

LintCode领扣题解 (/problem) / 逆序对 · Reverse Pairs

## 逆序对 · Reverse Pairs

中文

NetEase (/problem/?tags=netease)

谷歌 (/problem/?tags=google)

归并排序 (/problem/?tags=merge-sort)

数组 (/problem/?tags=array)

### 描述

在数组中的两个数字如果前面一个数字大于后面的数字, 则这两个数字组成一个逆序对。给你一个数组, 求出这个数组中逆序对的总数。

概括: 如果 $a[i] > a[j]$  且  $i < j$ ,  $a[i]$  和  $a[j]$  构成一个逆序对。

### 样例

#### 样例 1

输入: A = [2, 4, 1, 3, 5]  
输出: 3  
解释:  
(2, 1), (4, 1), (4, 3) 是逆序对

#### 样例 2

输入: A = [1, 2, 3, 4]  
输出: 0  
解释:  
没有逆序对

在线评测地址: <https://www.lintcode.com/problem/reverse-pairs/> (<https://www.lintcode.com/problem/reverse-pairs/>)

收起题目描述 ^

语言类型

ALL (17)

java (7)

python (7)

cpp (3)

上传题解



令狐冲

更新于 6/9/2020, 7:03:51 AM

利用归并排序的思想求逆序对, 复杂度 $O(n\log n)$  当然也可以用树状数组或者线段树求解

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    # @param {int[]} A an array
    # @return {int} total of reverse pairs
    def reversePairs(self, A):
        # Write your code here
        self.tmp = [0] * len(A)
        return self.mergeSort(A, 0, len(A) - 1)

    def mergeSort(self, A, l, r):
        if l >= r:
            return 0

        m = (l + r) >> 1
        ans = self.mergeSort(A, l, m) + self.mergeSort(A, m + 1, r)
        i, j, k = l, m + 1, l
        while i <= m and j <= r:
            if A[i] > A[j]:
                self.tmp[k] = A[j]
                j += 1
                ans += m - i + 1
            else:
                self.tmp[k] = A[i]
                i += 1
            k += 1

        while i <= m:
            self.tmp[k] = A[i]
            k += 1
            i += 1
        while j <= r:
            self.tmp[k] = A[j]
            k += 1
            j += 1
        for i in xrange(l, r + 1):
            A[i] = self.tmp[i]

        return ans
```

👍 获赞 2

💬 1 条评论

# 你的口袋题库

## 2000+算法真题、国内外名企题库免费开放



九章算法APP

令狐冲

更新于 6/9/2020, 7:03:58 AM

利用归并排序的思想求逆序对, 复杂度 $O(n\log n)$  当然也可以用树状数组或者线段树求解

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class Solution {
    /**
     * @param A an array
     * @return total of reverse pairs
     */
    public long reversePairs(int[] A) {
        return mergeSort(A, 0, A.length - 1);
    }

    private int mergeSort(int[] A, int start, int end) {
        if (start >= end) {
            return 0;
        }

        int mid = (start + end) / 2;
        int sum = 0;
        sum += mergeSort(A, start, mid);
        sum += mergeSort(A, mid+1, end);
        sum += merge(A, start, mid, end);
        return sum;
    }

    private int merge(int[] A, int start, int mid, int end) {
        int[] temp = new int[A.length];
        int leftIndex = start;
        int rightIndex = mid + 1;
        int index = start;
        int sum = 0;

        while (leftIndex <= mid && rightIndex <= end) {
            if (A[leftIndex] <= A[rightIndex]) {
                temp[index++] = A[leftIndex++];
            } else {
                temp[index++] = A[rightIndex++];
                sum += mid - leftIndex + 1;
            }
        }
        while (leftIndex <= mid) {
            temp[index++] = A[leftIndex++];
        }
        while (rightIndex <= end) {
            temp[index++] = A[rightIndex++];
        }

        for (int i = start; i <= end; i++) {
            A[i] = temp[i];
        }

        return sum;
    }
}
```

