GRAPHS... ↑

video-41

"let's make it easy too" →

Hard

Leetcode - 2959

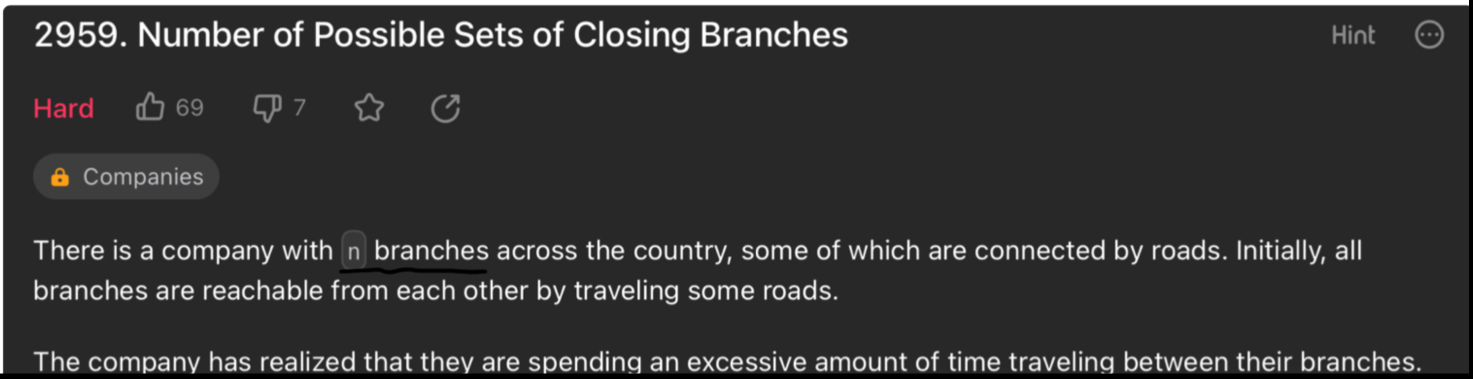
→ Easy

If you have tried my "Graph Concepts & Qns" playlist, these Qns, will seem very easy. Do try it once ;)

Facebook
Instagram → codestorywithMIK

(Twitter) → CSwithMIK

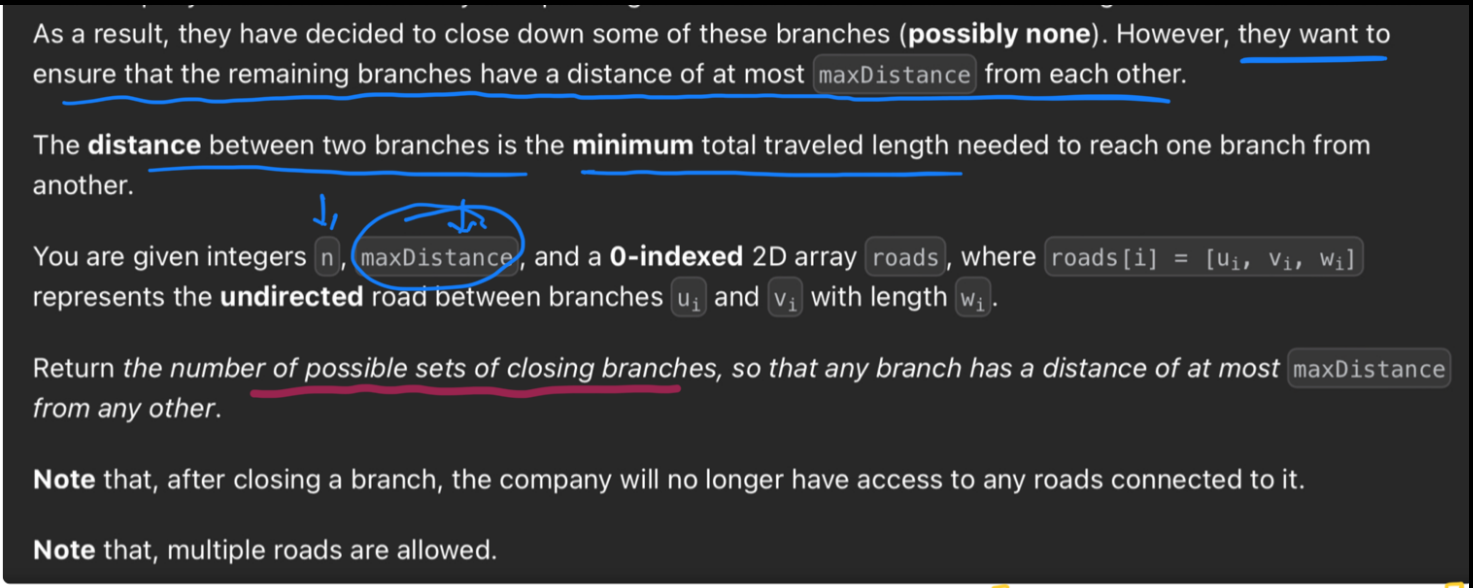
codestorywithMIK →



2959. Number of Possible Sets of Closing Branches

There is a company with `n` branches across the country, some of which are connected by roads. Initially, all branches are reachable from each other by traveling some roads.

