

Problem 3558: Number of Ways to Assign Edge Weights I

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

There is an undirected tree with

n

nodes labeled from 1 to

n

, rooted at node 1. The tree is represented by a 2D integer array

edges

of length

$n - 1$

, where

$\text{edges}[i] = [u$

i

$, v$

i

]

indicates that there is an edge between nodes

u

i

and

v

i

Initially, all edges have a weight of 0. You must assign each edge a weight of either

1

or

2

The

cost

of a path between any two nodes

u

and

v

is the total weight of all edges in the path connecting them.

Select any one node

x

at the

maximum

depth. Return the number of ways to assign edge weights in the path from node 1 to

x

such that its total cost is

odd

Since the answer may be large, return it

modulo

10

9

+ 7

Note:

Ignore all edges

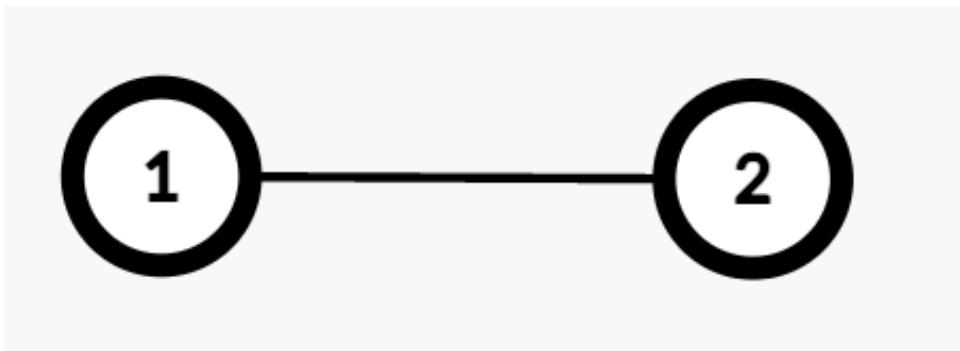
not

in the path from node 1 to

x

.

Example 1:



Input:

```
edges = [[1,2]]
```

Output:

1

Explanation:

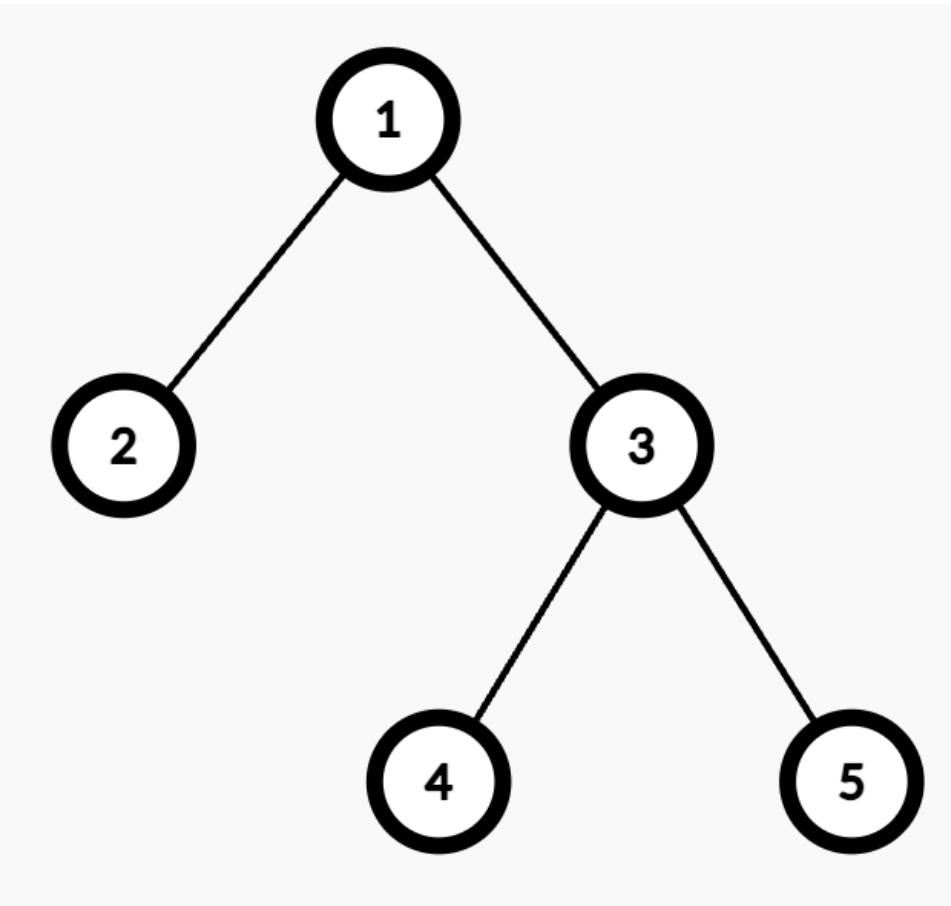
The path from Node 1 to Node 2 consists of one edge (

