

# 每日两道算法题总结

---

## 每日两道算法题总结

0519

83. 删除排序链表中的重复元素（简单）

912使用递归及非递归两种方式实现快速排序(中等)

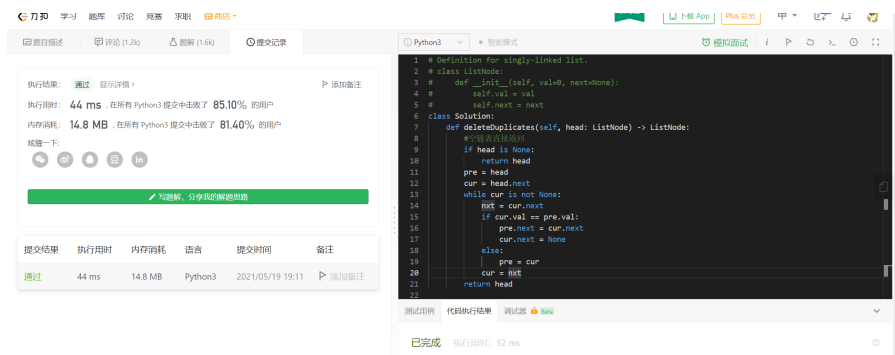
递归方法

## 0519

### 83. 删除排序链表中的重复元素（简单）

- 存在一个按升序排列的链表，给你这个链表的头节点 head，请你删除所有重复的元素，使每个元素 只出现一次。

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def deleteDuplicates(self, head: ListNode) -> ListNode:
        #空链表直接返回
        if head is None:
            return head
        pre = head
        cur = head.next
        ##时间O(N) 空间O (1)
        while cur is not None:
            nxt = cur.next
            if cur.val == pre.val:
                pre.next = cur.next
                cur.next = None
            else:
                pre = cur
            cur = nxt
        return head
```



## 912使用递归及非递归两种方式实现快速排序(中等)

### 力扣912排序

给你一个整数数组 `nums`，请你将该数组升序排列。

示例 1：

输入：nums = [5,2,3,1] 输出：[1,2,3,5] 示例 2：

输入：nums = [5,1,1,2,0,0] 输出：[0,0,1,1,2,5]

### 需要使用快速排序

提交代码，两个版本，一个pivot是由第一个元素替代，另一版本中pivot是random生成的，第一版本会报出时间超限

### 递归方法

```
class Solution:
    def sortArray(self, nums: List[int]) -> List[int]:
        def randomized_partition(self, nums, l, r):
            # pivot = l
            pivot = random.randint(l, r-1)
            ###生成不到r左开右闭
            nums[pivot], nums[r-1] = nums[r-1], nums[pivot]
            i = l
            for j in range(l, r-1):
                if nums[j] < nums[r-1]:
                    nums[j], nums[i] = nums[i], nums[j]
                    i += 1
            nums[i], nums[r-1] = nums[r-1], nums[i]
            return i

        def randomized_quicksort(self, nums, l, r):
            if r - l <= 1:
                return
            mid = randomized_partition(self, nums, l, r)
```

```
mid = randomized_partition(self,nums, l, r)
# print(mid,nums)
randomized_quicksort(self,nums, l, mid)
randomized_quicksort(self,nums, mid + 1, r)
```

# 时间复杂度 $O(n\log n)$  空间复杂度 $O(1)$

```
randomized_quicksort(self,nums,0,len(nums))
return nums
```

题目描述

讨论 (520)

题解 (824)

提交记录

执行结果: 通过 [显示详情](#)

执行用时: 576 ms, 在所有 Python3 提交中击败了 12.60% 的用户

内存消耗: 20.2 MB, 在所有 Python3 提交中击败了 16.99% 的用户

查看一下

[与题解](#) [分享我的解题思路](#)

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	576 ms	20.2 MB	Python3	2021/05/19 21:20	<a href="#">添加备注</a>
超出时间限制	N/A	N/A	Python3	2021/05/19 21:15	<a href="#">添加备注</a>
通过	544 ms	20.2 MB	Python3	2021/05/19 21:14	<a href="#">用random.randint</a>
超出时间限制	N/A	N/A	Python3	2021/05/19 20:41	<a href="#">超出时间限制</a>

Python3

智能模式

模拟面试

1 class Solution:

2 def sortArray(self, nums: List[int]) -> List[int]:

3 def randomized\_partition(self, nums, l, r):

4 # pivot = l

5 pivot = random.randint(l, r-1)

6 # 随机选取一个元素

7 nums[pivot], nums[r-1] = nums[r-1], nums[pivot]

8 i = l

9 for j in range(l, r-1):

10 if nums[j] < nums[r-1]:

11 nums[i], nums[j] = nums[j], nums[i]

12 i += 1

13 nums[i], nums[r-1] = nums[r-1], nums[i]

14 return i

15

16 def randomized\_quicksort(self, nums, l, r):

17 if r - l <= 1:

18 return

19 mid = randomized\_partition(self,nums, l, r)

20 # print(mid,nums)

21 randomized\_quicksort(self,nums, l, mid)

22 randomized\_quicksort(self,nums, mid + 1, r)

23

24 # 时间复杂度O(nlogn) 空间复杂度O(1)

25

26 randomized\_quicksort(self,nums,0,len(nums))

27 return nums