The company has realized that they are spending an excessive amount of time traveling between their branches.



As a result, they have decided to close down some of these branches (**possibly none**). However, they want to ensure that the remaining branches have a distance of at most `maxDistance` from each other.

The **distance** between two branches is the **minimum** total traveled length needed to reach one branch from another.

You are given integers `n`, `maxDistance`, and a **0-indexed** 2D array `roads`, where `roads[i] = [u_i, v_i, w_i]` represents the **undirected** road between branches `u_i` and `v_i` with length `w_i`.

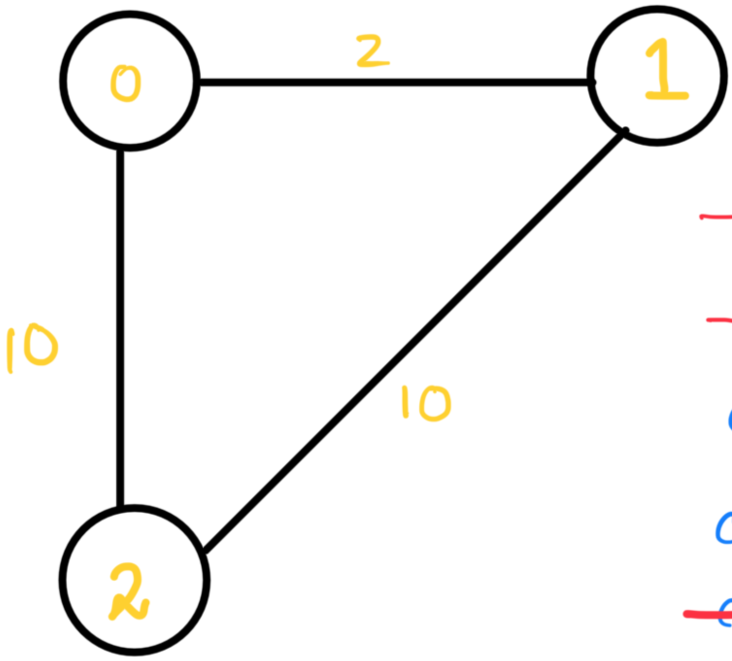
Return *the number of possible sets of closing branches, so that any branch has a distance of at most* `maxDistance` *from any other.*

Note that, after closing a branch, the company will no longer have access to any roads connected to it.

Note that, multiple roads are allowed.

Example:- n = 3 , maxDistance = 5 , roads = [(0,1,2), (1,2,10). (0,2,10)]

Output ⟹ 5



Close{0,1} → {2}

close{0,2} → {1}

Close{1,2} → {0}

~~Close{0} → {1,2}~~

~~Close{1} → {0,2}~~

Close{2} → {0,1}

Close{0,1,2} → { }

~~Close{ } → {0,1,2}~~

Subsets

n = 3

{0, 1, 2} ⟹ 2^n

{ }, {0}, {1}, {2}, {0,1}, {0,2}, {1,2}, {0,1,2}.

What can be the easiest approach that we can come to mind ???

(-) Generate all possible sets.

(-) Try all of them

Let's see this :-

maxOut = 5

⇒ { } ✓, {0} ✓, {1} ✓, {2} ✓, {0,1} ✓, ~~{0,2}~~, ~~{1,2}~~, ~~{0,1,2}~~

⇒ roads = [{0,1,2}, {1,2,10}, {0,2,10}]

Graph.

