Problem 259: 3Sum Smaller

Problem Information

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

Problem Description

Given an array of

n

integers

nums

and an integer

target

, find the number of index triplets

i

,

j

,

k

with

0 <= i < j < k < n

that satisfy the condition

nums[i] + nums[j] + nums[k] < target

.

Example 1:

Input:

nums = [-2,0,1,3], target = 2

Output:

2

Explanation:

Because there are two triplets which sums are less than 2: [-2,0,1] [-2,0,3]

Example 2:

Input:

nums = [], target = 0

Output:

0

Example 3:

Input:

nums = [0], target = 0

Output:

0

Constraints:

n == nums.length

0 <= n <= 3500

-100 <= nums[i] <= 100

-100 <= target <= 100

Code Snippets

C++:

```
class Solution {
public:
int threeSumSmaller(vector<int>& nums, int target) {

}
};
```

Java:

```
class Solution {
public int threeSumSmaller(int[] nums, int target) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def threeSumSmaller(self, nums, target):
```

```
"""
:type nums: List[int]
:type target: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function threeSumSmaller(nums: number[], target: number): number {

};
```

C#:

```
public class Solution {
public int ThreeSumSmaller(int[] nums, int target) {

}
}
```

C:

```
int threeSumSmaller(int* nums, int numsSize, int target) {

}
```

Go:

```
func threeSumSmaller(nums []int, target int) int {

}
```

Kotlin:

```
class Solution {
fun threeSumSmaller(nums: IntArray, target: Int): Int {

}
}
```

Swift:

```
class Solution {
func threeSumSmaller(_ nums: [Int], _ target: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn three_sum_smaller(nums: Vec<i32>, target: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} target
# @return {Integer}
def three_sum_smaller(nums, target)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $target
* @return Integer
*/
function threeSumSmaller($nums, $target) {
```

```
}
}
```

Dart:

```
class Solution {
int threeSumSmaller(List<int> nums, int target) {

}
}
```

Scala:

```
object Solution {
def threeSumSmaller(nums: Array[Int], target: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec three_sum_smaller(nums :: [integer], target :: integer) :: integer
def three_sum_smaller(nums, target) do

end
end
```

Erlang:

```
-spec three_sum_smaller(Nums :: [integer()], Target :: integer()) ->
integer().
three_sum_smaller(Nums, Target) ->
.
```

Racket:

```
(define/contract (three-sum-smaller nums target)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: 3Sum Smaller
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 threeSumSmaller(vector<int>& nums, int target) {

}
};
```

Java Solution:

```
/**
* Problem: 3Sum Smaller
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 threeSumSmaller(int[] nums, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: 3Sum Smaller
Difficulty: Medium
Tags: array, sort, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: 3Sum Smaller
* Difficulty: Medium
* Tags: array, sort, search
*
* 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
* @param {number} target
* @return {number}
*/
```

```
var threeSumSmaller = function(nums, target) {

};
```

TypeScript Solution:

```
/**
* Problem: 3Sum Smaller
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 threeSumSmaller(nums: number[], target: number): number {

};
```

C# Solution:

```
/*
* Problem: 3Sum Smaller
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 ThreeSumSmaller(int[] nums, int target) {

}
}
```

C Solution:

```
/*
* Problem: 3Sum Smaller
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 threeSumSmaller(int* nums, int numsSize, int target) {

}
```

Go Solution:

```
// Problem: 3Sum Smaller
// Difficulty: Medium
// Tags: array, sort, search
//
// 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 threeSumSmaller(nums []int, target int) int {

}
```

Kotlin Solution:

```
class Solution {
fun threeSumSmaller(nums: IntArray, target: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func threeSumSmaller(_ nums: [Int], _ target: Int) -> Int {

}
```

```
}
```

Rust Solution:

```
// Problem: 3Sum Smaller
// Difficulty: Medium
// Tags: array, sort, search
//
// 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 three_sum_smaller(nums: Vec<i32>, target: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} target
# @return {Integer}
def three_sum_smaller(nums, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $target
* @return Integer
*/
function threeSumSmaller($nums, $target) {

}
}
```

Dart Solution:

```
class Solution {
int threeSumSmaller(List<int> nums, int target) {

}
}
```

Scala Solution:

```
object Solution {
def threeSumSmaller(nums: Array[Int], target: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec three_sum_smaller(nums :: [integer], target :: integer) :: integer
def three_sum_smaller(nums, target) do

end
end
```

Erlang Solution:

```
-spec three_sum_smaller(Nums :: [integer()], Target :: integer()) ->
integer().
three_sum_smaller(Nums, Target) ->
.
```

Racket Solution:

```
(define/contract (three-sum-smaller nums target)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```