# Graph Problem Set

## Problem 1: Check Bipartite Graph

Given an undirected graph, check if the graph is bipartite. A bipartite graph is a graph whose set of vertices can be divided into two disjoint sets such that every edge connects a vertex in one set to a vertex in the other set.

| Sample Input | Sample Output |
|---|---|
| graph = [[0, 1, 0], [1, 0, 1], [0, 1, 0]] | The graph is bipartite |

## Problem 2: Check if Graph is Connected

Given a graph, check if the graph is connected. A connected graph is a graph in which there is a path between every pair of vertices.

| Sample Input | Sample Output |
|---|---|
| graph = [[0, 1, 0, 0], [1, 0, 1, 0], [0, 1, 0, 1], [0, 0, 1, 0]] | The graph is connected |

## Problem 3: Find the Shortest Path (BFS)

Given a graph and a starting node, find the shortest path from the start node to all other nodes using BFS.

| Sample Input | Sample Output |
|---|---|
| graph = [[0,1,0,0], [1,0,1,1], [0,1,0,1], [0,1,1,0]], start = 0 | Shortest path from 0: [0, 1, 2, 3] |

## Problem 4: Number of Islands (DFS on Matrix)

Given a 2D matrix representing a grid of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically.

| Sample Input | Sample Output |
|---|---|
| grid = [[1, 1, 0, 0, 0], [1, 1, 0, 1, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 1]] | Number of islands: 3 |

## Problem 5: Find the Connected Components

Given an undirected graph, find all connected components. A connected component is a subset of the graph such that there is a path between any two vertices in this subset.

| Sample Input | Sample Output |
|---|---|
| graph = [[0, 1, 0], [1, 0, 1], [0, 1, 0]] | Connected components: 0, 1, 2 |

## Problem 6: Cycle Detection in Directed Graph

Given a directed graph, detect if there is a cycle in the graph using DFS. A cycle exists in the graph if there is a path starting and ending at the same vertex.

| Sample Input | Sample Output |
|---|---|
| graph = [[0, 1], [1, 2], [2, 0]] | The graph contains a cycle |

## Problem 7: Topological Sort

Given a directed acyclic graph (DAG), return a topological ordering of its vertices. Topological sort is possible only for directed acyclic graphs.

| Sample Input | Sample Output |
|---|---|
| graph = [(0, 1), (0, 2), (1, 3), (2, 3)] | Topological sort: 0, 1, 2, 3 |

## Problem 8: Word Search in 2D Grid

Given a 2D grid of characters and a word, check if the word can be formed by consecutive letters in the grid (horizontally, vertically, or diagonally).

| Sample Input | Sample Output |
|---|---|
| grid = [['A', 'B', 'C', 'E'], ['S', 'F', 'C', 'S'], ['A', 'D', 'E', 'E']], word = 'ABCCED' | Word 'ABCCED' exists in the grid |

## Problem 8: Find the Treasure in the Maze

Given a 2D matrix representing a maze, where '0' represents a free cell and '1' represents a blocked cell, find the path from the top-left corner to the treasure (marked as ‘999’). You can move up, down, left, or right.

| Sample Input | Sample Output |
|---|---|
| maze = [[0, 0, 0, 0], [0, 1, 1, 0], [0, 1,999, 0], [0, 0, 0, 1]] | Path to treasure: (0,0), (0,1), (0,2), (0,3), (1,3), (2,3), (2,2) |

## Problem 10: Maze Solver (DFS in Matrix)

Given a maze represented as a 2D matrix, find a path from the top-left corner to the bottom-right corner. The path can only move through '1's (open spaces) and not through '0's (walls).

| Sample Input | Sample Output |
|---|---|
| maze = [[1, 0, 0, 0], [1, 1, 1, 0], [0, 1, 0, 0], [0, 1, 1, 1]] | Path from (0,0) to (3,3): (0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3) |