👍 获赞 1

💬 2 条评论

 令狐冲

更新于 6/9/2020, 7:04:29 AM

利用归并排序的思想求逆序对, 复杂度 $O(n\log n)$  当然也可以用树状数组或者线段树求解

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
private:
    int* tmp;
public:
    /**
     * @param A an array
     * @return total of reverse pairs
     */
    long long reversePairs(vector<int>& A) {
        // Write your code here
        int n = A.size();
        tmp = new int[n];
        return mergeSort(A, 0, n-1);
    }

    long long merge(vector<int> &A, int l, int m, int r) {
        int i = l, j = m + 1, k = l;
        long long ans = 0;
        while (i <= m && j <= r) {
            if (A[i] > A[j]) {
                tmp[k++] = A[j++];
                ans += m - i + 1;
            } else {
                tmp[k++] = A[i++];
            }
        }
        while (i <= m) tmp[k++] = A[i++];
        while (j <= r) tmp[k++] = A[j++];
        for (i = l; i <= r; ++i)
            A[i] = tmp[i];
        return ans;
    }

    long long mergeSort(vector<int> &A, int l, int r) {
        long long ans = 0;
        if (l < r) {
            int m = (l + r) >> 1;
            ans += mergeSort(A, l, m);
            ans += mergeSort(A, m + 1, r);
            ans += merge(A, l, m, r);
        }
        return ans;
    }
};
```

👍 获赞 0

💬 添加评论

 E同学

更新于 6/9/2020, 7:03:47 AM

使用树状数组求解, 以A的权值作为树状数组的下标。通过离散化将A的数值锁定在A.length以内

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
// BIT
public class Solution {
    /**
     * @param A: an array
     * @return: total of reverse pairs
     */
    class node {
        public int index;
        public int val;
        public node(int index, int val) {
            this.index = index;
            this.val = val;
        }
    }

    public long reversePairs(int[] A) {
        // write your code here
        if (A == null || A.length == 0) {
            return (long) 0;
        }

        List<node> list = new ArrayList<>();
        for (int i = 0; i < A.length; i++) {
            list.add(new node(i, A[i]));
        }

        Collections.sort(list, new Comparator<node>() {
            @Override
            public int compare(node n1, node n2) {
                return n1.val - n2.val;
            }
        });

        //离散化
        A[list.get(0).index] = 0;
        for (int i = 1; i < A.length; i++) {
            if (list.get(i).val == list.get(i - 1).val) {
                A[list.get(i).index] = i - 1;
            } else {
                A[list.get(i).index] = i;
            }
        }

        long count = 0;
        int[] BIT = new int[A.length + 1];
        for (int i = 0; i < A.length; i++) {
            update(A[i], 1, BIT);
            count += getPrefixSum(A.length - 1, BIT) - getPrefixSum(A[i], BIT);
        }
        return count;
    }

    private int getPrefixSum(int index, int[] BIT) {
        int sum = 0;
        for (int i = index + 1; i > 0; i -= lowbit(i)) {
            sum += BIT[i];
        }
    }
}
```

```
    }  
    return sum;  
}  
  
private void update(int index, int val, int[] BIT) {  
    for (int i = index + 1; i < BIT.length; i += lowbit(i)) {  
        BIT[i] += val;  
    }  
}  
  
private int lowbit(int x) {  
    return x & (-x);  
}  
}
```

👍 获赞 4

💬 1 条评论



kevin

更新于 6/9/2020, 7:03:48 AM

归并排序是将数列 $a[l, h]$ 分成两半 $a[l, mid]$ 和 $a[mid+1, h]$ 分别进行归并排序, 然后再将这两半合并起来。在合并的过程中 (设 $l \leq i \leq mid$ ,  $mid+1 \leq j \leq h$ ) , 当 $a_i() \leq a_j()$ 时, 并不产生逆序数; 当 $a_i() > a_j()$ 时, 在前半部分中比 $a_i()$ 大的数都比 $a_j()$ 大, 将 $a_j()$ 放在 $a_i()$ 前面的话, 逆序数要加上 $mid+1-i$ 。因此, 可以在归并排序中的合并过程中计算逆序数。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param A: an array
    @return: total of reverse pairs
    """
    def reversePairs(self, A):
        # write your code here
        tmp = [0] * len(A)
        self.count = 0
        self.mergesort(A, 0, len(A) - 1, tmp)
        return self.count

