2023-08-19 - Handout – Union Find

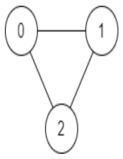
Q1. Find if path exists in Graph

Link: https://leetcode.com/problems/find-if-path-exists-in-graph/

There is a bi-directional graph with n vertices, where each vertex is labeled from 0 to n - 1 (inclusive). The edges in the graph are represented as a 2D integer array edges, where each edges[i] = [ui, vi] denotes a bi-directional edge between vertex ui and vertex vi. Every vertex pair is connected by at most one edge, and no vertex has an edge to itself. You want to determine if there is a valid path that exists from vertex source to vertex destination.

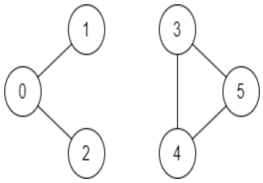
Given edges and the integers n, source, and destination, return true if there is a valid path from source to destination, or false otherwise.

Example 1:




```
Input: n = 3, edges = [[0,1],[1,2],[2,0]], source = 0, destination = 2
Output: true
Explanation: There are two paths from vertex 0 to vertex 2:
- 0 → 1 → 2
- 0 → 2
```

Example 2:



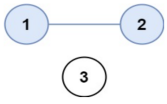

```
Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]], source = 0,
destination = 5
Output: false
Explanation: There is no path from vertex 0 to vertex 5.
```

Q2. Number of provinces

Link: https://leetcode.com/problems/number-of-provinces/

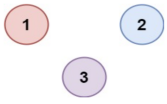
There are n cities. Some of them are connected, while some are not. If city a is connected directly with city b, and city b is connected directly with city c, then city a is connected indirectly with city c. A **province** is a group of directly or indirectly connected cities and no other cities outside of the group. You are given an n x n matrix isConnected where isConnected[i][j] = 1 if the i_{th} city and the j_{th} city are directly connected, and isConnected[i][j] = 0 otherwise. Return *the total number of* ***provinces***.

Example 1:




```
Input: isConnected = [[1,1,0],[1,1,0],[0,0,1]]
Output: 2
```

Example 2:




```
Input: isConnected = [[1,0,0],[0,1,0],[0,0,1]]
Output: 3
```

Q3. Longest Consecutive Sequence

Link: https://leetcode.com/problems/longest-consecutive-sequence/

Given an unsorted array of integers nums, return *the length of the longest consecutive elements sequence.* You must write an algorithm that runs in O(n) time.

Example 1:

Input: nums = [100,4,200,1,3,2]
Output: 4
Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

Example 2:

Input: nums = [0,3,7,2,5,8,4,6,0,1]
Output: 9

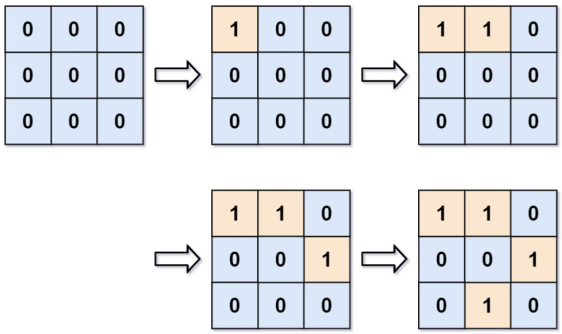
Q4. Number of Islands II

Link: https://leetcode.com/problems/number-of-islands-ii/

You are given an empty 2D binary grid grid of size m x n. The grid represents a map where 0's represent water and 1's represent land. Initially, all the cells of grid are water cells (i.e., all the cells are 0's). We may perform an add land operation which turns the water at position into a land. You are given an array positions where positions[i] = [r_i, c_i] is the position (r_i, c_i) at which we should operate the i_{th} operation.

Return *an array of integers* answer *where* answer[i] *is the number of islands after turning the cell* (r_i, c_i) *into a land*. An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:



Input: m = 3, n = 3, positions = [[0,0],[0,1],[1,2],[2,1]]
Output: [1,1,2,3]
Explanation:
Initially, the 2d grid is filled with water.
- Operation #1: addLand(0, 0) turns the water at grid[0][0] into a land. We have 1 island.
- Operation #2: addLand(0, 1) turns the water at grid[0][1] into a land. We still have 1 island.
- Operation #3: addLand(1, 2) turns the water at grid[1][2] into a land. We have 2 islands.
- Operation #4: addLand(2, 1) turns the water at grid[2][1] into a land. We have 3 islands.

Example 2:

Input: m = 1, n = 1, positions = [[0,0]] **Output:** [1]