Problem 2867: Count Valid Paths in a Tree

Problem Information

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

Problem Description

There is an undirected tree with

n

nodes labeled from

1

to

n

. You are given the integer

n

and a 2D integer array

edges

of length

n - 1

, where

edges[i] = [u

i

, v

i

]

indicates that there is an edge between nodes

u

i

and

v

i

in the tree.

Return

the

number of valid paths

in the tree

.

A path

(a, b)

is

valid

if there exists

exactly one

prime number among the node labels in the path from

a

to

b

.

Note

that:

The path

(a, b)

is a sequence of

distinct

nodes starting with node

a

and ending with node

b

such that every two adjacent nodes in the sequence share an edge in the tree.

Path

(a, b)

and path

(b, a)

are considered the

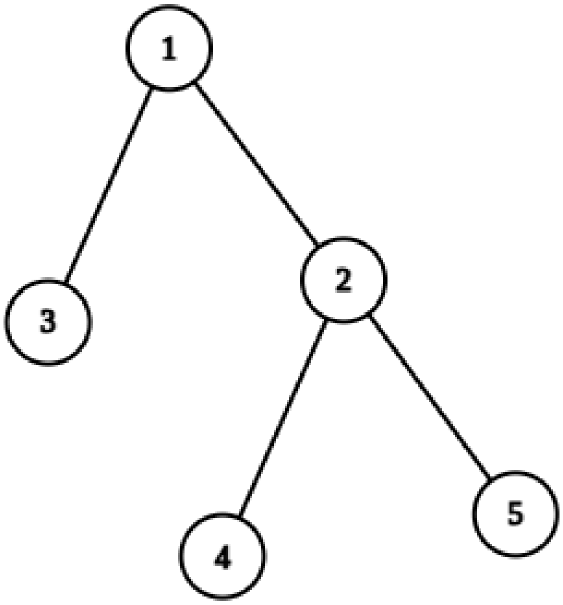
same

and counted only

once

.

Example 1:



Input:

n = 5, edges = [[1,2],[1,3],[2,4],[2,5]]

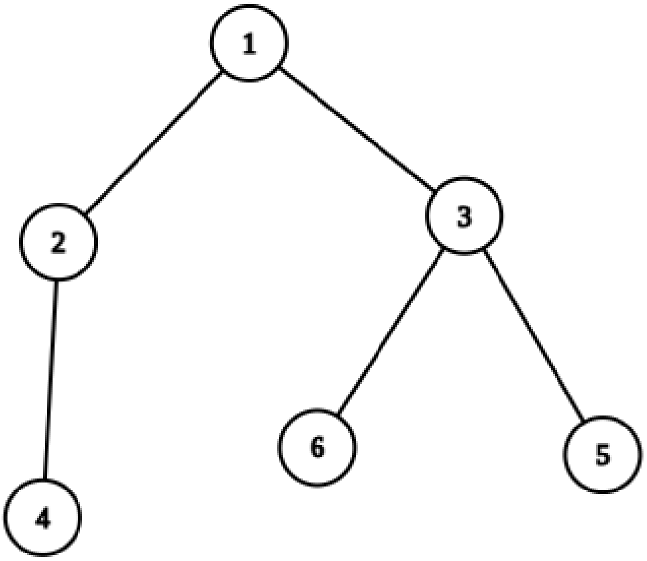
Output:

4

Explanation:

The pairs with exactly one prime number on the path between them are: - (1, 2) since the path from 1 to 2 contains prime number 2. - (1, 3) since the path from 1 to 3 contains prime number 3. - (1, 4) since the path from 1 to 4 contains prime number 2. - (2, 4) since the path from 2 to 4 contains prime number 2. It can be shown that there are only 4 valid paths.

Example 2:



Input:

n = 6, edges = [[1,2],[1,3],[2,4],[3,5],[3,6]]

Output:

6

Explanation:

The pairs with exactly one prime number on the path between them are: - (1, 2) since the path from 1 to 2 contains prime number 2. - (1, 3) since the path from 1 to 3 contains prime number 3. - (1, 4) since the path from 1 to 4 contains prime number 2. - (1, 6) since the path from 1 to 6 contains prime number 3. - (2, 4) since the path from 2 to 4 contains prime number 2. - (3, 6) since the path from 3 to 6 contains prime number 3. It can be shown that there are only 6 valid paths.

Constraints:

1 <= n <= 10

5

edges.length == n - 1

edges[i].length == 2

1 <= u

i

, v

i

<= n

The input is generated such that

edges

represent a valid tree.

