Problem 2959: Number of Possible Sets of Closing Branches

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

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

.

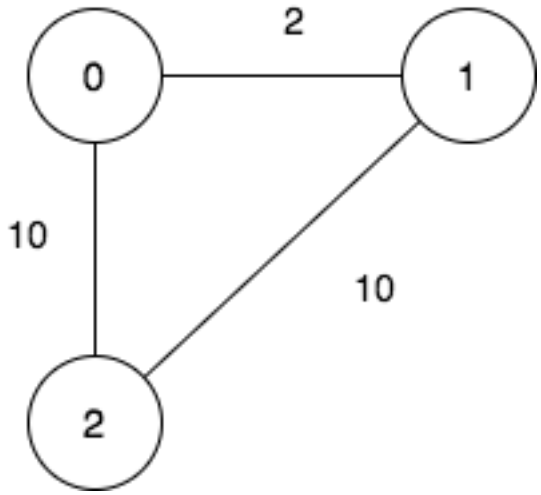
Note

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 1:



Input:

n = 3, maxDistance = 5, roads = [[0,1,2],[1,2,10],[0,2,10]]

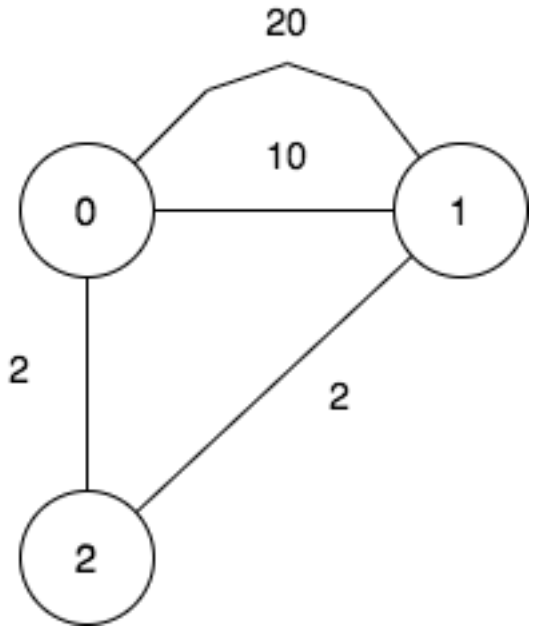
Output:

5

Explanation:

The possible sets of closing branches are: - The set [2], after closing, active branches are [0,1] and they are reachable to each other within distance 2. - The set [0,1], after closing, the active branch is [2]. - The set [1,2], after closing, the active branch is [0]. - The set [0,2], after closing, the active branch is [1]. - The set [0,1,2], after closing, there are no active branches. It can be proven, that there are only 5 possible sets of closing branches.

Example 2:



Input:

n = 3, maxDistance = 5, roads = [[0,1,20],[0,1,10],[1,2,2],[0,2,2]]

Output:

7

Explanation:

The possible sets of closing branches are: - The set [], after closing, active branches are [0,1,2] and they are reachable to each other within distance 4. - The set [0], after closing, active branches are [1,2] and they are reachable to each other within distance 2. - The set [1], after closing, active branches are [0,2] and they are reachable to each other within distance 2. - The set [0,1], after closing, the active branch is [2]. - The set [1,2], after closing, the active branch is [0]. - The set [0,2], after closing, the active branch is [1]. - The set [0,1,2], after closing, there are no active branches. It can be proven, that there are only 7 possible sets of closing branches.

Example 3:

Input:

n = 1, maxDistance = 10, roads = []

Output:

2

Explanation:

The possible sets of closing branches are: - The set [], after closing, the active branch is [0]. - The set [0], after closing, there are no active branches. It can be proven, that there are only 2 possible sets of closing branches.

Constraints:

1 <= n <= 10

1 <= maxDistance <= 10

5

0 <= roads.length <= 1000

roads[i].length == 3

0 <= u

i

, v

i

<= n - 1

u

i

!= v

i

1 <= w

i

<= 1000

All branches are reachable from each other by traveling some roads.

