Assignment #5: 链表、栈、队列和归并排序

Updated 1348 GMT+8 Mar 17, 2025

2025 spring, Complied by 颜鼎堃 工学院

> **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/

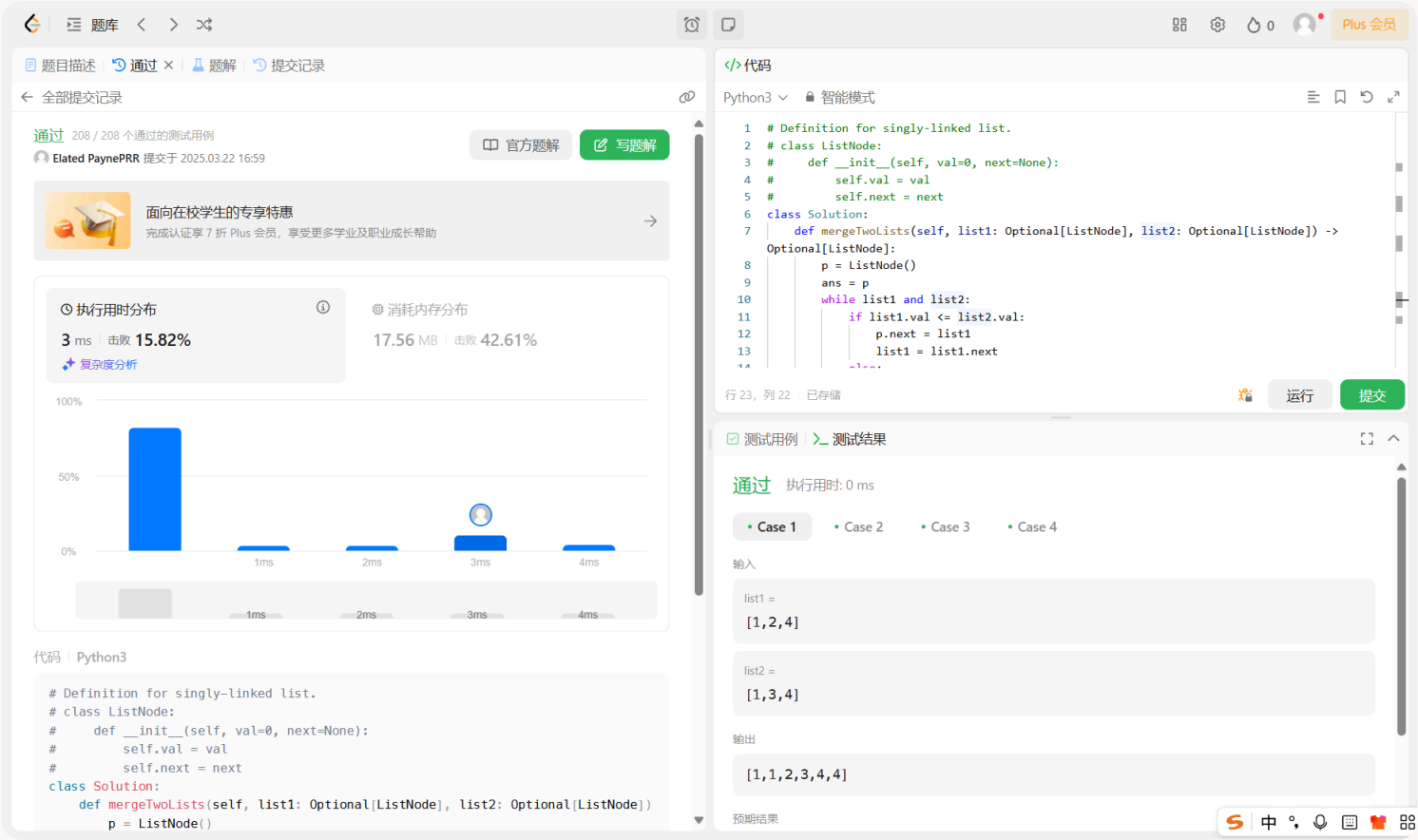
思路：

- 逐点合并，和归并排序有相似之处

代码：

```python
# Definition for singly-linked list.
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:
        p = ListNode()
        ans = p
        while list1 and list2:
            if list1.val <= list2.val:
                p.next = list1
                list1 = list1.next
            else:
                p.next = list2
                list2 = list2.next
            p = p.next
        while list1:
            p.next = list1
            p = p.next
            list1 = list1.next
        while list2:
            p.next = list2
            p = p.next
            list2 = list2.next
        return ans.next
```

