Problem 2445: Number of Nodes With Value One

Problem Information

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

Problem Description

There is an

undirected

connected tree with

n

nodes labeled from

1

to

n

and

n - 1

edges. You are given the integer

n

. The parent node of a node with a label

v

is the node with the label

floor (v / 2)

. The root of the tree is the node with the label

1

.

For example, if

n = 7

, then the node with the label

3

has the node with the label

floor(3 / 2) = 1

as its parent, and the node with the label

7

has the node with the label

floor(7 / 2) = 3

as its parent.

You are also given an integer array

queries

. Initially, every node has a value

0

on it. For each query

queries[i]

, you should flip all values in the subtree of the node with the label

queries[i]

.

Return

the total number of nodes with the value

1

after processing all the queries

.

Note

that:

Flipping the value of a node means that the node with the value

0

becomes

1

and vice versa.

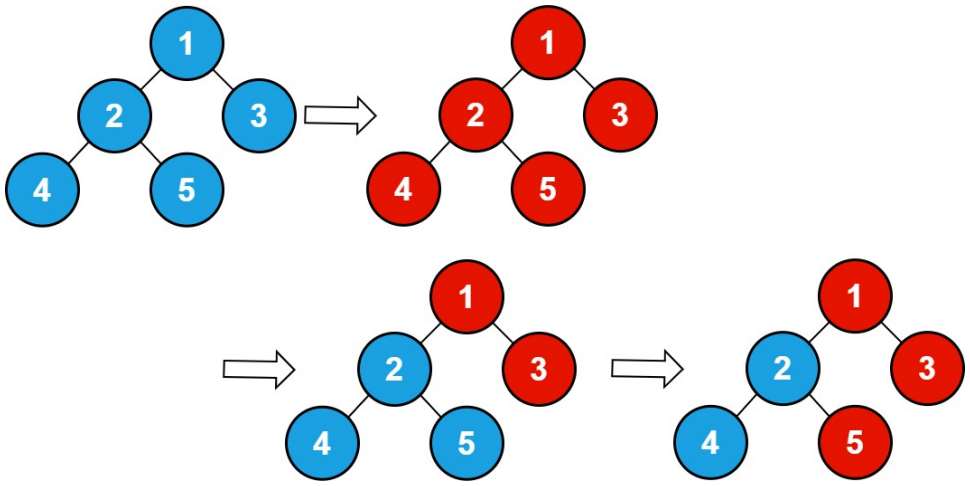
floor(x)

is equivalent to rounding

x

down to the nearest integer.

Example 1:



Input:

n = 5 , queries = [1,2,5]

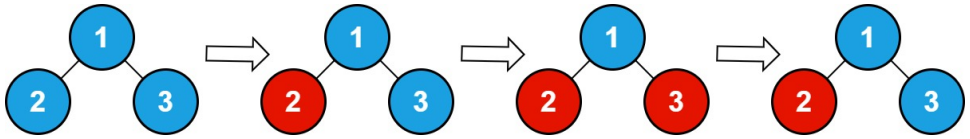
Output:

3

Explanation:

The diagram above shows the tree structure and its status after performing the queries. The blue node represents the value 0, and the red node represents the value 1. After processing the queries, there are three red nodes (nodes with value 1): 1, 3, and 5.

Example 2:



Input:

n = 3, queries = [2,3,3]

Output:

1

Explanation:

The diagram above shows the tree structure and its status after performing the queries. The blue node represents the value 0, and the red node represents the value 1. After processing the queries, there are one red node (node with value 1): 2.

Constraints:

1 <= n <= 10

5

1 <= queries.length <= 10

5

1 <= queries[i] <= n

Code Snippets

C++:

```
class Solution {
public:
int numberOfNodes(int n, vector<int>& queries) {

}
};
```

Java:

```
class Solution {
public int numberOfNodes(int n, int[] queries) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function numberOfNodes(n: number, queries: number[]): number {

};
```

C#:

```
public class Solution {
public int NumberOfNodes(int n, int[] queries) {
```

```
}
}
```

C:

```
int numberOfNodes(int n, int* queries, int queriesSize) {

}
```

Go:

```
func numberOfNodes(n int, queries []int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfNodes(n: Int, queries: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn number_of_nodes(n: i32, queries: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer[]} queries
# @return {Integer}
def number_of_nodes(n, queries)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer[] $queries
* @return Integer
*/
function numberOfNodes($n, $queries) {

}
}
```

Dart:

```
class Solution {
int numberOfNodes(int n, List<int> queries) {

}
}
```

Scala:

```
object Solution {
def numberOfNodes(n: Int, queries: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_nodes(n :: integer, queries :: [integer]) :: integer
def number_of_nodes(n, queries) do
```

```
end
end
```

Erlang:

```
-spec number_of_nodes(N :: integer(), Queries :: [integer()]) -> integer().
number_of_nodes(N, Queries) ->
.
```

Racket:

```
(define/contract (number-of-nodes n queries)
(-> exact-integer? (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Nodes With Value One
* Difficulty: Medium
* Tags: array, tree, 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 numberOfNodes(int n, vector<int>& queries) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Nodes With Value One
```

```
* Difficulty: Medium
* Tags: array, tree, 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 numberOfNodes(int n, int[] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Nodes With Value One
Difficulty: Medium
Tags: array, tree, 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 numberOfNodes(self, n: int, queries: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Nodes With Value One
* Difficulty: Medium
* Tags: array, tree, 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} n
* @param {number[]} queries
* @return {number}
*/
var numberOfNodes = function(n, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Nodes With Value One
* Difficulty: Medium
* Tags: array, tree, 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 numberOfNodes(n: number, queries: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Nodes With Value One
* Difficulty: Medium
* Tags: array, tree, 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 NumberOfNodes(int n, int[] queries) {

}
}
```

C Solution:

```
/*
* Problem: Number of Nodes With Value One
* Difficulty: Medium
* Tags: array, tree, 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 numberOfNodes(int n, int* queries, int queriesSize) {

}
```

Go Solution:

```
// Problem: Number of Nodes With Value One
// Difficulty: Medium
// Tags: array, tree, 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 numberOfNodes(n int, queries []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfNodes(n: Int, queries: IntArray): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Nodes With Value One
// Difficulty: Medium
// Tags: array, tree, 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 number_of_nodes(n: i32, queries: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer[]} queries
# @return {Integer}
def number_of_nodes(n, queries)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer[] $queries
* @return Integer
*/
function numberOfNodes($n, $queries) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfNodes(int n, List<int> queries) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfNodes(n: Int, queries: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_nodes(n :: integer, queries :: [integer]) :: integer
def number_of_nodes(n, queries) do

end
end
```

Erlang Solution:

```
-spec number_of_nodes(N :: integer(), Queries :: [integer()]) -> integer().
number_of_nodes(N, Queries) ->
.
```

Racket Solution:

```
(define/contract (number-of-nodes n queries)
(-> exact-integer? (listof exact-integer?) exact-integer?)
)
```