Lab Practice Sheet 5: BFS & DFS Graph Algorithms

CS G526: Advanced Algorithms & Complexity

Instructor: Tulasimohan Molli **Date:** 2026-10-26

! Evaluation Instructions for Students

- **Practice all 8 problems below.**
- In the next lab session, **one question will be chosen at random** for on-the-spot coding and evaluation.
- Your code will be evaluated by running it against automated hidden test cases.
- You may write your solutions in **Python**, **C++**, or **Java**.
- **Library Usage:** You are allowed and encouraged to use built-in standard libraries for data structures (e.g., `vector`, `queue`, `stack` in C++, `collections.deque` in Python, `ArrayList`, `ArrayDeque` in Java).
- Standard I/O (reading from standard input `stdin` and printing to standard output `stdout`) is required.

Q1: Number of Connected Components in an Undirected Graph

Description: Given an undirected graph with V vertices (labeled 0 to $V-1$) and E edges, find the total number of connected components in the graph using DFS or BFS.

- **Input Format:**
  - Line 1: Two space-separated integers V and E.
  - Next E lines: Two space-separated integers u and v representing an undirected edge between u and v.
- **Output Format:**
  - A single integer representing the number of connected components.

i Sample Input 1

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

i Sample Output 1

```
2
```

i Sample Input 2

```
4 0
```

i Sample Output 2

```
4
```

Q2: Shortest Path in Unweighted Graph (BFS)

Description: Given an unweighted, undirected graph with V vertices (0 to $V-1$) and E edges, find the length of the shortest path (minimum number of edges) from a given source vertex S to a destination vertex D. If D is unreachable from S, print `-1`.

- **Input Format:**
  - Line 1: Two integers V and E.
  - Next E lines: Two integers u and v representing an undirected edge.
  - Next line: Two integers S and D, the source and destination vertices.
- **Output Format:**
  - A single integer representing the shortest path distance, or `-1`.

Sample Input

```
6 7
0 1
0 2
1 3
2 3
2 4
3 5
4 5
0 5
```

Sample Output

```
3
```

Q3: Shortest Path in a Grid Maze with Obstacles

Description: Given an $M \times N$ binary grid where `0` represents an empty cell and `1` represents an obstacle, find the length of the shortest path from the top-left cell `(0, 0)` to the bottom-right cell `(M-1, N-1)`. You can move in 4 cardinal directions (Up, Down, Left, Right). If the starting or destination cell is blocked, or if no valid path exists, output `-1`. The path length is measured as the number of cells visited (including start and end).

- **Input Format:**
  - Line 1: Two space-separated integers M and N.
  - Next M lines: N space-separated integers (`0` or `1`) representing the grid.
- **Output Format:**
  - A single integer representing the minimum number of cells on the path, or `-1`.

Sample Input 1

```
3 4
0 0 0 0
0 1 1 0
0 0 0 0
```

Sample Output 1

```
6
```

Sample Input 2

```
3 3
0 1 0
1 1 0
0 0 0
```

Sample Output 2

```
-1
```

Q4: Number of Islands (2D Grid Connected Components)

Description: Given an $M \times N$ 2D grid map 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 (4-directional).

- **Input Format:**
  - Line 1: Two space-separated integers M and N.
  - Next M lines: A string of length N containing only `'0'` and `'1'`.
- **Output Format:**
  - A single integer representing the number of islands.

Sample Input 1

```
4 5
11110
11010
11000
00000
```

Sample Output 1

```
1
```

Sample Input 2

```
4 5
11000
11000
00100
00011
```

Sample Output 2

```
3
```

Q5: Cycle Detection in a Directed Graph

Description: Given a directed graph with V vertices (0 to $V-1$) and E directed edges, determine whether the graph contains at least one directed cycle. Use DFS with a recursion stack / 3-color vertex marking scheme (`WHITE`, `GRAY`, `BLACK`).

- **Input Format:**
  - Line 1: Two space-separated integers V and E.
  - Next E lines: Two space-separated integers u and v representing a directed edge from u to v.
- **Output Format:**
  - Print `True` if a directed cycle exists, or `False` otherwise.

Sample Input 1

```
4 4
0 1
1 2
2 3
3 1
```

Sample Output 1

```
True
```

Sample Input 2

```
4 3
0 1
1 2
2 3
```

Sample Output 2

```
False
```

Q6: Topological Sort & Course Schedule (Kahn's Algorithm)

Description: There are a total of N courses you have to take, labeled from 0 to $N-1$. Some courses have prerequisites given as directed edges: a pair $[u, v]$ indicates that course v must be taken before course u ($v \to u$).

Determine a valid ordering of courses you should take to finish all courses. If multiple valid orderings exist, output the one that is lexicographically smallest. If it is impossible to finish all courses due to a cyclic dependency, print `Impossible`.

- **Input Format:**
  - Line 1: Two integers N (number of courses) and M (number of prerequisite dependencies).
  - Next M lines: Two integers u and v indicating dependency $v \to u$ (v is taken before u).
- **Output Format:**
  - A single line containing N space-separated integers representing the course order, or `Impossible`.

Sample Input 1

```
4 4
1 0
2 0
3 1
3 2
```

Sample Output 1

```
0 1 2 3
```

Sample Input 2

```
2 2
1 0
0 1
```

Sample Output 2

```
Impossible
```

Q7: Bipartite Graph Verification (2-Coloring via BFS)

Description: Given an undirected graph with V vertices (0 to $V-1$) and E edges, determine if the graph is **Bipartite**. A graph is bipartite if its vertices can be partitioned into two independent sets U and V such that every edge connects a vertex in U to a vertex in V (i.e., graph has no odd-length cycle).

- **Input Format:**
  - Line 1: Two space-separated integers V and E.
  - Next E lines: Two integers u and v representing an undirected edge.
- **Output Format:**
  - Print `True` if the graph is bipartite, or `False` otherwise.

Sample Input 1

```
4 4
0 1
1 2
2 3
3 0
```

Sample Output 1

```
True
```

Sample Input 2

```
3 3
0 1
1 2
2 0
```

Sample Output 2

```
False
```

Q8: Rotting Oranges (Multi-Source BFS)

Description: You are given an $M \times N$ grid where each cell can have one of three values: - `0`: Empty cell - `1`: Fresh orange - `2`: Rotten orange

Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten. Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return `-1`.

- **Input Format:**
  - Line 1: Two integers M and N.
  - Next M lines: N space-separated integers (`0`, `1`, or `2`).
- **Output Format:**
  - A single integer representing the minimum elapsed minutes, or `-1`.

Sample Input 1

```
3 3
2 1 1
1 1 0
0 1 1
```

Sample Output 1

```
4
```

Sample Input 2

```
3 3
2 1 1
0 1 1
1 0 1
```

Sample Output 2

```
-1
```

Sample Input 3

```
1 2
0 2
```

Sample Output 3

```
0
```