Problem 952: Largest Component Size by Common Factor

Problem Information

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

Problem Description

You are given an integer array of unique positive integers

nums

. Consider the following graph:

There are

nums.length

nodes, labeled

nums[0]

to

nums[nums.length - 1]

,

There is an undirected edge between

nums[i]

and

nums[j]

if

nums[i]

and

nums[j]

share a common factor greater than

1

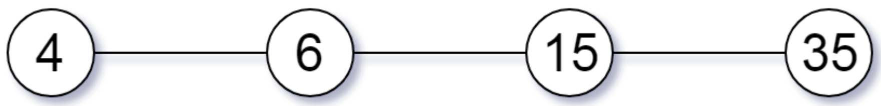
.

Return

the size of the largest connected component in the graph

.

Example 1:



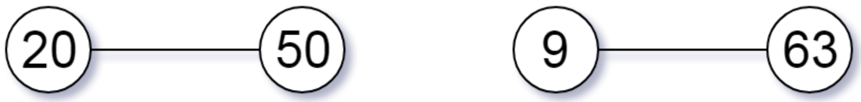
Input:

nums = [4,6,15,35]

Output:

4

Example 2:



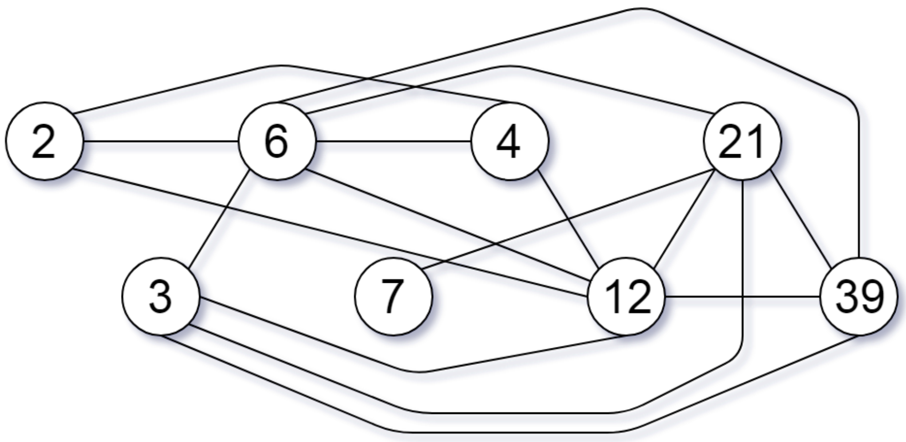
Input:

nums = [20,50,9,63]

Output:

2

Example 3:



Input:

nums = [2,3,6,7,4,12,21,39]

Output:

8

Constraints:

1 <= nums.length <= 2 * 10

4

1 <= nums[i] <= 10

5

All the values of

nums

are

unique

.

Code Snippets

C++:

```
class Solution {
public:
int largestComponentSize(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int largestComponentSize(int[] nums) {

}
}
```

Python3:

```
class Solution:
def largestComponentSize(self, nums: List[int]) -> int:
```

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var largestComponentSize = function(nums) {

};
```

TypeScript:

```
function largestComponentSize(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int LargestComponentSize(int[] nums) {

}
}
```

C:

```
int largestComponentSize(int* nums, int numsSize) {

}
```

Go:

```
func largestComponentSize(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun largestComponentSize(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func largestComponentSize(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn largest_component_size(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def largest_component_size(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function largestComponentSize($nums) {

}

}
```

Dart:

```
class Solution {
int largestComponentSize(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def largestComponentSize(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_component_size(nums :: [integer]) :: integer
def largest_component_size(nums) do

end
end
```

Erlang:

```
-spec largest_component_size(Nums :: [integer()]) -> integer().
largest_component_size(Nums) ->
.
```

Racket:

```
(define/contract (largest-component-size nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Component Size by Common Factor
* Difficulty: Hard
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int largestComponentSize(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Component Size by Common Factor
* Difficulty: Hard
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int largestComponentSize(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Component Size by Common Factor
Difficulty: Hard
Tags: array, graph, math, hash
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Component Size by Common Factor
* Difficulty: Hard
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[]} nums
* @return {number}
*/
var largestComponentSize = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Component Size by Common Factor
* Difficulty: Hard
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function largestComponentSize(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Largest Component Size by Common Factor
* Difficulty: Hard
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int LargestComponentSize(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Largest Component Size by Common Factor
* Difficulty: Hard
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

int largestComponentSize(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Largest Component Size by Common Factor
// Difficulty: Hard
// Tags: array, graph, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func largestComponentSize(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largestComponentSize(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func largestComponentSize(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Largest Component Size by Common Factor
// Difficulty: Hard
// Tags: array, graph, math, hash
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn largest_component_size(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def largest_component_size(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function largestComponentSize($nums) {

}
}
```

Dart Solution:

```
class Solution {
int largestComponentSize(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def largestComponentSize(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_component_size(nums :: [integer]) :: integer
def largest_component_size(nums) do

end
end
```

Erlang Solution:

```
-spec largest_component_size(Nums :: [integer()]) -> integer().
largest_component_size(Nums) ->
.
```

Racket Solution:

```
(define/contract (largest-component-size nums)
(-> (listof exact-integer?) exact-integer?)
)
```