Assignment #5: 链表、栈、队列和归并排序

Updated 1348 GMT+8 Mar 17, 2025

2025 spring, Complied by **袁奕 2400010766 数院**

> **说明：**
>
> 1. **解题与记录：**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。
>
> 3. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

LC21.合并两个有序链表

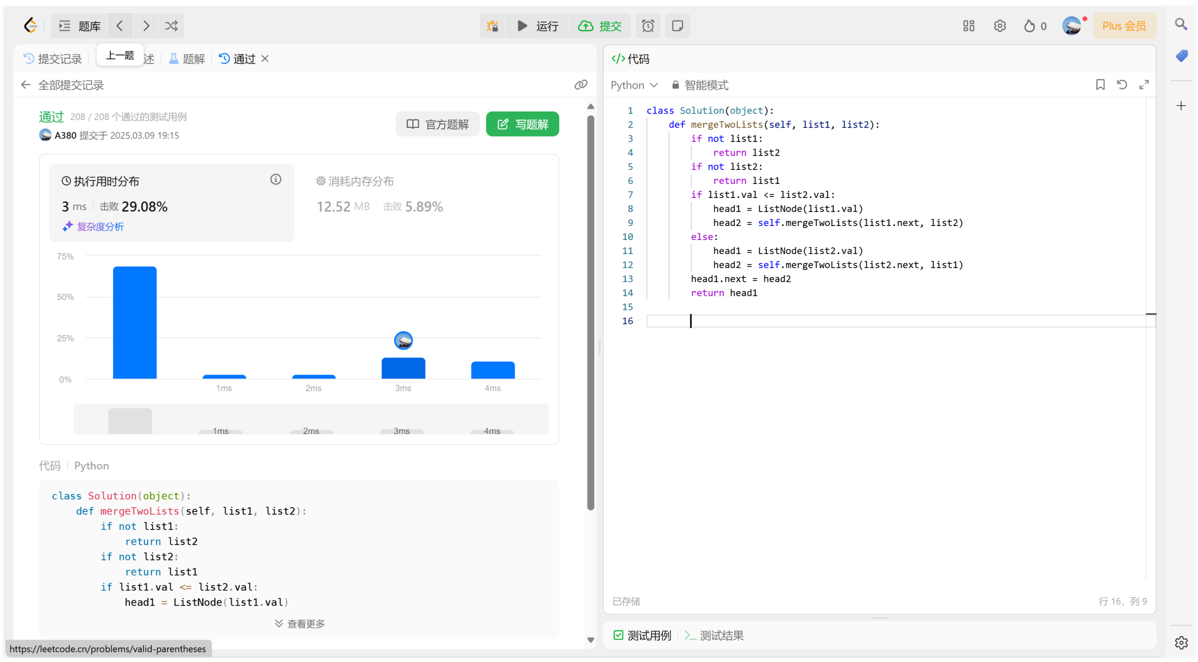
linked list, https://leetcode.cn/problems/merge-two-sorted-lists/

思路：

代码：

```python
class Solution(object):
    def mergeTwoLists(self, list1, list2):
        if not list1:
            return list2
        if not list2:
            return list1
        if list1.val <= list2.val:
            head1 = ListNode(list1.val)
            head2 = self.mergeTwoLists(list1.next, list2)
        else:
            head1 = ListNode(list2.val)
            head2 = self.mergeTwoLists(list2.next, list1)
        head1.next = head2
        return head1
```

21. 合并两个有序链表 - 力扣（LeetCode）



通过 208 / 208 个通过的测试用例

A380 提交于 2025.03.09 19:15

3 ms 击败 29.08%

12.52 MB 击败 5.89%

代码 | Python

```python
class Solution(object):
    def mergeTwoLists(self, list1, list2):
        if not list1:
            return list2
        if not list2:
            return list1
        if list1.val <= list2.val:
            head1 = ListNode(list1.val)
```

查看更多

https://leetcode.cn/problems/valid-parentheses

```python
class Solution(object):
    def mergeTwoLists(self, list1, list2):
        if not list1:
            return list2
        if not list2:
            return list1
        if list1.val <= list2.val:
            head1 = ListNode(list1.val)
            head2 = self.mergeTwoLists(list1.next, list2)
        else:
            head1 = ListNode(list2.val)
            head2 = self.mergeTwoLists(list2.next, list1)
        head1.next = head2
        return head1

