Problem 1519: Number of Nodes in the Sub-Tree With the Same Label

Problem Information

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

Problem Description

You are given a tree (i.e. a connected, undirected graph that has no cycles) consisting of

n

nodes numbered from

0

to

n - 1

and exactly

n - 1

edges

. The

root

of the tree is the node

0

, and each node of the tree has

a label

which is a lower-case character given in the string

labels

(i.e. The node with the number

i

has the label

labels[i]

).

The

edges

array is given on the form

edges[i] = [a

i

, b

i

]

, which means there is an edge between nodes

a

i

and

b

i

in the tree.

Return

an array of size

n

where

ans[i]

is the number of nodes in the subtree of the

i

th

node which have the same label as node

i

.

A subtree of a tree

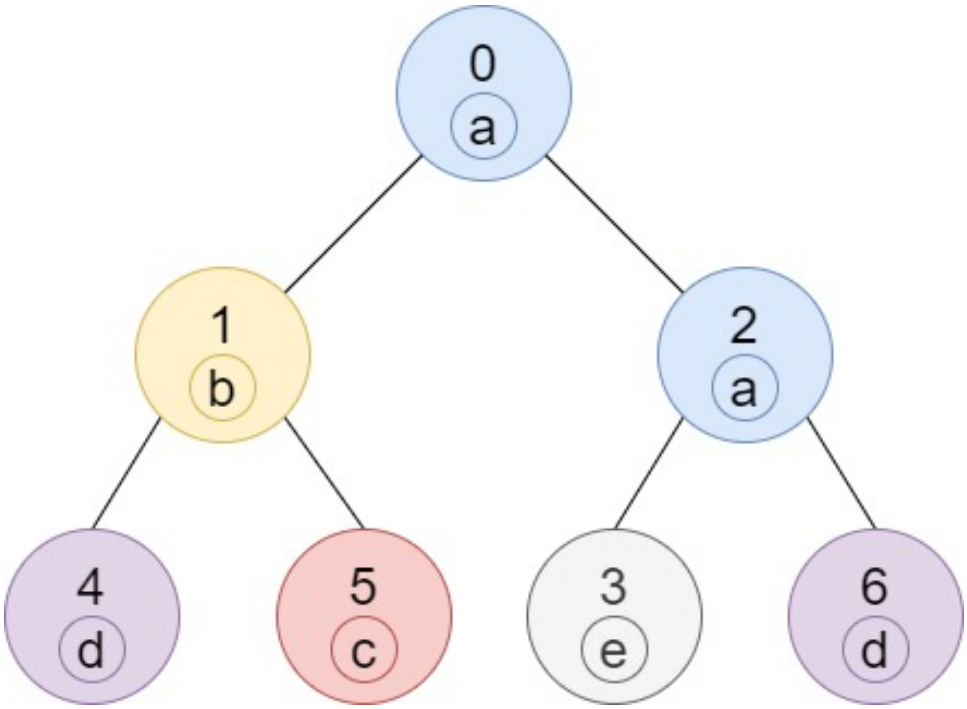
T

is the tree consisting of a node in

T

and all of its descendant nodes.

Example 1:



Input:

n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], labels = "abaedcd"

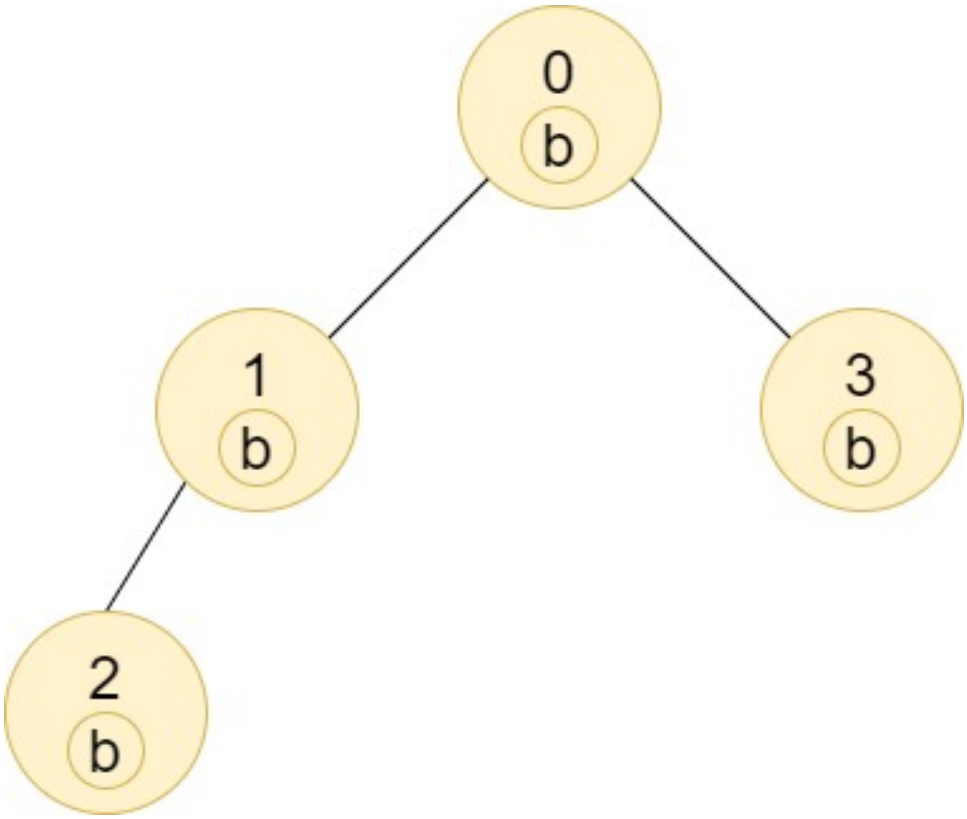
Output:

[2,1,1,1,1,1,1]

Explanation:

Node 0 has label 'a' and its sub-tree has node 2 with label 'a' as well, thus the answer is 2. Notice that any node is part of its sub-tree. Node 1 has a label 'b'. The sub-tree of node 1 contains nodes 1,4 and 5, as nodes 4 and 5 have different labels than node 1, the answer is just 1 (the node itself).

Example 2:



Input:

n = 4, edges = [[0,1],[1,2],[0,3]], labels = "bbbb"

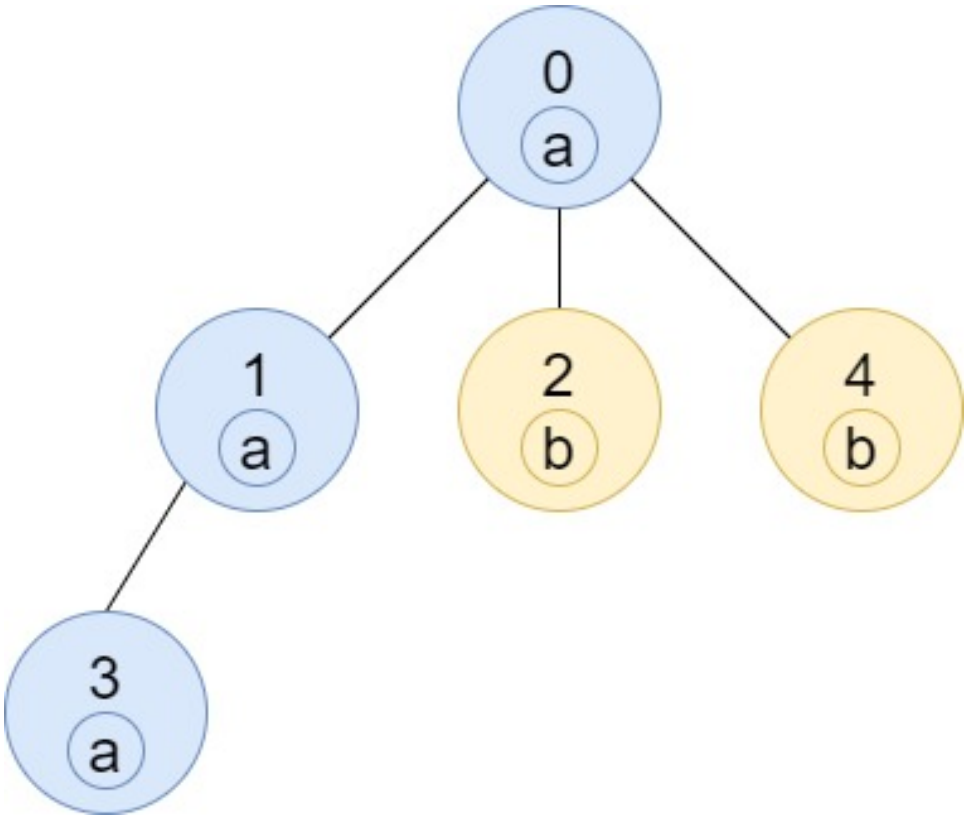
Output:

[4,2,1,1]

Explanation:

The sub-tree of node 2 contains only node 2, so the answer is 1. The sub-tree of node 3 contains only node 3, so the answer is 1. The sub-tree of node 1 contains nodes 1 and 2, both have label 'b', thus the answer is 2. The sub-tree of node 0 contains nodes 0, 1, 2 and 3, all with label 'b', thus the answer is 4.

