Problem 2475: Number of Unequal Triplets in Array

Problem Information

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

Problem Description

You are given a

0-indexed

array of positive integers

nums

. Find the number of triplets

(i, j, k)

that meet the following conditions:

0 <= i < j < k < nums.length

nums[i]

,

nums[j]

, and

nums[k]

are

pairwise distinct

.

In other words,

nums[i] != nums[j]

,

nums[i] != nums[k]

, and

nums[j] != nums[k]

.

Return

the number of triplets that meet the conditions.

Example 1:

Input:

nums = [4,4,2,4,3]

Output:

3

Explanation:

The following triplets meet the conditions: - (0, 2, 4) because 4 != 2 != 3 - (1, 2, 4) because 4 != 2 != 3 - (2, 3, 4) because 2 != 4 != 3 Since there are 3 triplets, we return 3. Note that (2, 0, 4) is not a valid triplet because 2 > 0.

Example 2:

Input:

nums = [1,1,1,1,1]

Output:

0

Explanation:

No triplets meet the conditions so we return 0.

Constraints:

3 <= nums.length <= 100

1 <= nums[i] <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (unequal-triplets nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Unequal Triplets in Array
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 unequalTriplets(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Unequal Triplets in Array
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 unequalTriplets(int[] nums) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Number of Unequal Triplets in Array
Difficulty: Easy
Tags: array, hash, sort

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 unequalTriplets(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Unequal Triplets in Array
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 unequalTriplets = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Unequal Triplets in Array
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 unequalTriplets(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Unequal Triplets in Array
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 UnequalTriplets(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Number of Unequal Triplets in Array
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 unequalTriplets(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Number of Unequal Triplets in Array
// Difficulty: Easy
// Tags: array, hash, sort
//
// 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 unequalTriplets(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Number of Unequal Triplets in Array
// Difficulty: Easy
// Tags: array, hash, sort
//
// 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 unequal_triplets(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (unequal-triplets nums)
(-> (listof exact-integer?) exact-integer?)
)
```