Assignment-8

Total Score: 70

Due Date : 4th December 2021

Question 1: A trie (pronounced as "try") or prefix tree is a tree data structure used to efficiently store and retrieve keys in a dataset of strings. There are various applications of this data structure, such as autocomplete and spellchecker.

Implement the Trie class:

- Trie() Initializes the trie object.
- void insert(String word) Inserts the string word into the trie.
- boolean search(String word) Returns true if the string word is in the trie (i.e., was inserted before), and false otherwise.
- boolean startsWith(String prefix) Returns true if there is a previously inserted string word that has the prefix prefix, and false otherwise.

Example 1:

```
Input

["Trie", "insert", "search", "search", "startsWith", "insert", "search"]

[[], ["apple"], ["apple"], ["app"], ["app"], ["app"], ["app"]]

Output

[null, null, true, false, true, null, true]
```

```
Explanation

Trie trie = new Trie();

trie.insert("apple");

trie.search("apple");   // return True

trie.search("app");     // return False

trie.startsWith("app"); // return True

trie.insert("app");

trie.search("app");     // return True
```

Constraints:

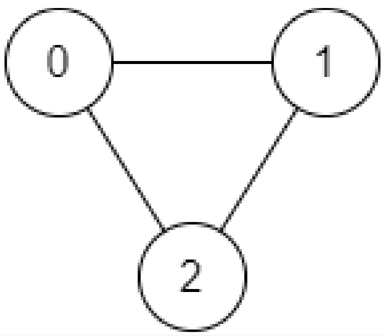
- 1 <= word.length, prefix.length <= 2000
- word and prefix consist only of lowercase English letters.
- At most $3 * 10^4$ calls in total will be made to insert, search, and startsWith.

Question 2: 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] = [u_i, v_i] denotes a bi-directional edge between vertex u_i and vertex v_i. 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 start to vertex end.

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

Example 1:



Input: n = 3, edges = [[0,1],[1,2],[2,0]], start = 0, end = 2

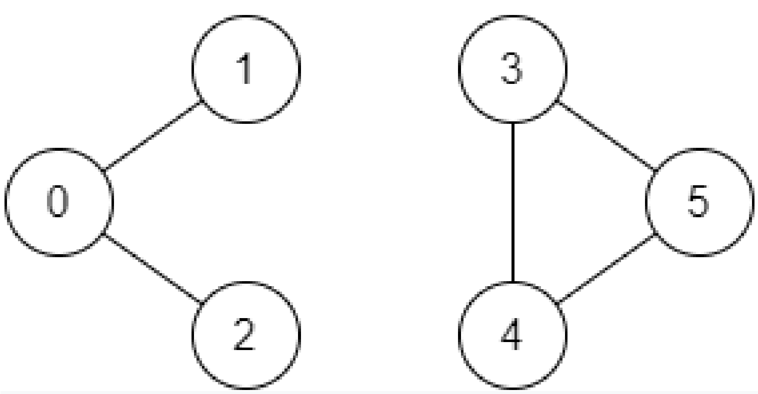
Output: true

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]], start = 0, end = 5

Output: false

Explanation: There is no path from vertex 0 to vertex 5.

Constraints:

- $1 <= n <= 2 * 10^5$
- $0 <= edges.length <= 2 * 10^5$
- edges[i].length == 2
- $0 <= u_i, v_i <= n - 1$
- $u_i != v_i$
- 0 <= start, end <= n - 1
- There are no duplicate edges.
- There are no self edges.

Question 3: Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return the number of islands.

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: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
```

```
]

Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
Output: 3
```

Constraints:

- m == grid.length
- n == grid[i].length
- 1 <= m, n <= 300
- grid[i][j] is '0' or '1'.

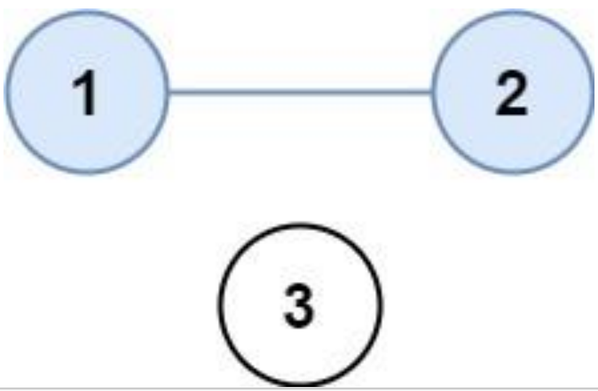
Question 4: There are n cities. Some of them are connected, while some are not. If city a is connected directly with city b, and city b is connected directly with city c, then city a is connected indirectly with city c.

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.

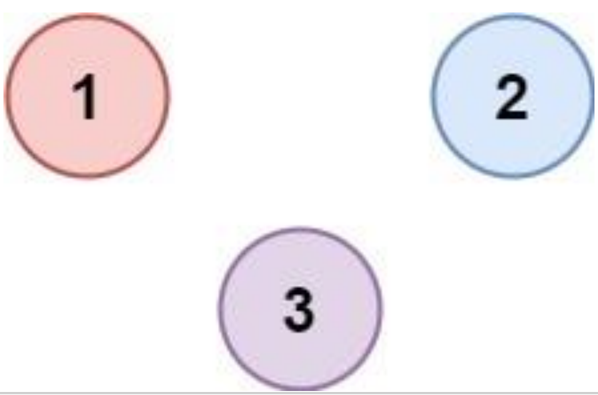
Example 1:



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

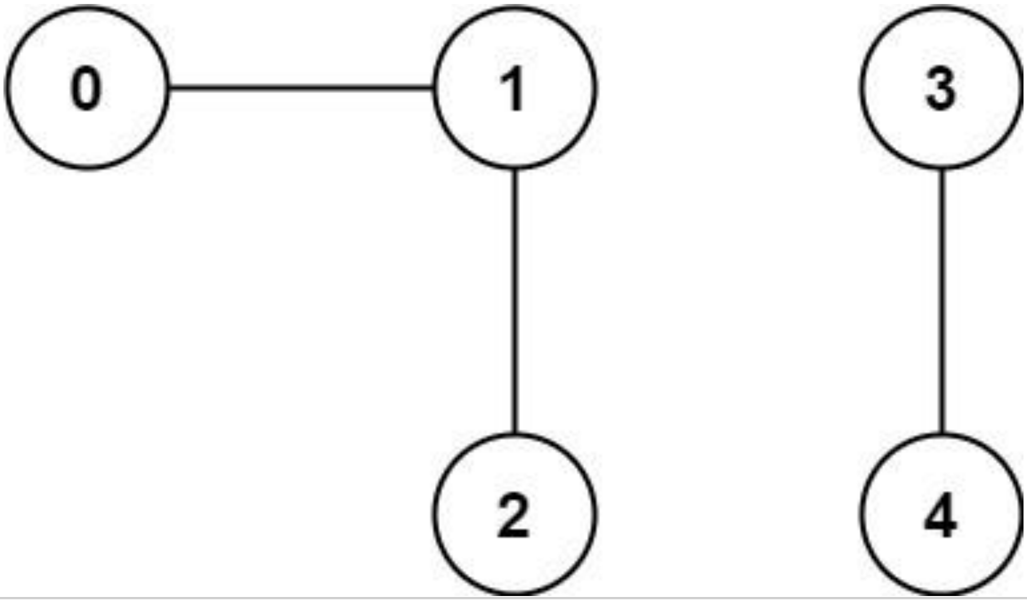
Constraints:

- 1 <= n <= 200
- n == isConnected.length
- n == isConnected[i].length
- isConnected[i][j] is 1 or 0.
- isConnected[i][i] == 1
- isConnected[i][j] == isConnected[j][i]

Question 5: You have a graph of n nodes. You are given an integer n and an array edges where edges[i] = [a_i, b_i] indicates that there is an edge between a_i and b_i in the graph.

Return the number of connected components in the graph.