    def mergesort(self, A, start, end, tmp):
        if start >= end:
            return

        mid = start + (end - start) // 2
        self.mergesort(A, start, mid, tmp)
        self.mergesort(A, mid + 1, end, tmp)
        self.merge(A, start, end, tmp)

    def merge(self, A, start, end, tmp):
        mid = start + (end - start) // 2
        i = start
        j = mid + 1
        index = start
        while i <= mid and j <= end:
            if A[i] > A[j]:
                tmp[index] = A[j]
                j += 1
                self.count += mid - i + 1
            else:
                tmp[index] = A[i]
                i += 1

            index += 1

        while i <= mid:
            tmp[index] = A[i]
            i += 1
            index += 1

        while j <= end:
            tmp[index] = A[j]
            j += 1
            index += 1

        for i in range(start, end + 1):
            A[i] = tmp[i]
```

👍 获赞 3

💬 1 条评论



lin

更新于 6/9/2020, 7:03:50 AM

使用归并排序, 因为归并排序时候后半段如果有元素先于前半段数并入数组中, 则说明当前比较的前半段 $i$ 到 $mid$ 都比它大, 则逆序对增量为  $mid - i + 1$ 。其实归并排序找逆序对的原理就是先从小部分比较找出逆序对, 然后在逐步扩大, 复杂度 $n \lg n$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class Solution {
    /**
     * @param A: an array
     * @return: total of reverse pairs
     */
    long res = 0;
    public long reversePairs(int[] A) {
        // write your code here
        mergeSort(A, 0, A.length - 1);
        return res;
    }

    private void mergeSort(int[] A, int start, int end){
        if(start >= end){ //至少有保证两个数才能归并
            return;
        }
        int mid = start + (end - start) / 2;
        mergeSort(A, start, mid);
        mergeSort(A, mid + 1, end);

        merge(A, start, mid, end);
    }

    private void merge(int[] A, int start, int mid, int end){
        int temp[] = new int[end - start + 1];

        int k = 0;
        int i = start;
        int j = mid + 1;

        while(i <= mid && j <= end){
            if(A[i] <= A[j]){
                temp[k] = A[i];
                i++;
                k++;
            }else{
                temp[k] = A[j];
                res += mid - i + 1;
                k++;
                j++;
            }
        }

        while(i <= mid){
            temp[k] = A[i];
            i++;
            k++;
        }
        while(j <= end){
            temp[k] = A[j];
            j++;
            k++;
        }
        for(int p = 0; p < temp.length; p++){
            A[start + p] = temp[p];
        }
    }
}
```



```
    } //别忘了把原数组给覆盖掉  
}  
}
```

👍 获赞 2      💬 添加评论



**kevin**

更新于 6/9/2020, 7:03:49 AM

使用binary indexed tree来做。具体请详见<https://www.jiuzhang.com/tutorial/binary-indexed-tree/297> (<https://www.jiuzhang.com/tutorial/binary-indexed-tree/297>)

这里引入一种新的树状数组建树机制，前面学到的树状数组，是基于下标而建立的，对于 $a_j()$ ，它的信息将更新 $c_j()$ 和 $c_j()$ 的祖先。 $c$ 数组的含义是一段区间的和。而这道题的树状数组，是基于权值建立的，对于 $a_j()$ ，它将更新 $c[a_j()]$ 和 $c[a_j()]$ 的祖先，每次加1（代表 $a_j()$ 这个权值的元素有1个）， $ci()$ 表示的是一段权值区间的元素个数。举个例子，2,4,1,3,5 ()这个序列，它的第四个数是3， $a_4() = 3$ ，那么我们将调用 $add(3,1)$ ，更新 $c_3()$ 以及 $c_3()$ 的祖先。当我们在求下标 $j$ ，在下标 $1 \sim j - 1$ 中有多少个数字大于 $a_j()$ 时，因为已经把 $1 \sim j-1$  这些元素的值添加进权值树状数组中了，我们求出此刻区间 $[a_j() + 1, MAX]$  ( $MAX$ 就是区间最大值，这里是100000) 的区间和，也就是在 $1 \sim j-1$ 中比 $a_j()$ 大的元素有多少个。这就是我们所求解的问题了。整个树状数组的框架、操作都没有发生变化，是 $ci()$ 所表示的逻辑发生了变化。即树状数组的下标变成了权值，而树状数组的权值代表着元素个数。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
import bisect