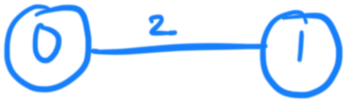
{ } ⇒ ✓

{0} ⇒ 0 ✓

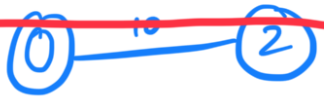
{1} ⇒ 1 ✓

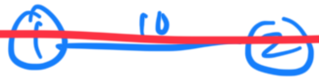
{2} ⇒ 2 ✓

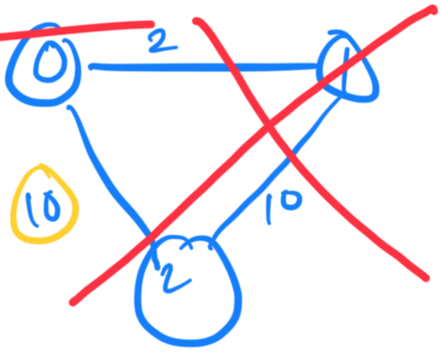
Outp = 5

{0,1} ⇒ 

✓

~~{0,2} ⇒~~ 

~~{1,2} ⇒~~ 

~~{0,1,2} ⇒~~ 

0−1 = 2 < 5

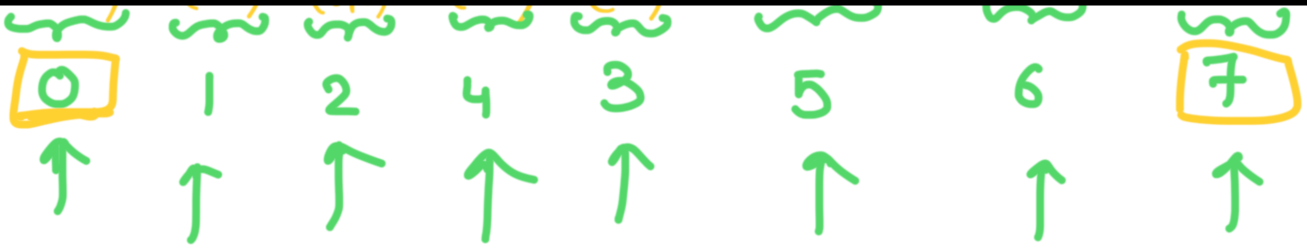
~~0−2 10 → 5~~

1−2

Some tips to improve :-

n = 3 (0, 1, 2)

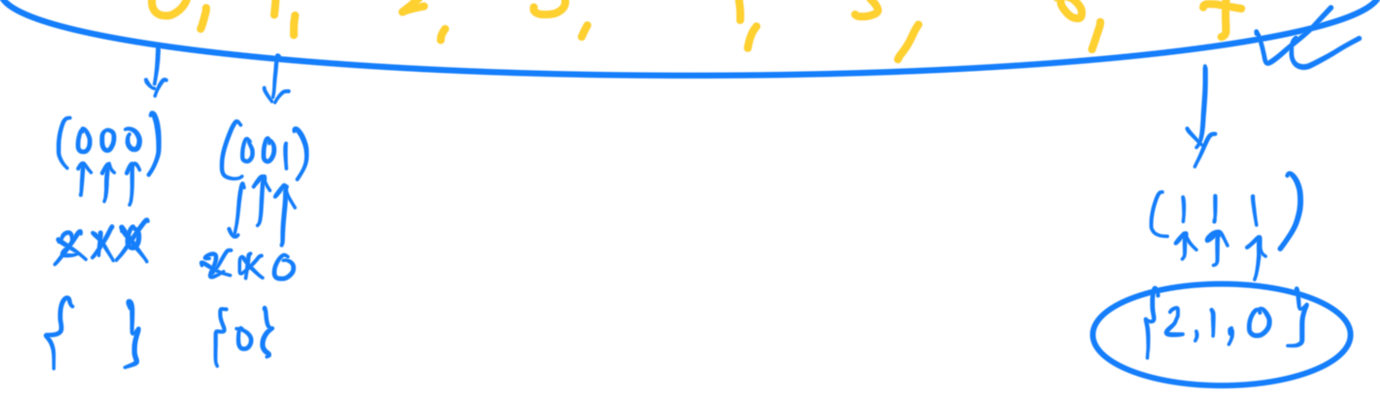
{ },	{0},	{1},	{2},	{0,1},	{0,2},	{1,2},	{0,1,2}
(000)	(001)	(010)	(100)	(011)	(101)	(110)	(111)



$n = 3$

$0 - (2^n - 1)$

0, 1, 2, 3, 4, 5, 6, 7 ✓



```
for( set = 0 ;  set <= 2^n-1  ; set++) {
        //0 -> (0 00)  -> { } .
               ↑ ↑↑
               2 1 0
        //1 -> (001) -> {0}
               ↑↑↑
        ⋮
```

{u, v, wt } ←

{ 0, 1, 2 }

set = 3 ←

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

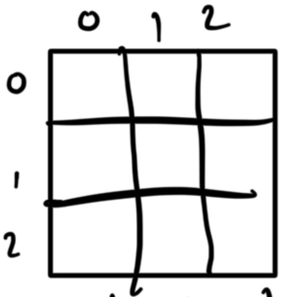
```
  0 0 1
& 0 0 1
-------
  0 0 1
```

//graph formation

```
for (vector<int> road : roads) {
  ← u = road[0];    // 0 (Present)
  ← v = road[1];    // 1 (Present)
    wt = road[2];

    if ((set >> u) &1 == 1
         (set >> v)&1 == 1 ) {

    }
```



Floyd Warshal

//Shortest distance from any
// node to any other node