```

LC234.回文链表

linked list, https://leetcode.cn/problems/palindrome-linked-list/

请用快慢指针实现。

询问 ChatGPT, 快慢指针 : Floyd's Tortoise and Hare Algorithm

Hare（兔子，快指针） 每次走 **两步**

Tortoise（乌龟，慢指针） 每次走 **一步**

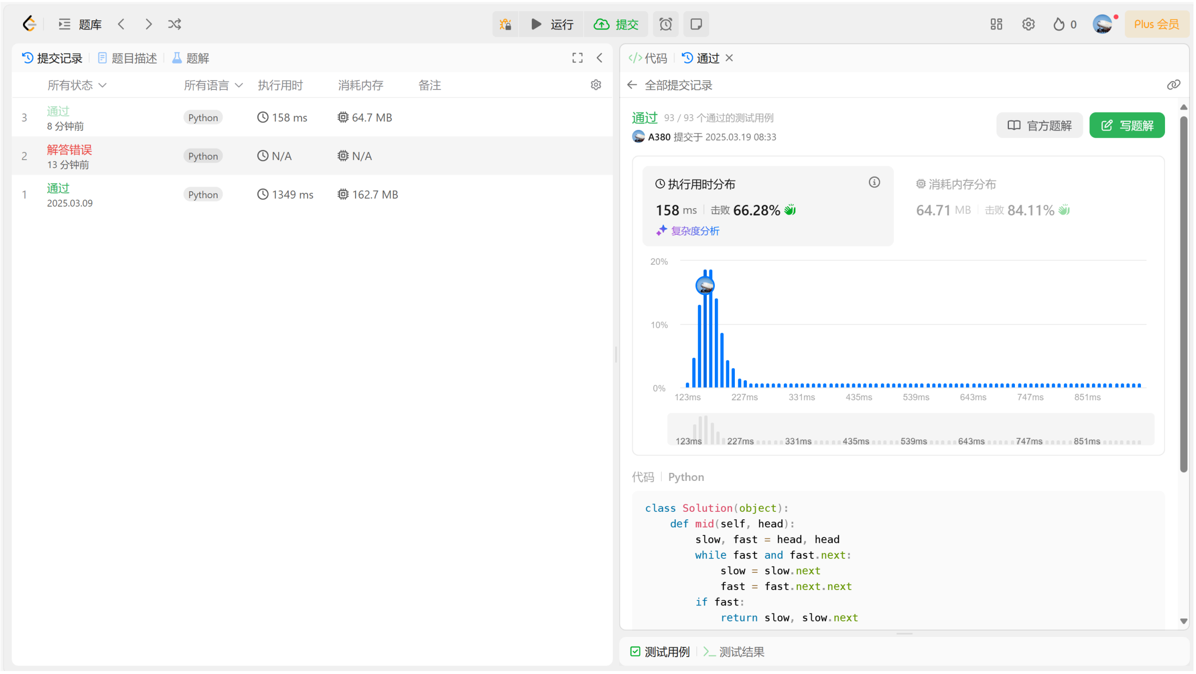
🐇走完全程, 🐢走完一半, 可以用来寻找中点.

以及上次作业的判断是否有环也是这样.

```python
class Solution(object):
    def mid(self, head):
        slow, fast = head, head
        while fast and fast.next:
            slow = slow.next
            fast = fast.next.next
        if fast:
            return slow, slow.next
        else:
            return slow, slow
    def reverse(self, head):
        pre = None
        curr = head
        while curr:
            curr_next = curr.next
            curr.next = pre
            pre = curr
            curr = curr_next
        return pre
    def isPalindrome(self, head):
        mid0, mid1 = self.mid(head)
        head1 = self.reverse(mid1)

        while head1:
            if head1.val != head.val:
                return False
            head1 = head1.next
            head = head.next

        if not head.next or not head.next.next:
            return True
        return False
```