Example 3:



Input:

n = 5, edges = [[0,1],[0,2],[1,3],[0,4]], labels = "aabab"

Output:

[3,2,1,1,1]

Constraints:

1 <= n <= 10

5

edges.length == n - 1

edges[i].length == 2

0 <= a

i

, b

i

< n

a

i

!= b

i

labels.length == n

labels

is consisting of only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
vector<int> countSubTrees(int n, vector<vector<int>>& edges, string labels) {

}
};
```

Java:

```
class Solution {
public int[] countSubTrees(int n, int[][] edges, String labels) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countSubTrees(n: number, edges: number[][], labels: string):
number[] {

};
```

C#:

```
public class Solution {
public int[] CountSubTrees(int n, int[][] edges, string labels) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* countSubTrees(int n, int** edges, int edgesSize, int* edgesColSize,
char* labels, int* returnSize) {

}
```

Go:

```
func countSubTrees(n int, edges [][]int, labels string) []int {

}
```

Kotlin:

```
class Solution {
fun countSubTrees(n: Int, edges: Array<IntArray>, labels: String): IntArray {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn count_sub_trees(n: i32, edges: Vec<Vec<i32>>, labels: String) ->
Vec<i32> {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
List<int> countSubTrees(int n, List<List<int>> edges, String labels) {

}
}
```

Scala:

```
object Solution {
def countSubTrees(n: Int, edges: Array[Array[Int]], labels: String):
Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_sub_trees(n :: integer, edges :: [[integer]], labels :: String.t)
:: [integer]
def count_sub_trees(n, edges, labels) do

end
end
```

Erlang:

```
-spec count_sub_trees(N :: integer(), Edges :: [[integer()]], Labels ::
unicode:unicode_binary()) -> [integer()].
count_sub_trees(N, Edges, Labels) ->
.
```

Racket:

```
(define/contract (count-sub-trees n edges labels)
(-> exact-integer? (listof (listof exact-integer?)) string? (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Nodes in the Sub-Tree With the Same Label
* Difficulty: Medium
* Tags: array, string, tree, graph, hash, 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:
vector<int> countSubTrees(int n, vector<vector<int>>& edges, string labels) {

}
```

```
};
```

Java Solution:

```
/**
* Problem: Number of Nodes in the Sub-Tree With the Same Label
* Difficulty: Medium
* Tags: array, string, tree, graph, hash, 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[] countSubTrees(int n, int[][] edges, String labels) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Nodes in the Sub-Tree With the Same Label
Difficulty: Medium
Tags: array, string, tree, graph, hash, 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 countSubTrees(self, n: int, edges: List[List[int]], labels: str) ->
List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Nodes in the Sub-Tree With the Same Label
* Difficulty: Medium
* Tags: array, string, tree, graph, hash, 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[][]} edges
* @param {string} labels
* @return {number[]}
*/
var countSubTrees = function(n, edges, labels) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Nodes in the Sub-Tree With the Same Label
* Difficulty: Medium
* Tags: array, string, tree, graph, hash, 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 countSubTrees(n: number, edges: number[][], labels: string):
number[] {

};
```

C# Solution:

```
/*
* Problem: Number of Nodes in the Sub-Tree With the Same Label
* Difficulty: Medium
* Tags: array, string, tree, graph, hash, 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[] CountSubTrees(int n, int[][] edges, string labels) {

}
}
```

C Solution:

```
/*
* Problem: Number of Nodes in the Sub-Tree With the Same Label
* Difficulty: Medium
* Tags: array, string, tree, graph, hash, 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* countSubTrees(int n, int** edges, int edgesSize, int* edgesColSize,
char* labels, int* returnSize) {
```

```
}
```

Go Solution:

```
// Problem: Number of Nodes in the Sub-Tree With the Same Label
// Difficulty: Medium
// Tags: array, string, tree, graph, hash, 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 countSubTrees(n int, edges [][]int, labels string) []int {

}
```

Kotlin Solution:

```
class Solution {
fun countSubTrees(n: Int, edges: Array<IntArray>, labels: String): IntArray {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Nodes in the Sub-Tree With the Same Label
// Difficulty: Medium
// Tags: array, string, tree, graph, hash, 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 count_sub_trees(n: i32, edges: Vec<Vec<i32>>, labels: String) ->
Vec<i32> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
List<int> countSubTrees(int n, List<List<int>> edges, String labels) {

}
}
```

Scala Solution:

```
object Solution {
def countSubTrees(n: Int, edges: Array[Array[Int]], labels: String):
Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_sub_trees(n :: integer, edges :: [[integer]], labels :: String.t)
:: [integer]
def count_sub_trees(n, edges, labels) do

end
end
```

Erlang Solution:

```
-spec count_sub_trees(N :: integer(), Edges :: [[integer()]], Labels ::
unicode:unicode_binary()) -> [integer()].
count_sub_trees(N, Edges, Labels) ->
.
```

Racket Solution:

```
(define/contract (count-sub-trees n edges labels)
(-> exact-integer? (listof (listof exact-integer?)) string? (listof
exact-integer?))
)
```