class Solution:
    """
    @param A: an array
    @return: total of reverse pairs
    """
    def reversePairs(self, A):
        # write your code here

        unique_A = sorted(list(set(A)))
        max_val = len(unique_A) - 1
        self.bit = [0] * (len(unique_A) + 1)

        result = 0
        for i in range(len(A)):
            A[i] = bisect.bisect_left(unique_A, A[i])
            result += self.getPresum(max_val) - self.getPresum(A[i])
            self.update(A[i], 1)

        return result

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

    def update(self, index, val):
        i = index + 1
        while i < len(self.bit):
            self.bit[i] += val
            i += self.lowbit(i)

    def getPresum(self, index):
        presum = 0
        i = index + 1
        while i > 0:
            presum += self.bit[i]
            i -= self.lowbit(i)

        return presum
```

👍 获赞 2

💬 1 条评论

**九章用户7XCFLI**

更新于 6/9/2020, 7:03:57 AM

leetcode解法

<https://leetcode.com/problems/reverse-pairs/description/> (https://leetcode.com/problems/reverse-pairs/description/)解法: merge sort 时间:  $O(n\log(n))$  空间:  $O(n)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
    public int reversePairs(int[] nums) {
        return mergeSort(nums, 0, nums.length - 1);
    }

    private int mergeSort(int[] nums, int l, int r) {
        if (l >= r) return 0;

        int m = l + (r - l) / 2;
        int cnt = mergeSort(nums, l, m) + mergeSort(nums, m + 1, r);
        int i, j;
        for (i = l, j = m + 1; i <= m; i++) {
            while (j <= r && nums[i] > nums[j] * 2L) {
                j++;
            }
            cnt += j - (m + 1);
        }
        merge(nums, l, m, r);
        return cnt;
    }

    private void merge(int[] nums, int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;
        int[] L = new int[n1];
        int[] R = new int[n2];
        for (int i = 0; i < n1; i++) {
            L[i] = nums[l + i];
        }
        for (int j = 0; j < n2; j++) {
            R[j] = nums[m + 1 + j];
        }
        int i = 0, j = 0, k = l;
        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                nums[k++] = L[i++];
            } else {
                nums[k++] = R[j++];
            }
        }
        while (i < n1) nums[k++] = L[i++];
        while (j < n2) nums[k++] = R[j++];
    }
}
```

👍 获赞 1    💬 添加评论



**Jimmy**

更新于 6/9/2020, 7:03:57 AM

解法也是通过归并排序, 写法相对简单一点, 可能并不是严格的 $O(n \log n)$ , 大家参考一下。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param A: an array
     * @return: total of reverse pairs
     */
    long long reversePairs(vector<int> &A) {
        // write your code here
        int len = A.size();
        if(len <= 1)
            return 0;
        vector<int> ve1, ve2;
        ve1.assign(A.begin(), A.begin()+len/2);
        ve2.assign(A.begin()+len/2, A.end());
        long long ans = 0;
        ans += reversePairs(ve1);
        ans += reversePairs(ve2);
        for(int i = 0; i < ve1.size(); i++)
        {
            ans += lower_bound(ve2.begin(), ve2.end(), ve1[i]) - ve2.begin();
        }
        sort(A.begin(), A.end());
        return ans;
    }
};
```

👍 获赞 1    💬 添加评论



N同学

更新于 6/27/2020, 9:10:31 PM

用双指针做, 两个for循环遍历整个数组。。用if做个比较。。几行就完事儿了。。

下

邀  
请

👤



我  
去

(/) 课程 (/course/) 旗舰课 (/premium-course/) 1对1私教 (/1on1/) 免费课 (/free-course/) 面试经验分享 (/interview-experience/) 成功上岸 (/success/) 更多... (/more/) 我的课程 (/accounts/profile/) (/accounts/)

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param A: an array
     * @return: total of reverse pairs
     */
    public long reversePairs(int[] A) {
        // write your code here
        if (A == null || A.length == 0) {
            return 0;
        }
        int count = 0;
        for (int i = 0; i < A.length; i++) {
            for (int j = i + 1; j < A.length; j++) {
                if (A[j] < A[i]) {
                    count++;
                }
            }
        }
        return count;
    }
}
```