题库　运行　提交　Plus 会员

提交记录　题目描述　题解

	所有状态	所有语言	执行用时	消耗内存	备注
3	通过 8 分钟前	Python	158 ms	64.7 MB	
2	解答错误 13 分钟前	Python	N/A	N/A	
1	通过 2025.03.09	Python	1349 ms	162.7 MB	

代码　通过

全部提交记录

通过 93 / 93 个通过的测试用例
A380 提交于 2025.03.19 08:33

官方题解　写题解

执行用时分布
158 ms | 击败 66.28%
复杂度分析

消耗内存分布
64.71 MB | 击败 84.11%

代码 | Python

```python
class Solution(object):
    def mid(self, head):
        slow, fast = head, head
        while fast and fast.next:
            slow = slow.next
            fast = fast.next.next
        if fast:
            return slow, slow.next
```

测试用例　测试结果

LC1472.设计浏览器历史记录

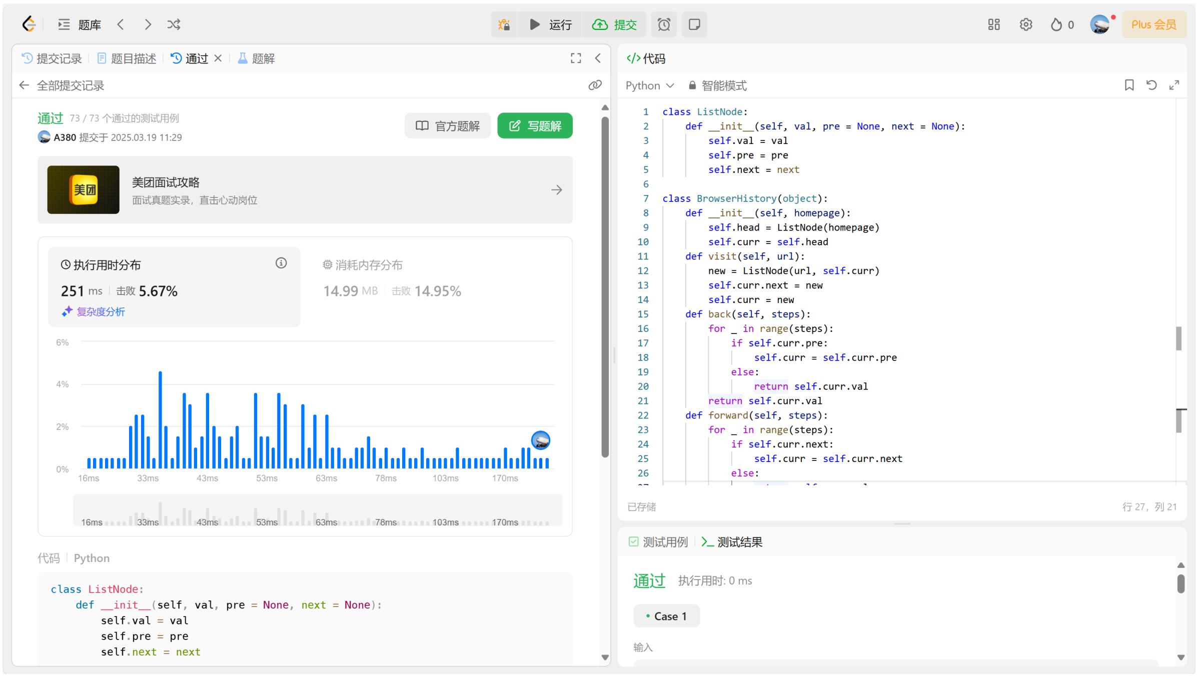
doubly-lined list, https://leetcode.cn/problems/design-browser-history/

请用双链表实现。

代码:

```python
class ListNode:
    def __init__(self, val, pre = None, next = None):
        self.val = val
        self.pre = pre
        self.next = next

class BrowserHistory(object):
    def __init__(self, homepage):
        self.head = ListNode(homepage)
        self.curr = self.head
    def visit(self, url):
        new = ListNode(url, self.curr)
        self.curr.next = new
        self.curr = new
    def back(self, steps):
        for _ in range(steps):
            if self.curr.pre:
                self.curr = self.curr.pre
            else:
                return self.curr.val
        return self.curr.val
    def forward(self, steps):
        for _ in range(steps):
            if self.curr.next:
                self.curr = self.curr.next
            else:
                return self.curr.val
        return self.curr.val
```



