题目地址（762.Number Stream to Intervals）

https://binarysearch.com/problems/Triple-Inversion

入选理由

1.通常二分都是基于有序序列二分，题目直接给了有序序列自然就简单，如果题目没给就需要我们自己构造，这道题就是。 2. 和专题篇的某一个专题联动（会是谁呢？） 3. 力扣中有换皮题，其实这道题就是典型的 xxx（猜猜是啥）。

标签

- 二分

难度

- 困难

题目描述

```
Given a list of integers nums, return the number of pairs i < j such that nums[i] > nums[j] * 3.

Constraints

n ≤ 100,000 where n is the length of nums
Example 1
Input
nums = [7, 1, 2]
Output
2
Explanation
We have the pairs (7, 1) and (7, 2)
```

前置知识

- 二分法

暴力法（超时）

思路

本题和力扣 **493. 翻转对**[1] 和 **剑指 Offer 51. 数组中的逆序对**[2] 一样，都是求逆序对的换皮题。

暴力的解法可以枚举所有可能的 j，然后往前找 i 使得满足 $\mathrm{nums[i]} > \mathrm{nums[j]} * 3$，我们要做的就是将满足这种条件的 i 数出来有几个即可。这种算法时间复杂度为 $\mathrm{O(n^2)}$。

代码

代码支持：Python3

Python3 Code:

```
class Solution:
    def solve(self, A):
        ans = 0
        for i in range(len(A)):
            for j in range(i+1,len(A)):
                if A[i] > A[j] * 3: ans += 1
        return ans
```

复杂度分析

令 n 为数组长度。

- 时间复杂度：$\mathrm{O(n^2)}$
- 空间复杂度：$\mathrm{O(1)}$

二分法

思路

这道题我们也可以反向思考。即思考：对于 nums 中的每一项 num，我们找前面出现过的大于 num * 3 的数。

我们可以自己构造有序序列 d，然后在 d 上做二分。如何构建 d 呢？很简单，就是将 nums 中已经遍历过的数字全部放到 d 中即可。

代码表示就是：

```
d = []
for a in A:
    bisect.insort(d, a)
```

bisect.insort 指的是使用二分找到插入点，并将数插入到数组中，使得**插入后数组仍然有序**。虽然使用了二分，使得找到插入点的时间复杂度为 $O(logn)$，但是由于数组的特性，插入导致的数组项后移的时间复杂度为 $O(n)$，因此总的时间复杂度为 $O(n^2)$。

Python3 Code:

```
class Solution:
    def solve(self, A):
        d = []
        ans = 0

        for a in A:
            i = bisect.bisect_right(d, a * 3)
            ans += len(d) - i
            bisect.insort(d, a)
```

由于上面的算法瓶颈在于数组的插入后移带来的时间。因此我们可以使用平衡二叉树来减少这部分时间，使用平衡二叉树可以使得插入时间稳定在 $O(logn)$，Python 可使用 SortedList 来实现， Java 可用 TreeMap 代替。

代码

代码支持：Python3

Python3 Code:

```
from sortedcontainers import SortedList
class Solution:
    def solve(self, A):
        d = SortedList()
        ans = 0

        for a in A:
            i = d.bisect_right(a * 3)
            ans += len(d) - i
            d.add(a)
        return ans
```

复杂度分析

令 n 为数组长度。

- 时间复杂度：$O(nlogn)$
- 空间复杂度：$O(n)$

分治法

思路

我们接下来介绍更广泛使用的，效率更高的解法 分治 。 我们进行一次归并排序，并在归并过程中计算逆序数，换句话说 逆序对是归并排序的副产物 。

> 如果不熟悉归并排序，可以先查下相关资料。 如果你直接想看归并排序代码，那么将的代码统计 cnt 部分删除就好了。

归并排序实际上会把数组分成两个有序部分，我们不妨称其为左和右，归并排序的过程中会将左右两部分合并成一个有序的部分，对于每一个左右部分，我们分别计算其逆序数，然后全部加起来就是我们要求的逆序数。 那么分别如何求解左右部分的逆序数呢?