👍 获赞 0    💬 2 条评论



九章用户DU9VH5

更新于 6/9/2020, 7:04:27 AM

可以遍历一次数组，利用二分查找降低时间复杂度至 $O(N\log N)$ ，空间复杂度仍为 $O(N)$



invitation/sh:



```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: A: an array
    @return: total of reverse pairs
    """
    def reversePairs(self,A):
        try:
            count = 0
            min = A[0]
            pre = []
            for i in range(len(A)):
                if A[i]<min:
                    count += len(pre)
                    min = A[i]
                    pre.append(A[i])
                else:
                    #find appropriat location
                    position = self.search2(pre,A[i])
                    count += position
                    pre.insert(position,A[i])
            return count
        except Exception as e:
            print (e)
            return 0

    def search2(self,list,target):
        if len(list)==0:
            return 0
        else:
            low = 0
            high = len(list)-1
            mid = int((low+high+1)/2)
            while low<=high:
                print (low,high,mid)
                if target>=list[mid]:
                    high = mid-1
                elif target<list[mid]:
                    low = mid+1
                mid = int((low+high+1)/2)
            return mid
```

👍 获赞 0

💬 2 条评论

**Adam57**

更新于 6/9/2020, 7:04:20 AM

use binary indexed tree,  $O(n \log n)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param A: an array
    @return: total of reverse pairs
    """
    def reversePairs(self, A):
        # write your code here
        if A is None or not A:
            return 0

        unique_A = list(set(A))
        max_v = len(unique_A)
        sorted_A = sorted(unique_A)

        self.bit = [0 for i in range(max_v + 1)]

        import bisect
        res = 0
        for num in A:
            rank = bisect.bisect_left(sorted_A, num) + 1
            # print('max_v:{} rank:{}'.format(max_v, rank))
            res += self.prefix_sum(max_v) - self.prefix_sum(rank)
            self.add(rank, 1)
        return res

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

    def prefix_sum(self, x):
        i = x
        res = 0
        while i > 0:
            # print('i:{}'.format(i))
            res += self.bit[i]
            i -= self.lowbits(i)
        return res

    def add(self, x, v):
        i = x
        while i < len(self.bit):
            self.bit[i] += v
            i += self.lowbits(i)
```

👍 获赞 0

💬 添加评论



九章用户SM7RY4

更新于 6/9/2020, 7:04:18 AM

使用merge sort的思想做的。代码是python3的代码。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param A: an array
    @return: total of reverse pairs
    """
    def reversePairs(self, A):
        # write your code here
        temp = [0] * len(A)
        return self.mergeSort(A, temp, 0, len(A) - 1)

    def mergeSort(self, A, temp, start, end):
        if start >= end:
            return 0

        sum = 0
        mid = (start + end) // 2

        sum += self.mergeSort(A, temp, start, mid)
        sum += self.mergeSort(A, temp, mid + 1, end)
        sum += self.merge(A, temp, start, mid, end)

        return sum

    def merge(self, A, temp, start, mid, end):
        left, right = start, mid + 1
        index = start
        sum = 0

        while left <= mid and right <= end:
            if A[left] <= A[right]:
                temp[index] = A[left]
                left += 1
                index += 1
            else:
                temp[index] = A[right]
                right += 1
                index += 1
                sum += mid - left + 1

        while left <= mid:
            temp[index] = A[left]
            index += 1
            left += 1

        while right <= end:
            temp[index] = A[right]
            index += 1
            right += 1

        for i in range(start, end + 1):
            A[i] = temp[i]

        return sum
```