通过 73 / 73 个通过的测试用例

A380 提交于 2025.03.19 11:29

执行用时分布 251 ms | 击败 5.67%

消耗内存分布 14.99 MB | 击败 14.95%

代码 | Python

```python
class ListNode:
    def __init__(self, val, pre = None, next = None):
        self.val = val
        self.pre = pre
        self.next = next
```

Python

```python
class ListNode:
    def __init__(self, val, pre = None, next = None):
        self.val = val
        self.pre = pre
        self.next = next

class BrowserHistory(object):
    def __init__(self, homepage):
        self.head = ListNode(homepage)
        self.curr = self.head
    def visit(self, url):
        new = ListNode(url, self.curr)
        self.curr.next = new
        self.curr = new
    def back(self, steps):
        for _ in range(steps):
            if self.curr.pre:
                self.curr = self.curr.pre
            else:
                return self.curr.val
        return self.curr.val
    def forward(self, steps):
        for _ in range(steps):
            if self.curr.next:
                self.curr = self.curr.next
            else:
```

已存储 行 27，列 21

通过 执行用时: 0 ms

Case 1

输入

24591: 中序表达式转后序表达式

stack, http://cs101.openjudge.cn/practice/24591/

思路：

代码：

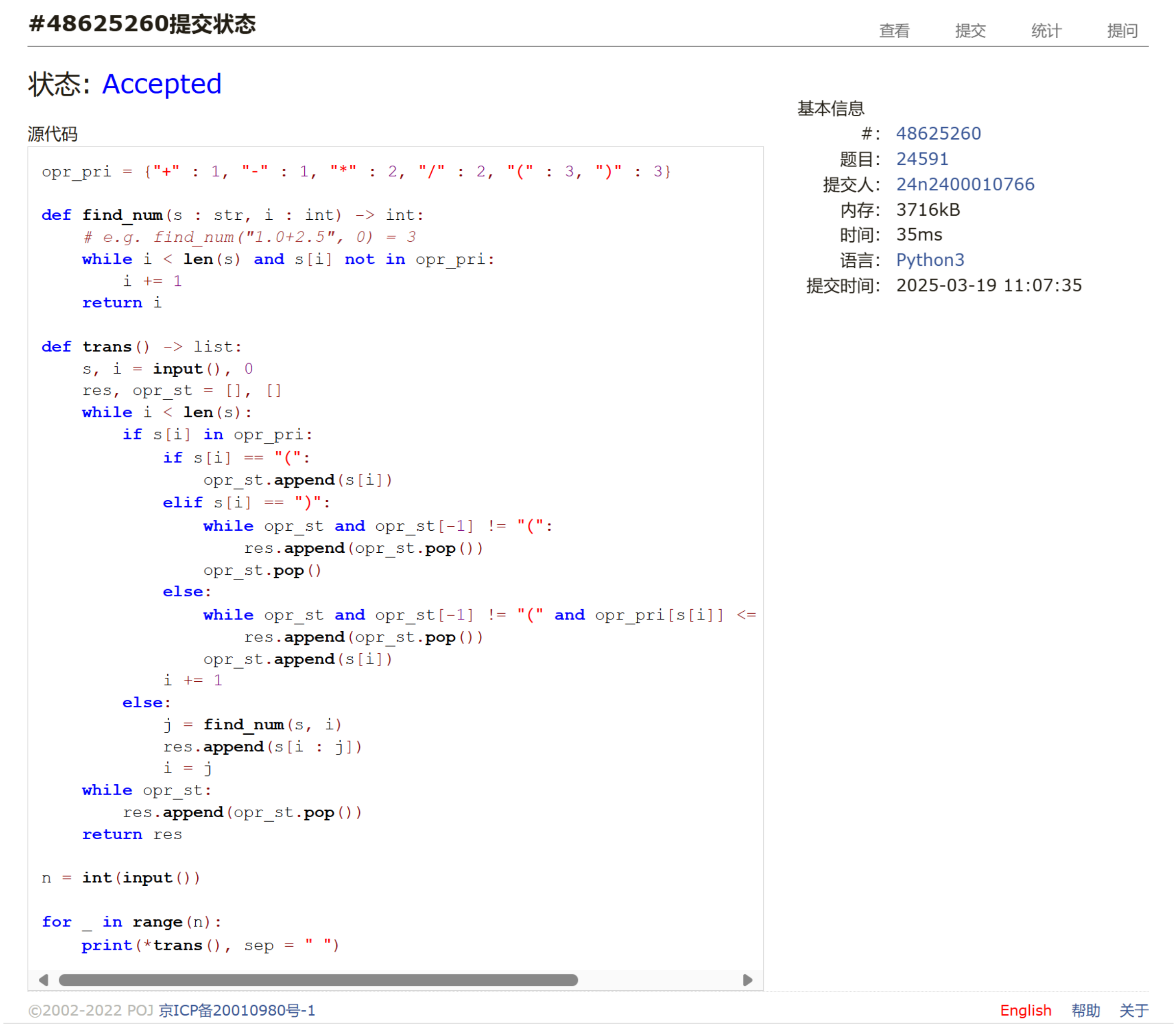
```python
opr_pri = {"+" : 1, "-" : 1, "*" : 2, "/" : 2, "(" : 3, ")" : 3}

def find_num(s : str, i : int) -> int:
    # e.g. find_num("1.0+2.5", 0) = 3
    while i < len(s) and s[i] not in opr_pri:
        i += 1
    return i

def trans() -> list:
    s, i = input(), 0
    res, opr_st = [], []
    while i < len(s):
        if s[i] in opr_pri:
            if s[i] == "(":
                opr_st.append(s[i])
            elif s[i] == ")":
                while opr_st and opr_st[-1] != "(":
                    res.append(opr_st.pop())
                opr_st.pop()
            else:
                while opr_st and opr_st[-1] != "(" and opr_pri[s[i]] <=
opr_pri[opr_st[-1]]:
                    res.append(opr_st.pop())
                opr_st.append(s[i])
            i += 1
        else:
            j = find_num(s, i)
            res.append(s[i : j])
            i = j
    while opr_st:
        res.append(opr_st.pop())
    return res

n = int(input())

for _ in range(n):
    print(*trans(), sep = " ")
```