首先我们知道归并排序的核心在于合并，而合并两个有序数组是一个**简单题目**。 我这里给贴一下大概算法：

```
def mergeTwo(nums1, nums2):
    res = []
    i = j = 0
    while i < len(nums1) and j < len(nums2):
        if nums1[i] < nums[j]:
            res.append(nums[i])
            i += 1
        else:
            res.append(nums[j])
            j += 1
    while i < len(nums1) :
        res.append(num[i])
        i += 1
    while j < len(nums1) :
        res.append(num[j])
        j += 1
    return res
```

而我们要做的就是在上面的合并过程中统计逆序数。

> 为了方便描述，我将题目中的：i < j such that nums[i] > nums[j] * 3，改成 i < j such that nums[i] > nums[j]。也就是将 3 倍变成一倍。 如果你理解了这个过程，只需要比较的时候乘以 3 就行，其他逻辑不变。

比如对于左：[1，2，**3**，4]右：[**2**，5]。由于我的算法是按照 [start, mid), [mid, end) 区间分割的，因此这里的 mid 为 3（具体可参考下方代码区）。 其中 `i` ， `j` 指针如粗体部分（左数组的 3 和右数组的 2）。 那么 逆序数就是 `mid - i + 1` 也就

是 `3 - 2 + 1 = 2` 即 `(3, 2)` 和 `(4, 2)` 。其原因在于如果 3 大于 2，那么 3 后面不用看了，肯定都大于 2。 之后会变成：[1，2，**3**，4] 右：[2，**5**] （左数组的 3 和 右数组的 5），继续按照上面的方法计算直到无法进行即可。

```
class Solution:
    def solve(self, nums: List[int]) -> int:
        self.cnt = 0
        def merge(nums, start, mid, end):
            i, j, temp = start, mid + 1, []
            while i <= mid and j <= end:
                if nums[i] <= nums[j]:
                    temp.append(nums[i])
                    i += 1
                else:
                    self.cnt += mid - i + 1
                    temp.append(nums[j])
                    j += 1
            while i <= mid:
                temp.append(nums[i])
                i += 1
            while j <= end:
                temp.append(nums[j])
                j += 1

            for i in range(len(temp)):
                nums[start + i] = temp[i]


        def mergeSort(nums, start, end):
            if start >= end: return
            mid = (start + end) >> 1
            mergeSort(nums, start, mid)
            mergeSort(nums, mid + 1, end)
            merge(nums, start, mid,  end)
        mergeSort(nums, 0, len(nums) - 1)
        return self.cnt
```

注意上述算法在 mergeSort 中我们每次都开辟一个新的 temp，这样空间复杂度大概相当于 NlogN，实际上我们完全每必要每次 mergeSort 都开辟一个新的，而是大家也都用一个。具体见下方代码区。

代码

代码支持：Python3

Python3 Code:

```python
class Solution:
    def solve(self, nums) -> int:
        self.cnt = 0
        def merge(nums, start, mid, end, temp):
            i, j = start, mid + 1
            while i <= mid and j <= end:
                if nums[i] <=  nums[j]:
                    temp.append(nums[i])
                    i += 1
                else:
                    temp.append(nums[j])
                    j += 1
            # 防住
            # 这里代码开始
            ti, tj = start, mid + 1
            while ti <= mid and tj <= end:
                if nums[ti] <=  3 * nums[tj]:
                    ti += 1
                else:
                    self.cnt += mid - ti + 1
                    tj += 1
            # 这里代码结束
            while i <= mid:
                temp.append(nums[i])
                i += 1
            while j <= end:
                temp.append(nums[j])
                j += 1
            for i in range(len(temp)):
                nums[start + i] = temp[i]
            temp.clear()


        def mergeSort(nums, start, end, temp):
            if start >= end: return
            mid = (start + end) >> 1
            mergeSort(nums, start, mid, temp)
            mergeSort(nums, mid + 1, end, temp)
            merge(nums, start, mid,  end, temp)
        mergeSort(nums, 0, len(nums) - 1, [])
        return self.cnt
```

复杂度分析

令 n 为数组长度。

- 时间复杂度：$O(nlogn)$
- 空间复杂度：$O(n)$

力扣的小伙伴可以**关注我**，这样就会第一时间收到我的动态啦~

以上就是本文的全部内容了。大家对此有何看法，欢迎给我留言，我有时间都会一一查看回答。更多算法套路可以访问我的 LeetCode 题解仓库：https://github.com/azl397985856/leetcode 。 目前已经 40K star 啦。大家也可以关注我的公众号《力扣加加》带你啃下算法这块硬骨头。

参考资料

[1] 493. 翻转对: *https://leetcode-cn.com/problems/reverse-pairs/solution/jian-dan-yi-dong-gui-bing-pai-xu-493-fan-zhuan-dui/*

[2] 剑指 Offer 51. 数组中的逆序对: *https://leetcode-cn.com/problems/shu-zu-zhong-de-ni-xu-dui-lcof/solution/jian-dan-yi-dong-gui-bing-pai-xu-python-by-azl3979/*

上一页 下一页