👍 获赞 0    💬 添加评论



ch

更新于 6/9/2020, 7:04:10 AM

binary index tree version , 基本上就是loop through Ai (), 然後同時把Ai () value update進binary index tree, 每次看過去有多少數值大於Ai ().

```

/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class BITree {
private:
    vector<int> bits;
public:
    BITree(int size) {
        bits.resize(size+1, 0);
    }
    int query(int x) {
        x++;
        int cnt = 0;
        while (x) {
            cnt += bits[x];
            x -= lowBit(x);
        }
        return cnt;
    }
    int update(int x, int delta) {
        x++;
        while (x < bits.size()) {
            bits[x] += delta;
            x += lowBit(x);
        }
    }
    int lowBit(int x) {
        return x & ~(x-1);
    }
};

class Solution {
public:
    /**
     * @param A: an array
     * @return: total of reverse pairs
     */
    long long reversePairs(vector<int> &A) {
        if (!A.size()) {
            return 0;
        }
        int maxVal = getMax(A);
        int minVal = getMin(A);
        BITree biTree(maxVal-minVal+1);
        int ans = 0;
        for (int n: A) {
            ans += biTree.query(maxVal-minVal) - biTree.query(n-minVal);
            biTree.update(n-minVal, 1);
        }
        return ans;
    }

    int getMax(vector<int> &A) {
        int maxVal = INT_MIN;

```

```

        for (int n: A) {
            maxVal = max(maxVal, n);
        }
        return maxVal;
    }
    int getMin(vector<int> &A) {
        int minVal = INT_MAX;
        for (int n: A) {
            minVal = min(minVal, n);
        }
        return minVal;
    }
};

```

👍 获赞 0    💬 添加评论



九章用户J9PB42

更新于 6/9/2020, 7:04:08 AM

树状数组能解, 线段树也可以。

首先进行离散化。

线段树节点count,表示[start-end]数字出现的总次数, 边插入边更新。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param A: an array
     * @return: total of reverse pairs
     */
    /**
     * 线段树解决
     */
    public long reversePairs(int[] A) {
        if(A==null || A.length==0){
            return 0;
        }
        //离散化
        discrete(A);
        int res = 0;
        SegementTree root = build(A,0,A.length-1);
        for (int i = 0; i < A.length; i++) {
            update(root,A[i],1);
            /**
             * 插入过程中, 右边的就是比自己大, 且早插入的
             */
            res += (root.count-query(root,0,A[i]));
        }
        return res;
    }

    /**
     * 离散化, 替换前, 替换后, 任意位置的相对大小不变。对替换后的数组进行求解答案就是原本数组的解。
     * @param A
     */
    private void discrete(int[] A) {

```

```
Pair[] pairs = new Pair[A.length];
for (int i = 0; i < A.length; i++) {
    pairs[i] = new Pair(i,A[i]);
}
//将原数组排序, 同时记下他们原先在a数组中的下标
Arrays.sort(pairs, new Comparator<Pair>() {
    @Override
    public int compare(Pair o1, Pair o2) {
        return o1.val-o2.val;
    }
});
//
for (int i = 0; i < pairs.length; i++) {
    /**
     * 排序之后, 将他们的排名作为他们的新值对原来下标的权值进行替换。这里要注意权值相等的情况。
     */
    A[pairs[i].index] = i;
}
}

public class Pair{
    int index,val;

    public Pair(int index, int val) {
        this.index = index;
        this.val = val;
    }
}

public class SegementTree{
    public int start,end,count;
    public SegementTree left,right;

    public SegementTree(int start, int end, int count) {
        this.start = start;
        this.end = end;
        this.count = count;
    }
}

public SegementTree build(int[] A,int start,int end){
    if(start>end){
        return null;
    }
    if(end==start){
        return new SegementTree(start,end,0);
    }
    SegementTree root = new SegementTree(start,end,0);
    int mid = start+(end-start)/2;
    root.left = build(A,start,mid);
    root.right = build(A,mid+1,end);
    return root;
}

public void update(SegementTree root,int index,int val){
    if(root==null){
        return;
    }
    if(root.start==index && root.end==index){
        root.count += 1;
        return;
    }
    int mid = (root.start+root.end)/2;
    if(mid>=index){
        update(root.left,index,val);
    }else{
        update(root.right,index,val);
    }
}
```

