EXPERIMENT - 9

Student Name: Yash Dixit
UID: 22BCS10730
Branch: BE-CSE
Section/Group: IOT-605'A'
Semester: 6th
Date of Performance: 02/04/25
Subject Name: Advance Programming
Subject Code: 22CSP-367

✨ Graphs Problems ✨

Day 17 (Graphs - Medium & Hard)

1. Number of Islands (Twitter)

Link

```
python
CopyEdit
def numIslands(grid):
    if not grid:
        return 0

    rows, cols = len(grid), len(grid[0])
    visited = set()
    islands = 0

    def dfs(r, c):
        if (r < 0 or c < 0 or
            r >= rows or c >= cols or
            grid[r][c] == "0" or
            (r, c) in visited):
            return
        visited.add((r, c))
        dfs(r+1, c)
        dfs(r-1, c)
        dfs(r, c+1)
        dfs(r, c-1)

    for r in range(rows):
        for c in range(cols):
            if grid[r][c] == "1" and (r, c) not in visited:
                dfs(r, c)
                islands += 1
    return islands
```

---

2. Word Ladder (Facebook)

Link

```
python
CopyEdit
from collections import deque

def ladderLength(beginWord, endWord, wordList):
    wordSet = set(wordList)
    if endWord not in wordSet:
```

```
        return 0

    queue = deque([(beginWord, 1)])
    while queue:
        word, steps = queue.popleft()
        if word == endWord:
            return steps

        for i in range(len(word)):
            for c in "abcdefghijklmnopqrstuvwxyz":
                nextWord = word[:i] + c + word[i+1:]
                if nextWord in wordSet:
                    wordSet.remove(nextWord)
                    queue.append((nextWord, steps + 1))
    return 0
```

Day 18 (Graphs)

3. Surrounded Regions (LinkedIn)

Link

```
python
CopyEdit
def solve(board):
    if not board:
        return

    rows, cols = len(board), len(board[0])

    def dfs(r, c):
        if (r < 0 or c < 0 or
            r >= rows or c >= cols or
            board[r][c] != 'O'):
            return
        board[r][c] = 'T'
        dfs(r+1, c)
        dfs(r-1, c)
        dfs(r, c+1)
        dfs(r, c-1)

    for r in range(rows):
        for c in [0, cols-1]:
            if board[r][c] == 'O':
                dfs(r, c)
    for c in range(cols):
        for r in [0, rows-1]:
            if board[r][c] == 'O':
                dfs(r, c)

    for r in range(rows):
        for c in range(cols):
            if board[r][c] == 'O':
                board[r][c] = 'X'
            if board[r][c] == 'T':
                board[r][c] = 'O'
```

---

4. Binary Tree Maximum Path Sum (Amazon)

Link

```
python
CopyEdit
class Solution:
    def maxPathSum(self, root):
        self.max_sum = float('-inf')

        def dfs(node):
            if not node:
                return 0
            left = max(dfs(node.left), 0)
            right = max(dfs(node.right), 0)
            self.max_sum = max(self.max_sum, node.val + left + right)
            return node.val + max(left, right)

        dfs(root)
        return self.max_sum
```

Day 19 (Graphs)

5. Friend Circles (Apple)

Link

```
python
CopyEdit
def findCircleNum(M):
    n = len(M)
    visited = set()

    def dfs(i):
        for j in range(n):
            if M[i][j] == 1 and j not in visited:
                visited.add(j)
                dfs(j)

    count = 0
    for i in range(n):
        if i not in visited:
            dfs(i)
            count += 1
    return count
```

---

6. Lowest Common Ancestor of a Binary Tree (Amazon)

Link

```
python
CopyEdit
class Solution:
    def lowestCommonAncestor(self, root, p, q):
        if not root or root == p or root == q:
            return root

        left = self.lowestCommonAncestor(root.left, p, q)
```

```
        right = self.lowestCommonAncestor(root.right, p, q)

        if left and right:
            return root
        return left if left else right
```

Day 20 (Graphs)

7. Course Schedule (Apple)

Link

```
python
CopyEdit
def canFinish(numCourses, prerequisites):
    from collections import defaultdict

    graph = defaultdict(list)
    for a, b in prerequisites:
        graph[a].append(b)

    visiting = set()

    def dfs(course):
        if course in visiting:
            return False
        if graph[course] == []:
            return True

        visiting.add(course)
        for prereq in graph[course]:
            if not dfs(prereq):
                return False
        visiting.remove(course)
        graph[course] = []
        return True

    for course in range(numCourses):
        if not dfs(course):
            return False
    return True
```

---

8. Longest Increasing Path in a Matrix (LinkedIn)

Link

```
python
CopyEdit
def longestIncreasingPath(matrix):
    if not matrix:
        return 0

    rows, cols = len(matrix), len(matrix[0])
    cache = [[-1] * cols for _ in range(rows)]

    def dfs(r, c, prevVal):
        if r < 0 or c < 0 or r >= rows or c >= cols or matrix[r][c] <= prevVal:
            return 0
        if cache[r][c] != -1:
```

```
            return cache[r][c]

        res = 1 + max(
            dfs(r+1, c, matrix[r][c]),
            dfs(r-1, c, matrix[r][c]),
            dfs(r, c+1, matrix[r][c]),
            dfs(r, c-1, matrix[r][c])
        )
        cache[r][c] = res
        return res

    ans = 0
    for r in range(rows):
        for c in range(cols):
            ans = max(ans, dfs(r, c, -float('inf')))
    return ans
```

---

9. Course Schedule 2 (Netflix)

Link

```
python
CopyEdit
def findOrder(numCourses, prerequisites):
    from collections import defaultdict, deque

    graph = defaultdict(list)
    indegree = [0] * numCourses
    for a, b in prerequisites:
        graph[b].append(a)
        indegree[a] += 1

    queue = deque([i for i in range(numCourses) if indegree[i] == 0])
    res = []

    while queue:
        course = queue.popleft()
        res.append(course)
        for neighbor in graph[course]:
            indegree[neighbor] -= 1
            if indegree[neighbor] == 0:
                queue.append(neighbor)

    return res if len(res) == numCourses else []
```