DFS)** from every boundary cell containing land (`1`). Mark all land cells reachable from these boundary sources as visited. After the traversal, count the land cells that remain unvisited — these are the enclaves.

—

Problem 4: Escape the Fire

A person is trapped in a maze where fire is spreading every minute. The maze is represented by an $N \times M$ grid consisting of:

[noitemsep]

- . — an empty cell,

- `'(#)'` — a wall (cannot be passed through),
- `'P'` — the initial position of the person,
- `'F'` — the initial position(s) of fire.

Each minute:

- The person may move one step up, down, left, or right into an empty cell.
- The fire spreads simultaneously to all adjacent empty cells (up, down, left, right).

The person escapes if they can reach **any boundary cell** (cell on the edge of the grid) **before or at the same time** the fire would reach that cell.

Determine whether the person can escape.

Input Format:

- The first line contains two integers N and M — the dimensions of the grid.
- The next N lines each contain M characters: `'.'`, `'(#)'`, `'P'`, `'F'`.

Output Format:

- Print `YES` if the person can reach a boundary cell before being caught by the fire.
- Otherwise, print `NO`.

Example:

```
Input:
4 4
P...
.##.
.F#.
....

Output:
YES
```

Explanation: The fire starts from cell $(2, 1)$ and spreads each minute to adjacent open cells. The person starts from $(0, 0)$ and can immediately move towards the boundary edge (already at top row). Thus, the person escapes successfully before the fire reaches.

Hint: Perform two BFS runs:

(a) First, run a **multi-source BFS** starting from all fire cells (`F`) to compute the time $t_F[x][y]$ when the fire reaches each cell.

(b) Then, run another BFS from the person's start position (`P`) to compute the earliest time $t_P[x][y]$ the person can reach each cell.

The person can safely move into a cell (x, y) only if either:

$$t_P[x][y] < t_F[x][y]$$

or the cell is never reached by fire (i.e., $t_F[x][y] = \infty$). If any boundary cell satisfies this condition, print `YES`; otherwise, `NO`. —

Problem 5: Shortest Path with One Wall Break

In a grid of 0s and 1s (1 = wall), you can break at most one wall and move up, down, left, or right. Find the minimum number of steps to reach from $(0, 0)$ to $(N - 1, M - 1)$.

Hint: Use BFS with an extra dimension: $(x, y, wall\_broken)$.

—