Description | Editorial | Solutions (1.2K) | Submissions

1615. Maximal Network Rank

Hint

Medium 2K 312

Companies

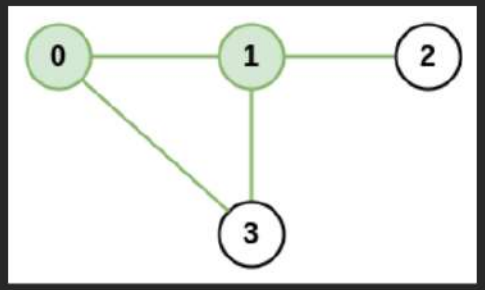
There is an infrastructure of `n` cities with some number of `roads` connecting these cities. Each `roads[i] = [a_i, b_i]` indicates that there is a bidirectional road between cities `a_i` and `b_i`.

The **network rank** of **two different cities** is defined as the total number of **directly** connected roads to **either** city. If a road is directly connected to both cities, it is only counted **once**.

The **maximal network rank** of the infrastructure is the **maximum network rank** of all pairs of different cities.

Given the integer `n` and the array `roads`, return *the **maximal network rank** of the entire infrastructure.*

Example 1:



Input: `n = 4, roads = [[0,1],[0,3],[1,2],[1,3]]`
Output: `4`
Explanation: The network rank of cities 0 and 1 is 4 as there

Java | Auto

```java
class Solution {
    public int maximalNetworkRank(int n, int[][] roads) {
        int[] degree = new int[n];
        Set<String> roadSet = new HashSet<>();

        for (int[] road : roads) {
            degree[road[0]]++;
            degree[road[1]]++;
            roadSet.add(road[0] + "," + road[1]);
            roadSet.add(road[1] + "," + road[0]);
        }

        int maxRank = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i+1; j < n; j++) {
                int rank = degree[i] + degree[j];
```

Ln 25, Col 6

Testcase | Result

Accepted Runtime: 7 ms

• Case 1 • Case 2 • Case 3

Input

n =
4

roads =
[[0,1],[0,3],[1,2],[1,3]]

Output

4

Console

Run Submit

Description | Editorial | Solutions (2.2K) | Submissions

1614. Maximum Nesting Depth of the Parentheses

Hint

Easy 1.6K 253

Companies

A string is a **valid parentheses string** (denoted **VPS**) if it meets one of the following:

- It is an empty string `""`, or a single character not equal to `"("` or `")"`,
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are **VPS**'s, or
- It can be written as `(A)`, where `A` is a **VPS**.

We can similarly define the **nesting depth** `depth(S)` of any VPS `S` as follows:

- `depth("") = 0`
- `depth(C) = 0`, where `C` is a string with a single character not equal to `"("` or `")"`.
- `depth(A + B) = max(depth(A), depth(B))`, where `A` and `B` are **VPS**'s.
- `depth("(" + A + ")") = 1 + depth(A)`, where `A` is a **VPS**.

For example, `""`, `"()()"`, and `"()(()())"` are **VPS**'s (with nesting depths 0, 1, and 2), and `")("` and `"(()"` are not **VPS**'s.

Given a **VPS** represented as string `s`, return *the **nesting depth** of* `s`.

Example 1:

```
Input: s = "(1+(2*3)+((8)/4))+1"
Output: 3
Explanation: Digit 8 is inside of 3 nested parentheses in the string.
```

Java | Auto

```java
class Solution {
    public int maxDepth(String s) {
        int upcount=0;
        int downcount=0;
        int ans=0;
        for(int i =0; i<s.length(); i++){
            if(s.charAt(i)=='('){
                upcount++;
            }
            ans=Math.max(ans,upcount);
            if(s.charAt(i)==')'){
                upcount--;
            }

        }
        return ans;
```

Ln 18, Col 2

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

s =
"(1+(2*3)+((8)/4))+1"

Output

3

Expected

Console | Run | Submit