$1 \rightarrow 2$

).

Assigning weight 1 makes the cost odd, while 2 makes it even. Thus, the number of valid assignments is 1.

Example 2:



Input:

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

Output:

2

Explanation:

The maximum depth is 2, with nodes 4 and 5 at the same depth. Either node can be selected for processing.

For example, the path from Node 1 to Node 4 consists of two edges (

$1 \rightarrow 3$

and

$3 \rightarrow 4$

).

Assigning weights (1,2) or (2,1) results in an odd cost. Thus, the number of valid assignments is 2.

Constraints:

$2 \leq n \leq 10$

5

`edges.length == n - 1`

`edges[i] == [u`

`i`

`, v`

`i`

`]`

$1 \leq u$

`i`

`, v`

`i`

$\leq n$

`edges`

represents a valid tree.

Code Snippets

C++:

```
class Solution {
public:
    int assignEdgeWeights(vector<vector<int>>& edges) {
        }
};
```

Java:

```
class Solution {
    public int assignEdgeWeights(int[][] edges) {
        }
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
 * @param {number[][]} edges
 * @return {number}
 */
var assignEdgeWeights = function(edges) {
    };
```

TypeScript:

```
function assignEdgeWeights(edges: number[][]): number {  
}  
};
```

C#:

```
public class Solution {  
    public int AssignEdgeWeights(int[][] edges) {  
        }  
    }  
}
```

C:

```
int assignEdgeWeights(int** edges, int edgesSize, int* edgesColSize) {  
  
}
```

Go:

```
func assignEdgeWeights(edges [][]int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun assignEdgeWeights(edges: Array<IntArray>): Int {  
        }  
    }  
}
```

Swift:

```
class Solution {  
    func assignEdgeWeights(_ edges: [[Int]]) -> Int {  
        }  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn assign_edge_weights(edges: Vec<Vec<i32>>) -> i32 {  
  
    }  
}
```

Ruby:

```
# @param {Integer[][][]} edges  
# @return {Integer}  
def assign_edge_weights(edges)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[][][] $edges  
     * @return Integer  
     */  
    function assignEdgeWeights($edges) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int assignEdgeWeights(List<List<int>> edges) {  
  
    }  
}
```

Scala:

```
object Solution {  
    def assignEdgeWeights(edges: Array[Array[Int]]): Int = {  
  
    }
```

```
}
```

Elixir:

```
defmodule Solution do
  @spec assign_edge_weights(edges :: [[integer]]) :: integer
  def assign_edge_weights(edges) do
    end
  end
```

Erlang:

```
-spec assign_edge_weights(Edges :: [[integer()]]) -> integer().
assign_edge_weights(Edges) ->
  .
```

Racket:

```
(define/contract (assign-edge-weights edges)
  (-> (listof (listof exact-integer?)) exact-integer?))
```