Code Snippets

C++:

```
class Solution {
public:
int numberOfSets(int n, int maxDistance, vector<vector<int>>& roads) {

}
};
```

Java:

```
class Solution {
public int numberOfSets(int n, int maxDistance, int[][] roads) {

}
}
```

Python3:

```
class Solution:
def numberOfSets(self, n: int, maxDistance: int, roads: List[List[int]]) ->
int:
```

Python:

```
class Solution(object):
def numberOfSets(self, n, maxDistance, roads):
"""
:type n: int
:type maxDistance: int
```

```
:type roads: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} maxDistance
* @param {number[][]} roads
* @return {number}
*/
var numberOfSets = function(n, maxDistance, roads) {

};
```

TypeScript:

```
function numberOfSets(n: number, maxDistance: number, roads: number[][]):
number {

};
```

C#:

```
public class Solution {
public int NumberOfSets(int n, int maxDistance, int[][] roads) {

}
}
```

C:

```
int numberOfSets(int n, int maxDistance, int** roads, int roadsSize, int*
roadsColSize) {

}
```

Go:

```
func numberOfSets(n int, maxDistance int, roads [][]int) int {
```

```
}
```

Kotlin:

```
class Solution {
fun numberOfSets(n: Int, maxDistance: Int, roads: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func numberOfSets(_ n: Int, _ maxDistance: Int, _ roads: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_sets(n: i32, max_distance: i32, roads: Vec<Vec<i32>>) -> i32
{

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} max_distance
# @param {Integer[][]} roads
# @return {Integer}
def number_of_sets(n, max_distance, roads)

end
```

PHP:

```
class Solution {

/**
```

```
* @param Integer $n
* @param Integer $maxDistance
* @param Integer[][] $roads
* @return Integer
*/
function numberOfSets($n, $maxDistance, $roads) {

}
}
```

Dart:

```
class Solution {
int numberOfSets(int n, int maxDistance, List<List<int>> roads) {

}
}
```

Scala:

```
object Solution {
def numberOfSets(n: Int, maxDistance: Int, roads: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_sets(n :: integer, max_distance :: integer, roads ::
[[integer]]) :: integer
def number_of_sets(n, max_distance, roads) do

end
end
```

Erlang:

```
-spec number_of_sets(N :: integer(), MaxDistance :: integer(), Roads ::
[[integer()]]) -> integer().
number_of_sets(N, MaxDistance, Roads) ->
.
```

Racket:

```
(define/contract (number-of-sets n maxDistance roads)
(-> exact-integer? exact-integer? (listof (listof exact-integer?))
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Possible Sets of Closing Branches
* Difficulty: Hard
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int numberOfSets(int n, int maxDistance, vector<vector<int>>& roads) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Possible Sets of Closing Branches
* Difficulty: Hard
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
```

```
public int numberOfSets(int n, int maxDistance, int[][] roads) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Possible Sets of Closing Branches
Difficulty: Hard
Tags: array, graph, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def numberOfSets(self, n: int, maxDistance: int, roads: List[List[int]]) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfSets(self, n, maxDistance, roads):
"""
:type n: int
:type maxDistance: int
:type roads: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Possible Sets of Closing Branches
* Difficulty: Hard
* Tags: array, graph, queue, heap
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number} maxDistance
* @param {number[][]} roads
* @return {number}
*/
var numberOfSets = function(n, maxDistance, roads) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Possible Sets of Closing Branches
* Difficulty: Hard
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function numberOfSets(n: number, maxDistance: number, roads: number[][]):
number {

};
```

C# Solution:

```
/*
* Problem: Number of Possible Sets of Closing Branches
* Difficulty: Hard
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int NumberOfSets(int n, int maxDistance, int[][] roads) {

}
}
```

C Solution:

```
/*
* Problem: Number of Possible Sets of Closing Branches
* Difficulty: Hard
* Tags: array, graph, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int numberOfSets(int n, int maxDistance, int** roads, int roadsSize, int*
roadsColSize) {

}
```

Go Solution:

```
// Problem: Number of Possible Sets of Closing Branches
// Difficulty: Hard
// Tags: array, graph, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func numberOfSets(n int, maxDistance int, roads [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfSets(n: Int, maxDistance: Int, roads: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfSets(_ n: Int, _ maxDistance: Int, _ roads: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Possible Sets of Closing Branches
// Difficulty: Hard
// Tags: array, graph, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn number_of_sets(n: i32, max_distance: i32, roads: Vec<Vec<i32>>) -> i32
{

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} max_distance
# @param {Integer[][]} roads
# @return {Integer}
def number_of_sets(n, max_distance, roads)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $maxDistance
* @param Integer[][] $roads
* @return Integer
*/
function numberOfSets($n, $maxDistance, $roads) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfSets(int n, int maxDistance, List<List<int>> roads) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfSets(n: Int, maxDistance: Int, roads: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_sets(n :: integer, max_distance :: integer, roads ::
[[integer]]) :: integer
def number_of_sets(n, max_distance, roads) do

end
end
```

Erlang Solution:

```
-spec number_of_sets(N :: integer(), MaxDistance :: integer(), Roads ::
[[integer()]]) -> integer().
number_of_sets(N, MaxDistance, Roads) ->
.
```

Racket Solution:

```
(define/contract (number-of-sets n maxDistance roads)
(-> exact-integer? exact-integer? (listof (listof exact-integer?))
exact-integer?)
)
```