状态: Accepted

源代码

```python
opr_pri = {"+" : 1, "-" : 1, "*" : 2, "/" : 2, "(" : 3, ")" : 3}

def find_num(s : str, i : int) -> int:
    # e.g. find_num("1.0+2.5", 0) = 3
    while i < len(s) and s[i] not in opr_pri:
        i += 1
    return i

def trans() -> list:
    s, i = input(), 0
    res, opr_st = [], []
    while i < len(s):
        if s[i] in opr_pri:
            if s[i] == "(":
                opr_st.append(s[i])
            elif s[i] == ")":
                while opr_st and opr_st[-1] != "(":
                    res.append(opr_st.pop())
                opr_st.pop()
            else:
                while opr_st and opr_st[-1] != "(" and opr_pri[s[i]] <=
                    res.append(opr_st.pop())
                opr_st.append(s[i])
            i += 1
        else:
            j = find_num(s, i)
            res.append(s[i : j])
            i = j
    while opr_st:
        res.append(opr_st.pop())
    return res

n = int(input())

for _ in range(n):
    print(*trans(), sep = " ")
```

基本信息

#:	48625260
题目:	24591
提交人:	24n2400010766
内存:	3716kB
时间:	35ms
语言:	Python3
提交时间:	2025-03-19 11:07:35

