leetcode Count of Smaller Numbers After Self

2015年12月6日 (https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/) hrwhisper (https://www.hrwhisper.me/author/hrsay/) 4 Comments (https://www.hrwhisper.me/leetcode-count-of-smalle numbers-after-self/#comments) 3,914 views

> ## leetcode Count of Smaller Numbers After Self
>
> You are given an integer array *nums* and you have to return a new *counts* array The *counts* array has the property where `counts[i]` is the number of smaller elements to the right of `nums[i]`.
>
> **Example:**
>
> Given nums = [5, 2, 6, 1]
>
> To the right of 5 there are 2 smaller elements (2 and 1).
> To the right of 2 there is only 1 smaller element (1).
> To the right of 6 there is 1 smaller element (1).
> To the right of 1 there is 0 smaller element.
>
> Return the array `[2, 1, 1, 0]`.

题目地址 leetcode Count of Smaller Numbers After Self (https://leetcode.com/problems/count-of-smaller-numbers-after-self/)

题意：

给定nums数组，求数组中每个元素i的右边比其小的数

思路：

简单的说就是求逆序数。

1. 使用逆序数有经典的解法为合并排序。
2. 用Fenwick树 关于Fenwick 树介绍 Binary indexed tree (Fenwick tree) (https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/)
    - 简单说就是看当前数在nums中排第几，然后对小于它的数求个数和
    - 具体的做法是先离散化，确定每个数载nums中排到第几（去重和排序）
    - 然后从右向左扫描，每次统计比其小于1的个数（就是求和），然后把当前的数 Fenwick中。

merge_sort

C++

```
struct Node {
    int val;
    int index;
    int cnt;
    Node(int val, int index) : val(val), index(index), cnt(0) {}
    bool operator <= (const Node &node2)const {
        return val <= node2.val;
    }
};

class Solution {
public:
    void combine(vector<Node> &nums, int Lpos, int Lend, int Rend, vector<Node> &temp
        int Rpos = Lend + 1;
        int Tpos = Lpos;
        int n = Rend - Lpos + 1;
        int t = Rpos;
        while (Lpos <= Lend && Rpos <= Rend) {
            if (nums[Lpos] <= nums[Rpos]) {
                temp[Tpos] = nums[Lpos];
                temp[Tpos].cnt += Rpos - t ;
                Tpos++; Lpos++;
            }
            else {
                temp[Tpos++] = nums[Rpos++];
            }
        }

        while (Lpos <= Lend) {
            temp[Tpos] = nums[Lpos];
            temp[Tpos].cnt += Rpos - t;
            Tpos++; Lpos++;
        }

        while (Rpos <= Rend)
            temp[Tpos++] = nums[Rpos++];

        for (int i = 0; i< n; i++, Rend--)
            nums[Rend] = temp[Rend];
    }

    void merge_sort(vector<Node> & nums, int L, int R, vector<Node> &temp) {
        if (L < R) {
            int m = (L + R) >> 1;
            merge_sort(nums, L, m, temp);
            merge_sort(nums, m + 1, R, temp);
            combine(nums, L, m, R, temp);
        }
    }

    vector<int> countSmaller(vector<int>& nums) {
        vector<Node> mynums;
        vector<Node> temp(nums.size(), Node(0, 0));
        for (int i = 0; i < nums.size(); i++)
            mynums.push_back(Node(nums[i], i));

        vector<int> ans(nums.size(), 0);
        merge_sort(mynums, 0, nums.size() - 1, temp);

        for (int i = 0; i < nums.size(); i++)
            ans[mynums[i].index] = mynums[i].cnt;

        return ans;
    }
};
```

Binary indexed tree (Fenwick tree)

C++

```
class FenwickTree {
    vector<int> sum_array;
    int n;
    inline int lowbit(int x) {
        return x & -x;
    }

public:
    FenwickTree(int n) :n(n), sum_array(n + 1, 0) {}

    void add(int x, int val) {
        while (x <= n) {
            sum_array[x] += val;
            x += lowbit(x);
        }
    }

    int sum(int x) {
        int res = 0;
        while (x > 0) {
            res += sum_array[x];
            x -= lowbit(x);
        }
        return res;
    }
};

class Solution {
public:
    vector<int> countSmaller(vector<int>& nums) {
        vector<int> temp_num = nums;
        sort(temp_num.begin(), temp_num.end());
        unordered_map<int,int> dic;
        for (int i = 0; i < temp_num.size(); i++)
            dic[temp_num[i]] = i + 1;

        FenwickTree tree(nums.size());
        vector<int> ans(nums.size(),0);
        for (int i = nums.size() - 1; i >= 0; i--) {
            ans[i] = tree.sum(dic[nums[i]] - 1);
```