```

    }
    root.count = root.left.count + root.right.count;
}

public int query(SegmentTree root,int start,int end){
    if(start<=root.start && root.end<=end){
        return root.count;
    }
    int ans = 0;
    int mid = root.start+(root.end-root.start)/2;
    if(start<=mid){
        ans += query(root.left,start,end);
    }
    if(end>=mid+1){
        ans += query(root.right,start,end);
    }
    return ans;
}
}

```

👍 获赞 0    💬 添加评论



**num3ers**

更新于 6/9/2020, 7:04:05 AM

use binary index tree to solve this problem. my method different part from others in the following: 1. no need to check whether there's any duplicates in the array for instant: 4,0,0,0 if we set three 0s to 1, then the new array is 2, 1, 1, 1; reverse pairs is 3 but if we set their val after hash function by the order, the new array is 4, 1, 2, 3; reverse pairs # is still 4. In reverse pairs, we only care about ordering, not val; after sorting, if they are behind some #, just make sure set them greater val than the fronts

2. can get prefix sum, no need to insert elements first; just for loop from end to begin, it means find some #, which is less than me, because I am closing to the front.

```

/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    private int n;
    private int[] C;

    private int lowbit(int x) {
        return x & -x;
    }

    private void add(int index, int val) {
        for (int i = index; i <= n; i += lowbit(i)) {
            C[index] += val;
        }
    }

    private long prefixSum(int x) {
        long res = 0;
        for (int i = x; i > 0; i -= lowbit(i)) {
            res += C[i];
        }
        return res;
    }
}

```

```
private class Node {
    int val;
    int index;
    Node(int val, int index) {
        this.val = val;
        this.index = index;
    }
}

public long reversePairs(int[] A) {
    if (A == null || A.length == 0) {
        return 0;
    }
    n = A.length;
    List<Node> listA = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        listA.add(new Node(A[i], i));
    }

    Collections.sort(listA, new Comparator<Node>() {
        @Override
        public int compare(Node a, Node b) {
            return a.val - b.val;
        }
    });

    for (int i = 0; i < n; i++) {
        A[listA.get(i).index] = i + 1;
    }

    C = new int[n + 1];
    long res = 0;
    for (int i = n - 1; i >= 0; i--) {
        res += prefixSum(A[i]);
        add(A[i], 1);
    }
    return res;
}
```

👍 获赞 0    💬 添加评论



杨TONGXUE

更新于 6/9/2020, 7:04:03 AM

借鉴 1297-count-of-smaller-numbers-after-self 思路 分别求每一位的右边有多少小于自己, 然后求和 使用的是binary index tree

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param A: an array
    @return: total of reverse pairs
    """
    def reversePairs(self, A):
        # write your code here
        res = 0
        rank = {val:key for key, val in enumerate(sorted(A))}
        self.N = len(A)
        self.BITree = [0] * (self.N + 1)
        dic = {}
        for val in A[::-1]:
            key = rank[val]
            dic[key] = dic.get(key, 0)
            res += self.sum(key) - dic[key]
            dic[key] += 1
            self.add(key)
        return res

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

    def sum(self, i):
        i += 1
        res = 0
        while i > 0:
            res += self.BITree[i]
            i -= self.lowbit(i)
        return res

    def add(self, i):
        i += 1
        while i <= self.N:
            self.BITree[i] += 1
            i += self.lowbit(i)
```

👍 获赞 0

💬 添加评论

## 进阶课程

视频+互动	直播+互动	直播+互动	互动课
<div>九章算法班 2021 版</div> <div>8周时间精通 57 个核心高频考点，9招击破 FLAG、BATJ 算法面试。22....</div>	<div>系统架构设计 System Design 2021 版</div> <div>成为百万架构师必上。30 课时带你快速掌握18大系统架构设计知识点与面...</div>	<div>九章算法面试高频题冲刺班</div> <div>每期更新 15% 题目，考前押题，一举拿下FLAG &amp; BATJ Offer</div>	<div>面向对象设计 OOD</div> <div>应届生及亚马逊面试必考，IT求职必备基础</div>