Code Snippets

C++:

```
class Solution {
public:
long long countPaths(int n, vector<vector<int>>& edges) {

}
};
```

Java:

```
class Solution {
public long countPaths(int n, int[][] edges) {

}
}
```

Python3:

```
class Solution:
def countPaths(self, n: int, edges: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def countPaths(self, n, edges):
"""
:type n: int
:type edges: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number[][]} edges
* @return {number}
*/
var countPaths = function(n, edges) {

};
```

TypeScript:

```
function countPaths(n: number, edges: number[][]): number {

};
```

C#:

```
public class Solution {
public long CountPaths(int n, int[][] edges) {

}
}
```

C:

```
long long countPaths(int n, int** edges, int edgesSize, int* edgesColSize) {

}
```

Go:

```
func countPaths(n int, edges [][]int) int64 {

}
```

Kotlin:

```
class Solution {
fun countPaths(n: Int, edges: Array<IntArray>): Long {

}
}
```

Swift:

```
class Solution {
func countPaths(_ n: Int, _ edges: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_paths(n: i32, edges: Vec<Vec<i32>>) -> i64 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer[][]} edges
# @return {Integer}
def count_paths(n, edges)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer[][] $edges
* @return Integer
*/
function countPaths($n, $edges) {

}
}
```

Dart:

```
class Solution {
int countPaths(int n, List<List<int>> edges) {

}
}
```

Scala:

```
object Solution {
def countPaths(n: Int, edges: Array[Array[Int]]): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_paths(n :: integer, edges :: [[integer]]) :: integer
def count_paths(n, edges) do

end
end
```

Erlang:

```
-spec count_paths(N :: integer(), Edges :: [[integer()]]) -> integer().
count_paths(N, Edges) ->
.
```

Racket:

```
(define/contract (count-paths n edges)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Valid Paths in a Tree
* Difficulty: Hard
* Tags: array, tree, dp, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
long long countPaths(int n, vector<vector<int>>& edges) {
}
};
```

Java Solution:

```
/**
* Problem: Count Valid Paths in a Tree
* Difficulty: Hard
* Tags: array, tree, dp, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public long countPaths(int n, int[][] edges) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Valid Paths in a Tree
Difficulty: Hard
Tags: array, tree, dp, math, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

```
class Solution(object):
def countPaths(self, n, edges):
"""
:type n: int
:type edges: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Valid Paths in a Tree
* Difficulty: Hard
* Tags: array, tree, dp, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} n
* @param {number[][]} edges
* @return {number}
*/
var countPaths = function(n, edges) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Valid Paths in a Tree
* Difficulty: Hard
* Tags: array, tree, dp, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function countPaths(n: number, edges: number[][]): number {
};
```

C# Solution:

```
/*
* Problem: Count Valid Paths in a Tree
* Difficulty: Hard
* Tags: array, tree, dp, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public long CountPaths(int n, int[][] edges) {

}
}
```

C Solution:

```
/*
* Problem: Count Valid Paths in a Tree
* Difficulty: Hard
* Tags: array, tree, dp, math, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

long long countPaths(int n, int** edges, int edgesSize, int* edgesColSize) {

}
```

Go Solution:

```
// Problem: Count Valid Paths in a Tree
// Difficulty: Hard
// Tags: array, tree, dp, math, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func countPaths(n int, edges [][]int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun countPaths(n: Int, edges: Array<IntArray>): Long {

}
}
```

Swift Solution:

```
class Solution {
func countPaths(_ n: Int, _ edges: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Valid Paths in a Tree
// Difficulty: Hard
// Tags: array, tree, dp, math, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn count_paths(n: i32, edges: Vec<Vec<i32>>) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer[][]} edges
# @return {Integer}
def count_paths(n, edges)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer[][] $edges
* @return Integer
*/
function countPaths($n, $edges) {

}
}
```

Dart Solution:

```
class Solution {
int countPaths(int n, List<List<int>> edges) {

}
}
```

Scala Solution:

```
object Solution {
def countPaths(n: Int, edges: Array[Array[Int]]): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_paths(n :: integer, edges :: [[integer]]) :: integer
def count_paths(n, edges) do

end
end
```

Erlang Solution:

```
-spec count_paths(N :: integer(), Edges :: [[integer()]]) -> integer().
count_paths(N, Edges) ->
.
```

Racket Solution:

```
(define/contract (count-paths n edges)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```