Python

```python
class FenwickTree(object):
    def __init__(self, n):
        self.sum_array = [0] * (n + 1)
        self.n = n

    def lowbit(self, x):
        return x & -x

    def add(self, x, val):
        while x <= self.n:
            self.sum_array[x] += val
            x += self.lowbit(x)

    def sum(self, x):
        res = 0
        while x > 0:
            res += self.sum_array[x]
            x -= self.lowbit(x)
        return res


class Solution(object):
    def countSmaller(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        dic = {}
        for i, num in enumerate(sorted(list(set(nums)))):
            dic[num] = i + 1
        tree = FenwickTree(len(nums))
        ans = [0] * len(nums)
        for i in xrange(len(nums) - 1, -1, -1):
            ans[i] = tree.sum(dic[nums[i]] - 1)
            tree.add(dic[nums[i]], 1)
        return ans
```

> *本博客若无特殊说明则由 hrwhisper (https://www.hrwhisper.me) 原创发布*
> *转载请点名出处：细语呢喃 (https://www.hrwhisper.me) > leetcode Count of Sm*
> *Numbers After Self (https://www.hrwhisper.me/leetcode-count-of-smaller-numb*
> *after-self/)*
> *本文地址：https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-s*
> *(https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/)*

分享到： 新浪微博 微信 QQ空间 人人 腾讯微博 豆瓣 印象笔记

Leetcode (https://www.hrwhisper.me/category/code/leetcode/) c++ (https://www.hrwhisper.me/tag/c/), lee (https://www.hrwhisper.me/tag/leetcode/), python (https://www.hrwhisper.me/tag/python/), 算法 (https://www.hrwhisper.me/tag/algorithm/). permalink (https://www.hrwhisper.me/leetcode-count-of-smaller-nu after-self/).

‹ windows 10 远程桌面提示凭证无法工作解决办法 (https://www.hrwhisper.me/windows-10-remote-desktop-credential-not-work-solution/)

Binary indexed tree (Fenwick tree) › (https://www.hrwhisper.me/binary-indexed-tree-fenwick-

4 thoughts on “leetcode Count of Smaller Numbers After Self”

Xiaoye says:
2016年5月11日 at pm3:11 (https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/#comment-998)

你好博主，能不能对

for i in xrange(len(nums) – 1, -1, -1):
ans[i] = tree.sum(dic[nums[i]] – 1)

tree.add(dic[nums[i]], 1)

这段循环做个注释呢？
我看不懂里面的逻辑…..

Reply (https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/?replytocom=998#respond

hrwhisper (https://www.hrwhisper.me) says:
2016年5月11日 at pm7:28 (https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/#comment-1002)

从右向左扫描，每次统计比其小于1的个数（就是求和），然后把当前的数加入Fenwick中（当前数在序列中排的位置（因为前面离散化了）个数加1）。

tps://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/?replytocom=1002#respond

Pingback: leetcode Count of Range Sum - 细语呢喃 (https://www.hrwhisper.me/leetcode-count-of-range
Pingback: leetcode — Count of Smaller Numbers After Self — 经典求逆序数-IT大道 (http://www.itdadao.
/article/74301/)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

(http://weibo.com/murmured)

(https://twitter.com/hrwhisper)

(https://instagram.com/hr_say/)

(https://github.com/hrwhisper)

(https://www.hrwhisper.me/feed)

Csdn (http://blog.csdn.net/murmured) 博客园 (http://www.cnblogs.com/murmured/)

Lofter (http://hrsay.lofter.com/) 知 乎 (http://www.zhihu.com/people/hrwhisper)

豆瓣 (http://www.douban.com/people/hrwhisper/)

努力的人本身就有奇迹 | 快乐是我们共同的

by hrwhipse