2020-03-21 - Handout – Graphs

Q1. Find the Town Judge

Link: https://leetcode.com/problems/find-the-town-judge/

In a town, there are N people labelled from 1 to N. There is a rumor that one of these people is secretly the town judge.

If the town judge exists, then:

1. The town judge trusts nobody.
2. Everybody (except for the town judge) trusts the town judge.
3. There is exactly one person that satisfies properties 1 and 2.

You are given trust, an array of pairs trust[i] = [a, b] representing that the person labelled a trusts the person labelled b.

If the town judge exists and can be identified, return the label of the town judge. Otherwise, return -1.

Example 1:

```
Input: N = 2, trust = [[1,2]]

Output: 2
```

Example 2:

```
Input: N = 3, trust = [[1,3],[2,3]]

Output: 3
```

Example 3:

```
Input: N = 3, trust = [[1,3],[2,3],[3,1]]

Output: -1
```

Example 4:

```
Input: N = 4, trust =
[[1,3],[1,4],[2,3],[2,4],[4,3]]

Output: 3
```

Q2. Evaluate Division

Link: https://leetcode.com/problems/evaluate-division/

Equations are given in the format A / B = k, where A and B are variables represented as strings, and k is a real number (floating point number). Given some queries, return the answers. If the answer does not exist, return -1.0.

Example:

Given a / b = 2.0, b / c = 3.0.
queries are: a / c = ?, b / a = ?, a / e = ?, a / a = ?, x / x = ? .
return [6.0, 0.5, -1.0, 1.0, -1.0].

The input is: vector<pair<string, string>> equations, vector<double>& values, vector<pair<string, string>> queries , where equations.size() == values.size(), and the values are positive. This represents the equations. Return vector<double>.

According to the example above:

equations = [["a", "b"], ["b", "c"]],

values = [2.0, 3.0],

queries = [["a", "c"], ["b", "a"], ["a", "e"], ["a", "a"], ["x", "x"]].

The input is always valid. You may assume that evaluating the queries will result in no division by zero and there is no contradiction.

Q3. Parallel Courses

Link: https://leetcode.com/problems/parallel-courses/

There are N courses, labelled from 1 to N.

We are given relations[i] = [X, Y], representing a prerequisite relationship between course X and course Y: course X has to be studied before course Y.

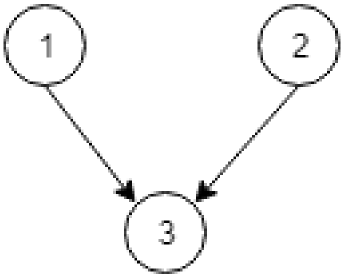
In one semester you can study any number of courses as long as you have studied all the prerequisites for the course you are studying.

Return the minimum number of semesters needed to study all courses. If there is no way to study all the courses, return -1.

Note:

- 1 <= N <= 5000
- 1 <= relations.length <= 5000
- relations[i][0] != relations[i][1]
- There are no repeated relations in the input

Example 1:



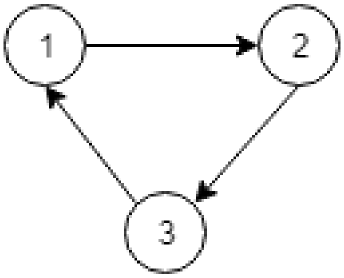
Input: N = 3, relations = [[1,3],[2,3]]

Output: 2

Explanation:

In the first semester, courses 1 and 2 are studied. In the second semester, course 3 is studied.

Example 2:



Input: N = 3, relations = [[1,2],[2,3],[3,1]]

Output: -1

Explanation:

No course can be studied because they depend on each other.