©2002-2022 POJ 京ICP备20010980号-1

03253: 约瑟夫问题No.2

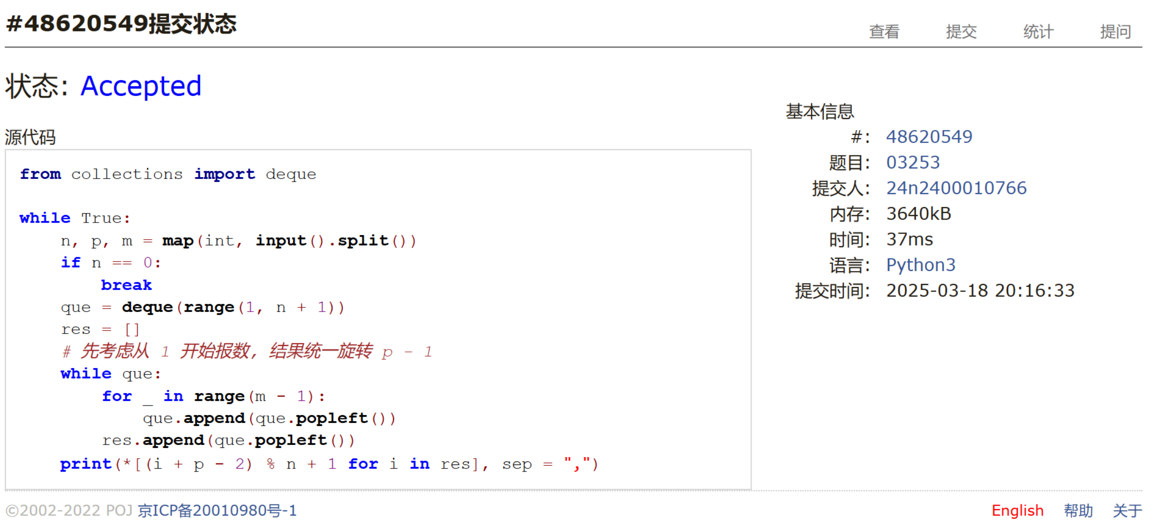
queue, http://cs101.openjudge.cn/practice/03253/

请用队列实现。

代码:

```python
from collections import deque

while True:
    n, p, m = map(int, input().split())
    if n == 0:
        break
    que = deque(range(1, n + 1))
    res = []
    # 先考虑从 1 开始报数, 结果统一旋转 p - 1
    while que:
        for _ in range(m - 1):
            que.append(que.popleft())
        res.append(que.popleft())
    print(*[(i + p - 2) % n + 1 for i in res], sep = ",")
```


#48620549提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from collections import deque

while True:
    n, p, m = map(int, input().split())
    if n == 0:
        break
    que = deque(range(1, n + 1))
    res = []
    # 先考虑从 1 开始报数，结果统一旋转 p - 1
    while que:
        for _ in range(m - 1):
            que.append(que.popleft())
        res.append(que.popleft())
    print(*[(i + p - 2) % n + 1 for i in res], sep = ",")
```

基本信息

#:	48620549
题目:	03253
提交人:	24n2400010766
内存:	3640kB
时间:	37ms
语言:	Python3
提交时间:	2025-03-18 20:16:33

©2002-2022 POJ 京ICP备20010980号-1　English 帮助 关于

20018: 蚂蚁王国的越野跑

merge sort, http://cs101.openjudge.cn/practice/20018/

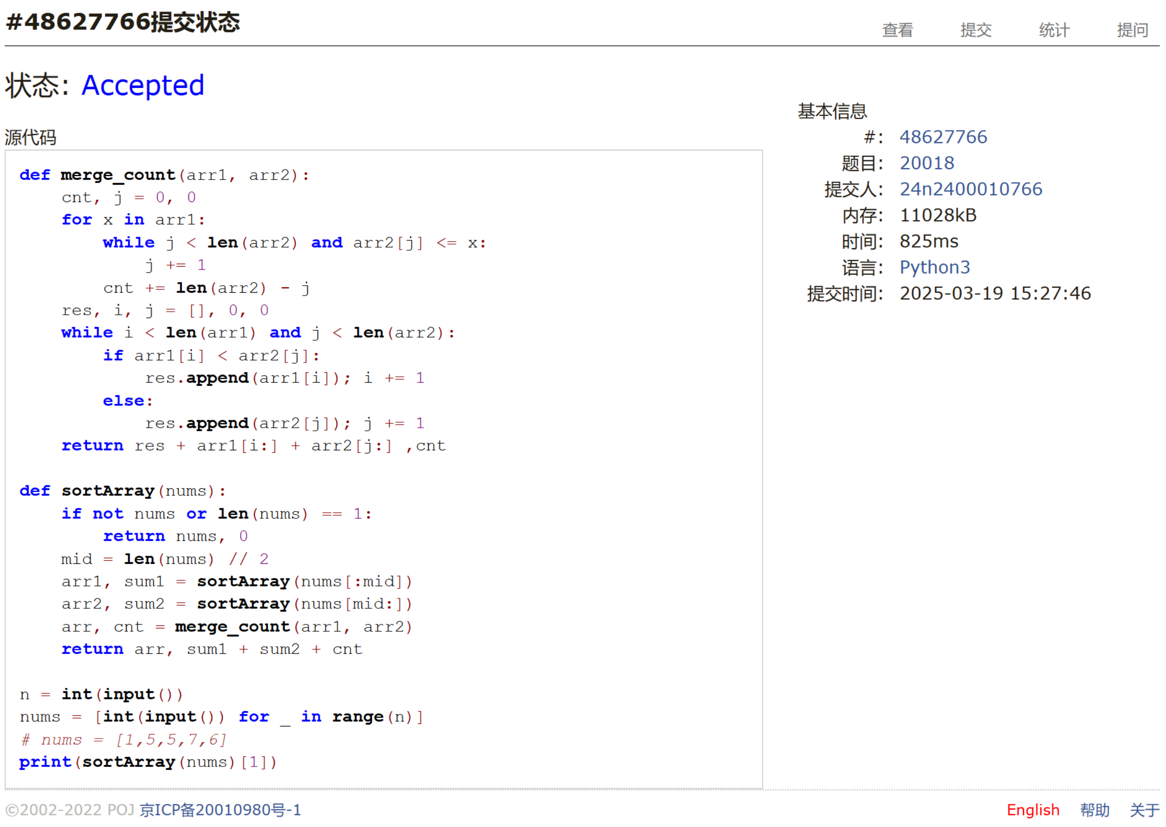
思路：本质求逆序对

代码：

