Problem 1857: Largest Color Value in a Directed Graph

Problem Information

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

Problem Description

There is a

directed graph

of

n

colored nodes and

m

edges. The nodes are numbered from

0

to

n - 1

.

You are given a string

colors

where

colors[i]

is a lowercase English letter representing the

color

of the

i

th

node in this graph (

0-indexed

). You are also given a 2D array

edges

where

edges[j] = [a

j

, b

j

]

indicates that there is a

directed edge

from node

a

j

to node

b

j

.

A valid

path

in the graph is a sequence of nodes

x

1

-> x

2

-> x

3

-> ... -> x

k

such that there is a directed edge from

x

i

to

x

i+1

for every

1 <= i < k

. The

color value

of the path is the number of nodes that are colored the

most frequently

occurring color along that path.

Return

the

largest color value

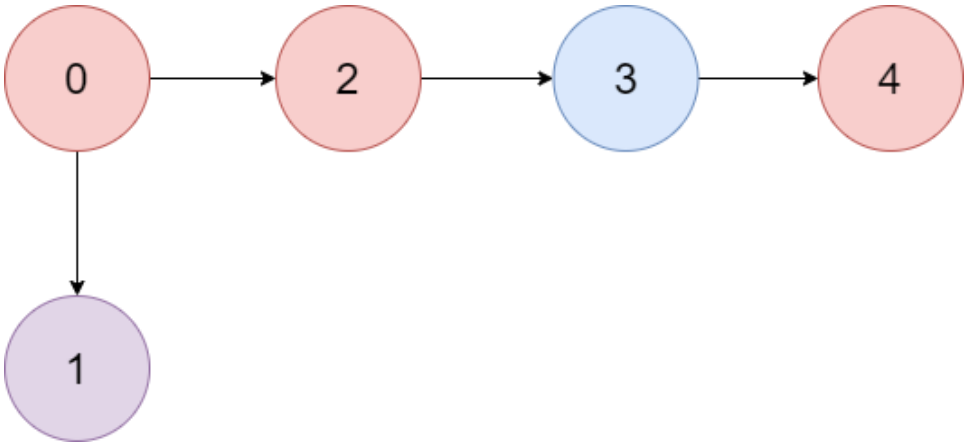
of any valid path in the given graph, or

-1

if the graph contains a cycle

.

Example 1:



Input:

colors = "abaca", edges = [[0,1],[0,2],[2,3],[3,4]]

Output:

3

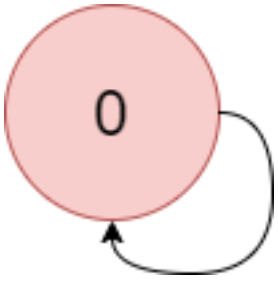
Explanation:

The path 0 -> 2 -> 3 -> 4 contains 3 nodes that are colored

"a" (red in the above image)

.

Example 2:



Input:

colors = "a", edges = [[0,0]]

Output:

-1

Explanation:

There is a cycle from 0 to 0.

Constraints:

n == colors.length

m == edges.length

1 <= n <= 10

5

0 <= m <= 10

5

colors

consists of lowercase English letters.

0 <= a

j

, b

j

< n

Code Snippets

C++:

```
class Solution {
public:
int largestPathValue(string colors, vector<vector<int>>& edges) {

}
};
```

Java:

```
class Solution {
public int largestPathValue(String colors, int[][] edges) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {string} colors
* @param {number[][]} edges
* @return {number}
*/
var largestPathValue = function(colors, edges) {

};
```

TypeScript:

```
function largestPathValue(colors: string, edges: number[][]): number {

};
```

C#:

```
public class Solution {
public int LargestPathValue(string colors, int[][] edges) {

}
}
```

C:

```


int largestPathValue(char * colors, int** edges, int edgesSize, int*
edgesColSize){

}
```

Go:

```
func largestPathValue(colors string, edges [][]int) int {

}
```

Kotlin:

```
class Solution {
fun largestPathValue(colors: String, edges: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func largestPathValue(_ colors: String, _ edges: [[Int]]) -> Int {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn largest_path_value(colors: String, edges: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {String} colors
# @param {Integer[][]} edges
# @return {Integer}
def largest_path_value(colors, edges)

end
```

PHP:

```
class Solution {

/**
* @param String $colors
* @param Integer[][] $edges
* @return Integer
*/
function largestPathValue($colors, $edges) {

}
}
```

Scala:

```
object Solution {
def largestPathValue(colors: String, edges: Array[Array[Int]]): Int = {

}
}
```

Racket:

```
(define/contract (largest-path-value colors edges)
(-> string? (listof (listof exact-integer?)) exact-integer?)

)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Color Value in a Directed Graph
* Difficulty: Hard
* Tags: array, string, graph, dp, hash, sort
*
* 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:
int largestPathValue(string colors, vector<vector<int>>& edges) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Color Value in a Directed Graph
* Difficulty: Hard
* Tags: array, string, graph, dp, hash, sort
*
* 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 int largestPathValue(String colors, int[][] edges) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Largest Color Value in a Directed Graph
Difficulty: Hard
Tags: array, string, graph, dp, hash, sort

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 largestPathValue(self, colors: str, edges: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Color Value in a Directed Graph
* Difficulty: Hard
* Tags: array, string, graph, dp, hash, sort
*
* 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 {string} colors
* @param {number[][]} edges
* @return {number}
*/
var largestPathValue = function(colors, edges) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Color Value in a Directed Graph
* Difficulty: Hard
* Tags: array, string, graph, dp, hash, sort
*
* 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 largestPathValue(colors: string, edges: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Largest Color Value in a Directed Graph
* Difficulty: Hard
* Tags: array, string, graph, dp, hash, sort
*
* 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 int LargestPathValue(string colors, int[][] edges) {
```

```
}
}
```

C Solution:

```
/*
* Problem: Largest Color Value in a Directed Graph
* Difficulty: Hard
* Tags: array, string, graph, dp, hash, sort
*
* 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
*/



int largestPathValue(char * colors, int** edges, int edgesSize, int*
edgesColSize){

}
```

Go Solution:

```
// Problem: Largest Color Value in a Directed Graph
// Difficulty: Hard
// Tags: array, string, graph, dp, hash, sort
//
// 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 largestPathValue(colors string, edges [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largestPathValue(colors: String, edges: Array<IntArray>): Int {
```

```
}
}
```

Swift Solution:

```
class Solution {
func largestPathValue(_ colors: String, _ edges: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Largest Color Value in a Directed Graph
// Difficulty: Hard
// Tags: array, string, graph, dp, hash, sort
//
// 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 largest_path_value(colors: String, edges: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} colors
# @param {Integer[][]} edges
# @return {Integer}
def largest_path_value(colors, edges)

end
```

PHP Solution:

```
class Solution {

/**
```

```
* @param String $colors
* @param Integer[][] $edges
* @return Integer
*/
function largestPathValue($colors, $edges) {

}
}
```

Scala Solution:

```
object Solution {
def largestPathValue(colors: String, edges: Array[Array[Int]]): Int = {

}
}
```

Racket Solution:

```
(define/contract (largest-path-value colors edges)
(-> string? (listof (listof exact-integer?)) exact-integer?)

)
```