Problem 1979: Find Greatest Common Divisor of Array

Problem Information

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

Problem Description

Given an integer array

nums

, return

the

greatest common divisor

of the smallest number and largest number in

nums

.

The

greatest common divisor

of two numbers is the largest positive integer that evenly divides both numbers.

Example 1:

Input:

nums = [2,5,6,9,10]

Output:

2

Explanation:

The smallest number in nums is 2. The largest number in nums is 10. The greatest common divisor of 2 and 10 is 2.

Example 2:

Input:

nums = [7,5,6,8,3]

Output:

1

Explanation:

The smallest number in nums is 3. The largest number in nums is 8. The greatest common divisor of 3 and 8 is 1.

Example 3:

Input:

nums = [3,3]

Output:

3

Explanation:

The smallest number in nums is 3. The largest number in nums is 3. The greatest common divisor of 3 and 3 is 3.

Constraints:

2 <= nums.length <= 1000

1 <= nums[i] <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
```

```
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-gcd nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Greatest Common Divisor of Array
* Difficulty: Easy
* Tags: array, math
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Find Greatest Common Divisor of Array
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Find Greatest Common Divisor of Array
Difficulty: Easy
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Greatest Common Divisor of Array
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Find Greatest Common Divisor of Array
* Difficulty: Easy
* Tags: array, math
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

C# Solution:

```
/*
* Problem: Find Greatest Common Divisor of Array
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

C Solution:

```
/*
* Problem: Find Greatest Common Divisor of Array
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

```
}
```

Go Solution:

```
// Problem: Find Greatest Common Divisor of Array
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findGCD(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Greatest Common Divisor of Array
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
```

```
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

```
}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-gcd nums)
(-> (listof exact-integer?) exact-integer?)
)
```