```python
def merge_count(arr1, arr2):
    cnt, j = 0, 0
    for x in arr1:
        while j < len(arr2) and arr2[j] <= x:
            j += 1
        cnt += len(arr2) - j
    res, i, j = [], 0, 0
    while i < len(arr1) and j < len(arr2):
        if arr1[i] < arr2[j]:
            res.append(arr1[i]); i += 1
        else:
            res.append(arr2[j]); j += 1
    return res + arr1[i:] + arr2[j:] ,cnt

def sortArray(nums):
    if not nums or len(nums) == 1:
        return nums, 0
    mid = len(nums) // 2
    arr1, sum1 = sortArray(nums[:mid])
    arr2, sum2 = sortArray(nums[mid:])
    arr, cnt = merge_count(arr1, arr2)
    return arr, sum1 + sum2 + cnt

n = int(input())
nums = [int(input()) for _ in range(n)]
print(sortArray(nums)[1])
```



#48627766提交状态

状态: Accepted

源代码

```python
def merge_count(arr1, arr2):
    cnt, j = 0, 0
    for x in arr1:
        while j < len(arr2) and arr2[j] <= x:
            j += 1
        cnt += len(arr2) - j
    res, i, j = [], 0, 0
    while i < len(arr1) and j < len(arr2):
        if arr1[i] < arr2[j]:
            res.append(arr1[i]); i += 1
        else:
            res.append(arr2[j]); j += 1
    return res + arr1[i:] + arr2[j:] ,cnt

def sortArray(nums):
    if not nums or len(nums) == 1:
        return nums, 0
    mid = len(nums) // 2
    arr1, sum1 = sortArray(nums[:mid])
    arr2, sum2 = sortArray(nums[mid:])
    arr, cnt = merge_count(arr1, arr2)
    return arr, sum1 + sum2 + cnt

n = int(input())
nums = [int(input()) for _ in range(n)]
# nums = [1,5,5,7,6]
print(sortArray(nums)[1])
```

基本信息

#:	48627766
题目:	20018
提交人:	24n2400010766
内存:	11028kB
时间:	825ms
语言:	Python3
提交时间:	2025-03-19 15:27:46

©2002-2022 POJ 京ICP备20010980号-1 English 帮助 关于

2. 学习总结和收获

总结了链表引用与赋值的cheating sheet

