# 题目

给你一个下标从 0 开始、严格递增 的整数数组 nums 和一个正整数 diff 。如果满足下述全部条件，则三元组 (i, j, k) 就是一个 算术三元组 ：

i < j < k，

nums[j] - nums[i] == diff 且

nums[k] - nums[j] == diff

返回不同 算术三元组 的数目。

示例 1：

输入：nums = [0,1,4,6,7,10], diff = 3

输出：2

解释：

(1, 2, 4) 是算术三元组：7 - 4 == 3 且 4 - 1 == 3 。

(2, 4, 5) 是算术三元组：10 - 7 == 3 且 7 - 4 == 3 。

示例 2：

输入：nums = [4,5,6,7,8,9], diff = 2

输出：2

解释：

(0, 2, 4) 是算术三元组：8 - 6 == 2 且 6 - 4 == 2 。

(1, 3, 5) 是算术三元组：9 - 7 == 2 且 7 - 5 == 2 。

提示：

3 <= nums.length <= 200

0 <= nums[i] <= 200

1 <= diff <= 50

nums 严格 递增

# 分析

## 方法一：暴力破解

思路：

代码：

class Solution {

public:

int arithmeticTriplets(vector<int>& nums, int diff) {

int ans = 0;

int n = nums.size();

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (nums[j] - nums[i] != diff) {

continue;

}

for (int k = j + 1; k < n; k++) {

if (nums[k] - nums[j] == diff) {

ans++;

}

}

}

}

return ans;

}

};

复杂度分析

时间复杂度：O(n^3)，其中 n是数组nums的长度。使用三重循环暴力枚举需要O(n^3)

的时间。

空间复杂度：O(1)。只需要常数的额外空间。

## 方法二：哈希表

思路：

代码：

class Solution {

public:

int arithmeticTriplets(vector<int>& nums, int diff) {

unordered\_set<int> hashSet;

for (int x : nums) {

hashSet.emplace(x);

}

int ans = 0;

for (int x : nums) {

if (hashSet.count(x + diff) && hashSet.count(x + 2 \* diff)) {

ans++;

}

}

return ans;

}

};

复杂度分析

时间复杂度：O(n)，其中n是数组nums的长度。需要遍历数组两次，每次将元素加入哈希集合与判断元素是否在哈希集合中的时间都是O(1)。

空间复杂度：O(n)，其中n是数组nums的长度。哈希集合需要O(n)的空间。