Solutions

C++ Solution:

```
/*
 * Problem: Number of Ways to Assign Edge Weights I
 * Difficulty: Medium
 * Tags: array, tree, math, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

class Solution {
public:
  int assignEdgeWeights(vector<vector<int>>& edges) {
```

```
}
```

```
} ;
```

Java Solution:

```
/**  
 * Problem: Number of Ways to Assign Edge Weights I  
 * Difficulty: Medium  
 * Tags: array, tree, math, search  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(h) for recursion stack where h is height  
 */  
  
class Solution {  
    public int assignEdgeWeights(int[][][] edges) {  
        return 0;  
    }  
}
```

Python3 Solution:

```
"""  
Problem: Number of Ways to Assign Edge Weights I  
Difficulty: Medium  
Tags: array, tree, math, search  
  
Approach: Use two pointers or sliding window technique  
Time Complexity: O(n) or O(n log n)  
Space Complexity: O(h) for recursion stack where h is height  
"""  
  
class Solution:  
    def assignEdgeWeights(self, edges: List[List[int]]) -> int:  
        # TODO: Implement optimized solution  
        pass
```

Python Solution:

```

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

```

JavaScript Solution:

```

/**
 * Problem: Number of Ways to Assign Edge Weights I
 * Difficulty: Medium
 * Tags: array, tree, math, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

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

};

```

TypeScript Solution:

```

/**
 * Problem: Number of Ways to Assign Edge Weights I
 * Difficulty: Medium
 * Tags: array, tree, math, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

function assignEdgeWeights(edges: number[][]): number {

};

```

C# Solution:

```
/*
 * Problem: Number of Ways to Assign Edge Weights I
 * Difficulty: Medium
 * Tags: array, tree, math, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

public class Solution {
    public int AssignEdgeWeights(int[][][] edges) {
        return 0;
    }
}
```

C Solution:

```
/*
 * Problem: Number of Ways to Assign Edge Weights I
 * Difficulty: Medium
 * Tags: array, tree, math, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

int assignEdgeWeights(int** edges, int edgesSize, int* edgesColSize) {
    return 0;
}
```

Go Solution:

```
// Problem: Number of Ways to Assign Edge Weights I
// Difficulty: Medium
// Tags: array, tree, math, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
```

```

// Space Complexity: O(h) for recursion stack where h is height

func assignEdgeWeights(edges [][]int) int {
}

```

Kotlin Solution:

```

class Solution {
    fun assignEdgeWeights(edges: Array<IntArray>): Int {
        return 0
    }
}

```

Swift Solution:

```

class Solution {
    func assignEdgeWeights(_ edges: [[Int]]) -> Int {
        return 0
    }
}

```

Rust Solution:

```

// Problem: Number of Ways to Assign Edge Weights I
// Difficulty: Medium
// Tags: array, tree, math, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
    pub fn assign_edge_weights(edges: Vec<Vec<i32>>) -> i32 {
        return 0
    }
}

```

Ruby Solution:

```
# @param {Integer[][]} edges
# @return {Integer}
def assign_edge_weights(edges)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[][] $edges
     * @return Integer
     */
    function assignEdgeWeights($edges) {

    }
}
```

Dart Solution:

```
class Solution {
int assignEdgeWeights(List<List<int>> edges) {
}
```

Scala Solution:

```
object Solution {
def assignEdgeWeights(edges: Array[Array[Int]]): Int = {
}
```

Elixir Solution:

```
defmodule Solution do
@spec assign_edge_weights(edges :: [[integer]]) :: integer
def assign_edge_weights(edges) do
end
```

```
end
```

Erlang Solution:

```
-spec assign_edge_weights(Edges :: [[integer()]]) -> integer().  
assign_edge_weights(Edges) ->  
.
```

Racket Solution:

```
(define/contract (assign-edge-weights edges)  
(-> (listof (listof exact-integer?)) exact-integer?)  
)
```