```python
# 定义链表节点类
class ListNode:
    def __init__(self, val, next = None):
        self.val = val
        self.next = next
    def __str__(self):
        return f"ListNode({self.val} -> {self.next.val})"

d = ListNode(4)
c = ListNode(3, d)
b = ListNode(2, c)
a = ListNode(1, b)
```

1.
   ```python
   # Example 1 : `prev` 和 `curr` 指向相同的节点，修改 `prev` 后 `curr` 不受影响
   prev = a
   curr = prev
   prev = b
   print(curr == a, a) # output : True ListNode(1 -> 2)
   ```

2.
   ```python
   # Example 2 : `curr` 指向 `a.next` (i.e. `b`)，修改 `prev` 后 `curr` 不受影响
   prev = a
   curr = prev.next
   prev = c
   print(curr == b, b) # output : True ListNode(2 -> 3)
   ```

3.
```python
# Example 3 : `curr` 指向 `a`, 修改 `a.val`, `curr.val` 也受影响
curr = a
a.val = 0
print(curr) # output : ListNode(0 -> 2)
```

4.
```python
# Example 4 : `prev` 和 `curr` 指向相同对象 `a`, 修改 `prev.val`, `curr.val` 也受影响
prev = a
curr = a
prev.val = 0
print(curr) # output : ListNode(0 -> 2)
```

5.
```python
# Example 5 : `curr` 指向 `a`, 修改 `a.next`, `curr.next` 也受影响
prev = a
curr = prev
prev.next = c
print(curr) # output : ListNode(0 -> 3)
```

引用变更不会同步, 赋值变更 (`prev.next = ...` 或者 `prev.val = ...`) 会同步

做的比较有意义 ~~(困难的)~~ 题目 :

25. K 个一组翻转链表 - 力扣（LeetCode）

```python
class Solution(object):
    def len(self, node, k):
        # e.g. k = 2, 1 -> 2 -> 3 -> 4, len(3) = 2
        if not node:
            return False
        for i in range(k - 1):
            if not node.next:
                return False
            node = node.next
        return True
    def reverse_next_k(self, head, k):
        #e.g. 1 -> 2 -> 3 -> 4 -> 5, k = 3, return 3, 1
        #e.g. 1 -> 2, k = 3, return None None
        if not self.len(head, k):
            return None, None
        pre = None
        curr = head
        cnt = 1
        while curr and cnt <= k:
            curr_next = curr.next
            curr.next = pre
            pre = curr
            curr = curr_next
            cnt += 1
        head.next = curr
        return pre, head

    def reverseKGroup(self, head, k):
        start, end = self.reverse_next_k(head, k)
        head = start
        while end and end.next:
            new_start, new_end = self.reverse_next_k(end.next, k)
            if new_start:
                end.next = new_start
                start, end = new_start, new_end
            else:
                break
        return head
```

295. 数据流的中位数 - 力扣（LeetCode）

```python
from heapq import heappush, heappop

class Heap:
    def __init__(self, is_max = True):
        self.hp = []
        self.is_max = is_max
    def push(self, ele):
        heappush(self.hp, -ele if self.is_max else ele)
    def pop(self):
        if self.hp:
            ele = heappop(self.hp)
            ele = - ele if self.is_max else ele
            return ele
    def peek(self):
        if self.is_max:
            return - self.hp[0] if self.hp else None
        else:
            return self.hp[0] if self.hp else None

class MedianFinder(object):
    def __init__(self):
        self._min = Heap()
        self._max = Heap(False)
        self.mid = None
    def update(self):
        self.balance()
        if len(self._min.hp) + 1 == len(self._max.hp):
            self.mid = self._max.peek()
        if len(self._min.hp) == len(self._max.hp) + 1:
            self.mid = self._min.peek()
        if len(self._min.hp) == len(self._max.hp):
            if self._min.hp:
                self.mid = (self._min.peek() + self._max.peek()) / 2.0
            else:
                self.mid = None
    def balance(self):
        if len(self._min.hp) + 1 < len(self._max.hp):
            self._min.push(self._max.pop())
        elif len(self._min.hp) > len(self._max.hp) + 1:
            self._max.push(self._min.pop())
    def addNum(self, num):
        self.update()
        if self.mid == None:
            self.mid = num
        if num <= self.mid:
            self._min.push(num)
        else:
            self._max.push(num)
    def findMedian(self):
        self.update()
        return self.mid
```