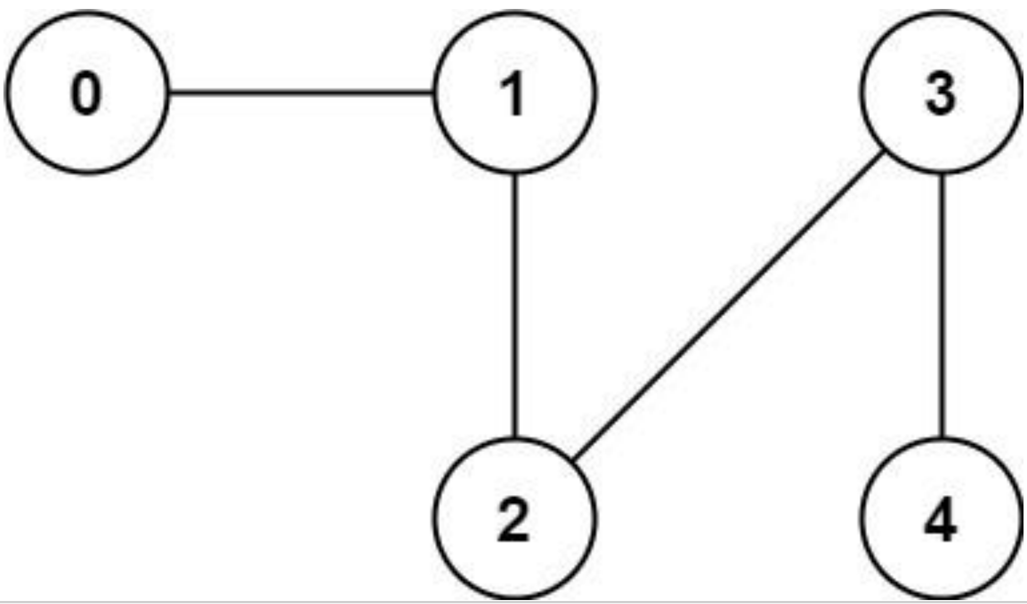
Example 1:



Input: n = 5, edges = [[0,1],[1,2],[3,4]]

Output: 2

Example 2:



Input: n = 5, edges = [[0,1],[1,2],[2,3],[3,4]]

Output: 1

Constraints:

- 1 <= n <= 2000
- 1 <= edges.length <= 5000
- edges[i].length == 2
- 0 <= a_i <= b_i < n
- a_i != b_i
- There are no repeated edges.

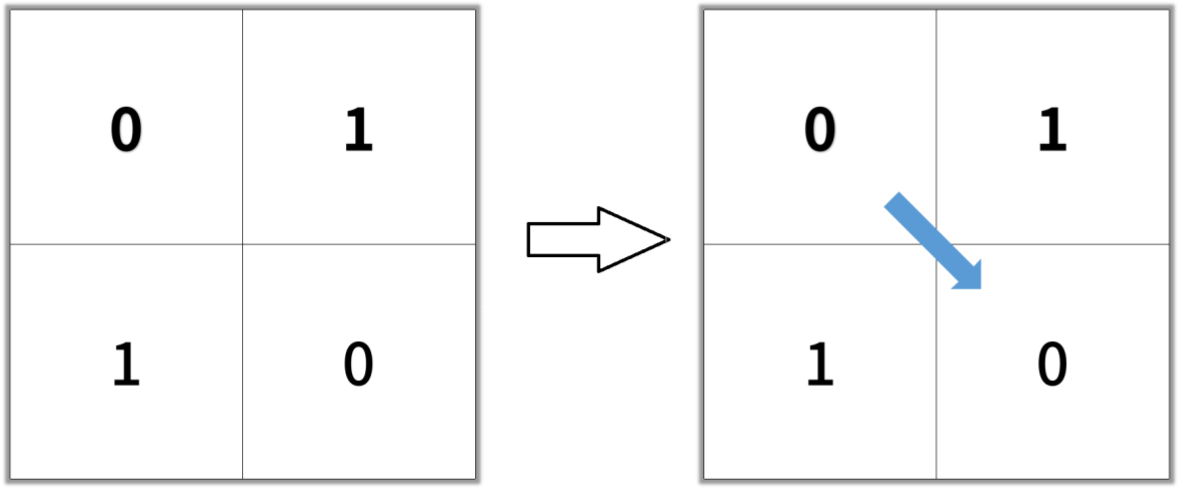
Question 6: Given an n x n binary matrix grid, return the length of the shortest clear path in the matrix. If there is no clear path, return -1.

A clear path in a binary matrix is a path from the top-left cell (i.e., (0, 0)) to the bottom-right cell (i.e., (n - 1, n - 1)) such that:

- All the visited cells of the path are 0.
- All the adjacent cells of the path are 8-directionally connected (i.e., they are different and they share an edge or a corner).

The length of a clear path is the number of visited cells of this path.

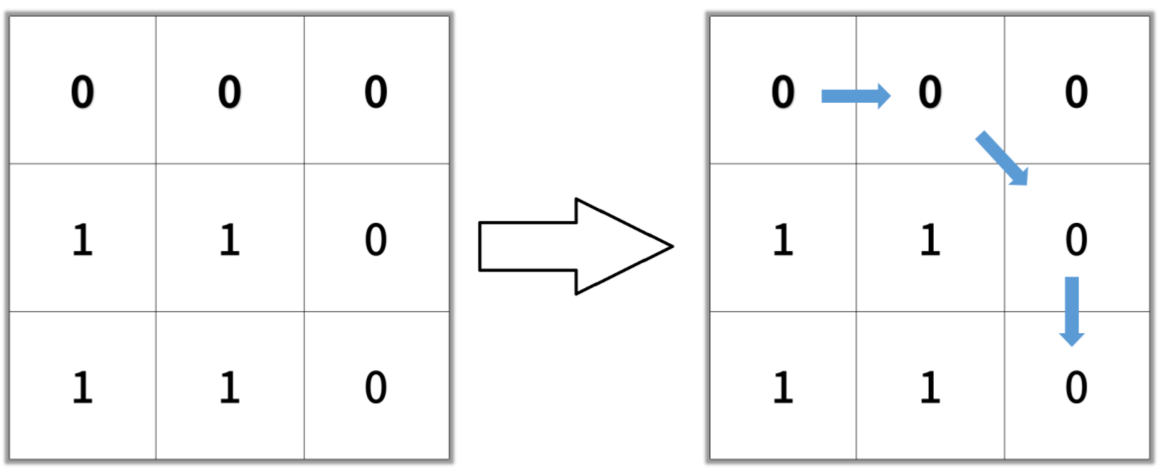
Example 1:



Input: grid = [[0,1],[1,0]]

Output: 2

Example 2:



Input: grid = [[0,0,0],[1,1,0],[1,1,0]]

Output: 4

Example 3:

```
Input: grid = [[1,0,0],[1,1,0],[1,1,0]]

Output: -1
```

Constraints:

- n == grid.length
- n == grid[i].length
- 1 <= n <= 100
- grid[i][j] is 0 or 1

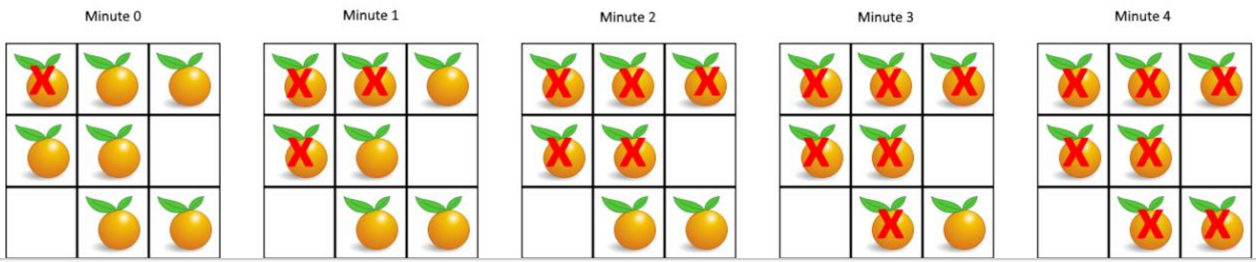
Question 7: You are given an m x n grid where each cell can have one of three values:

- 0 representing an empty cell,
- 1 representing a fresh orange, or
- 2 representing a 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.

Example 1:




```
Input: grid = [[2,1,1],[1,1,0],[0,1,1]]

Output: 4
```

Example 2:

```
Input: grid = [[2,1,1],[0,1,1],[1,0,1]]

Output: -1

Explanation: The orange in the bottom left corner (row 2, column 0) is n
ever rotten, because rotting only happens 4-directionally.
```

Example 3:

```
Input: grid = [[0,2]]

Output: 0

Explanation: Since there are already no fresh oranges at minute 0, the a
nswer is just 0.
```

Constraints:

- m == grid.length
- n == grid[i].length
- 1 <= m, n <= 10
- grid[i][j] is 0, 1, or 2.