

Problem 2367: Number of Arithmetic Triplets

Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a

0-indexed

,

strictly increasing

integer array

nums

and a positive integer

diff

. A triplet

(i, j, k)

is an

arithmetic triplet

if the following conditions are met:

$i < j < k$

,

$\text{nums}[j] - \text{nums}[i] == \text{diff}$

, and

$\text{nums}[k] - \text{nums}[j] == \text{diff}$

.

Return

the number of unique

arithmetic triplets

.

Example 1:

Input:

$\text{nums} = [0, 1, 4, 6, 7, 10]$, $\text{diff} = 3$

Output:

2

Explanation:

(1, 4, 7) is an arithmetic triplet because both $7 - 4 == 3$ and $4 - 1 == 3$. (4, 7, 10) is an arithmetic triplet because both $10 - 7 == 3$ and $7 - 4 == 3$.

Example 2:

Input:

nums = [4,5,6,7,8,9], diff = 2

Output:

2

Explanation:

(0, 2, 4) is an arithmetic triplet because both $8 - 6 == 2$ and $6 - 4 == 2$. (1, 3, 5) is an arithmetic triplet because both $9 - 7 == 2$ and $7 - 5 == 2$.

Constraints:

$3 \leq \text{nums.length} \leq 200$

$0 \leq \text{nums}[i] \leq 200$

$1 \leq \text{diff} \leq 50$

nums

is

strictly

increasing.

Code Snippets

C++:

```
class Solution {
public:
    int arithmeticTriplets(vector<int>& nums, int diff) {

    }
};
```

Java:

```

class Solution {
public int arithmeticTriplets(int[] nums, int diff) {

}

}

```

Python3:

```

class Solution:
def arithmeticTriplets(self, nums: List[int], diff: int) -> int:

```

Python:

```

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

```

JavaScript:

```

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

};

```

TypeScript:

```

function arithmeticTriplets(nums: number[], diff: number): number {

};

```

C#:

```

public class Solution {
public int ArithmeticTriplets(int[] nums, int diff) {

```

```
}  
}
```

C:

```
int arithmeticTriplets(int* nums, int numsSize, int diff) {  
  
}
```

Go:

```
func arithmeticTriplets(nums []int, diff int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun arithmeticTriplets(nums: IntArray, diff: Int): Int {  
  
    }  
}
```

Swift:

```
class Solution {  
    func arithmeticTriplets(_ nums: [Int], _ diff: Int) -> Int {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn arithmetic_triplets(nums: Vec<i32>, diff: i32) -> i32 {  
  
    }  
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} diff
# @return {Integer}
def arithmetic_triplets(nums, diff)

end
```

PHP:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @param Integer $diff
     * @return Integer
     */
    function arithmeticTriplets($nums, $diff) {

    }

}
```

Dart:

```
class Solution {
  int arithmeticTriplets(List<int> nums, int diff) {

  }
}
```

Scala:

```
object Solution {
  def arithmeticTriplets(nums: Array[Int], diff: Int): Int = {

  }
}
```

Elixir:

```
defmodule Solution do
  @spec arithmetic_triplets(nums :: [integer], diff :: integer) :: integer
  def arithmetic_triplets(nums, diff) do
```

```
end  
end
```

Erlang:

```
-spec arithmetic_triplets(Nums :: [integer()], Diff :: integer()) ->  
integer().  
arithmetic_triplets(Nums, Diff) ->  
.
```

Racket:

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

Solutions

C++ Solution:

```
/*  
 * Problem: Number of Arithmetic Triplets  
 * Difficulty: Easy  
 * Tags: array, 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 arithmeticTriplets(vector<int>& nums, int diff) {  
  
    }  
};
```

Java Solution:

```

/**
 * Problem: Number of Arithmetic Triplets
 * Difficulty: Easy
 * Tags: array, 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 arithmeticTriplets(int[] nums, int diff) {

}
}

```

Python3 Solution:

```

"""
Problem: Number of Arithmetic Triplets
Difficulty: Easy
Tags: array, 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 arithmeticTriplets(self, nums: List[int], diff: int) -> int:
# TODO: Implement optimized solution
pass

```

Python Solution:

```

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

```

JavaScript Solution:

```
/**
 * Problem: Number of Arithmetic Triplets
 * Difficulty: Easy
 * Tags: array, 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
 * @param {number} diff
 * @return {number}
 */
var arithmeticTriplets = function(nums, diff) {

};
```

TypeScript Solution:

```
/**
 * Problem: Number of Arithmetic Triplets
 * Difficulty: Easy
 * Tags: array, 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 arithmeticTriplets(nums: number[], diff: number): number {

};
```

C# Solution:

```
/*
 * Problem: Number of Arithmetic Triplets
 * Difficulty: Easy
```

```

* Tags: array, 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 ArithmeticTriplets(int[] nums, int diff) {

}

}

```

C Solution:

```

/*
* Problem: Number of Arithmetic Triplets
* Difficulty: Easy
* Tags: array, 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 arithmeticTriplets(int* nums, int numsSize, int diff) {

}

```

Go Solution:

```

// Problem: Number of Arithmetic Triplets
// Difficulty: Easy
// Tags: array, 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 arithmeticTriplets(nums []int, diff int) int {

```

```
}
```

Kotlin Solution:

```
class Solution {  
    fun arithmeticTriplets(nums: IntArray, diff: Int): Int {  
  
    }  
}
```

Swift Solution:

```
class Solution {  
    func arithmeticTriplets(_ nums: [Int], _ diff: Int) -> Int {  
  
    }  
}
```

Rust Solution:

```
// Problem: Number of Arithmetic Triplets  
// Difficulty: Easy  
// Tags: array, 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 arithmetic_triplets(nums: Vec<i32>, diff: i32) -> i32 {  
  
    }  
}
```

Ruby Solution:

```
# @param {Integer[]} nums  
# @param {Integer} diff  
# @return {Integer}  
def arithmetic_triplets(nums, diff)
```

```
end
```

PHP Solution:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @param Integer $diff  
     * @return Integer  
     */  
    function arithmeticTriplets($nums, $diff) {  
  
    }  
}
```

Dart Solution:

```
class Solution {  
    int arithmeticTriplets(List<int> nums, int diff) {  
  
    }  
}
```

Scala Solution:

```
object Solution {  
    def arithmeticTriplets(nums: Array[Int], diff: Int): Int = {  
  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
    @spec arithmetic_triplets(nums :: [integer], diff :: integer) :: integer  
    def arithmetic_triplets(nums, diff) do  
  
    end  
end
```

Erlang Solution:

```
-spec arithmetic_triplets(Nums :: [integer()], Diff :: integer()) ->
integer().
arithmetic_triplets(Nums, Diff) ->
.
```

Racket Solution:

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