代码运行截图（至少包含有"Accepted"）



LC234.回文链表

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

请用快慢指针实现。

- 寒假写过一遍，当时选择把整个链表反转再分别比较，要用`deepcopy()`，很慢
- 现在用快慢指针，只用反转后半段，而且不用`deepcopy()`

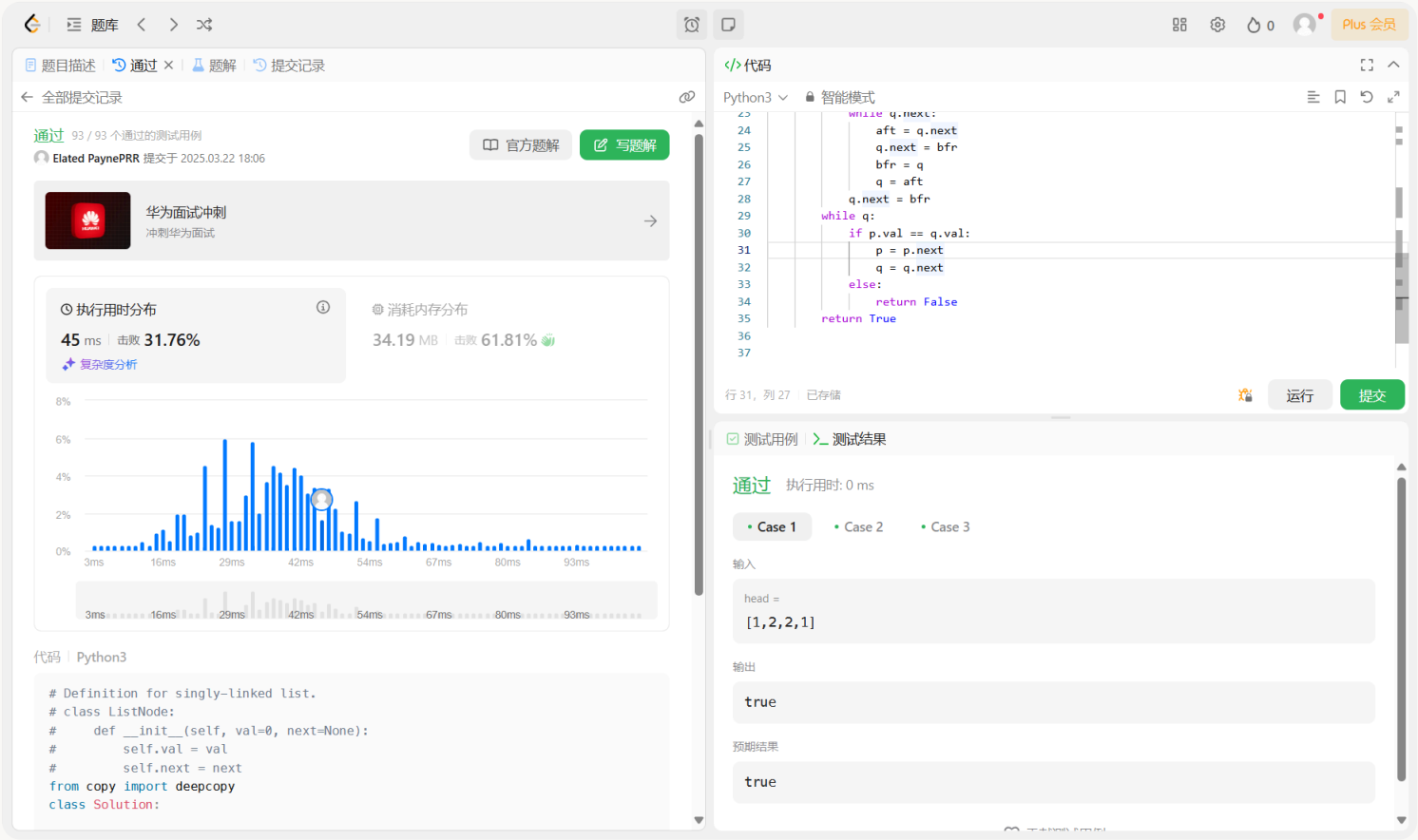
代码：

Python

```python
# Definition for singly-linked list.
from typing import *
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        p, q = head, head
        if not p:
            return head
        while q.next and q.next.next:
            p = p.next
            q = q.next.next
        q = p.next
        p.next = None
        p = head
        bfr = q
        if q and q.next:
            q = q.next
            bfr.next = None
            while q.next:
                aft = q.next
                q.next = bfr
                bfr = q
```

```
                q = aft
            q.next = bfr
        while q:
            if p.val == q.val:
                p = p.next
                q = q.next
            else:
                return False
        return True

```

代码运行截图（至少包含有"Accepted"）



LC1472.设计浏览器历史记录

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

- 用栈写的

请用双链表实现。

- 刚发现要用双链表

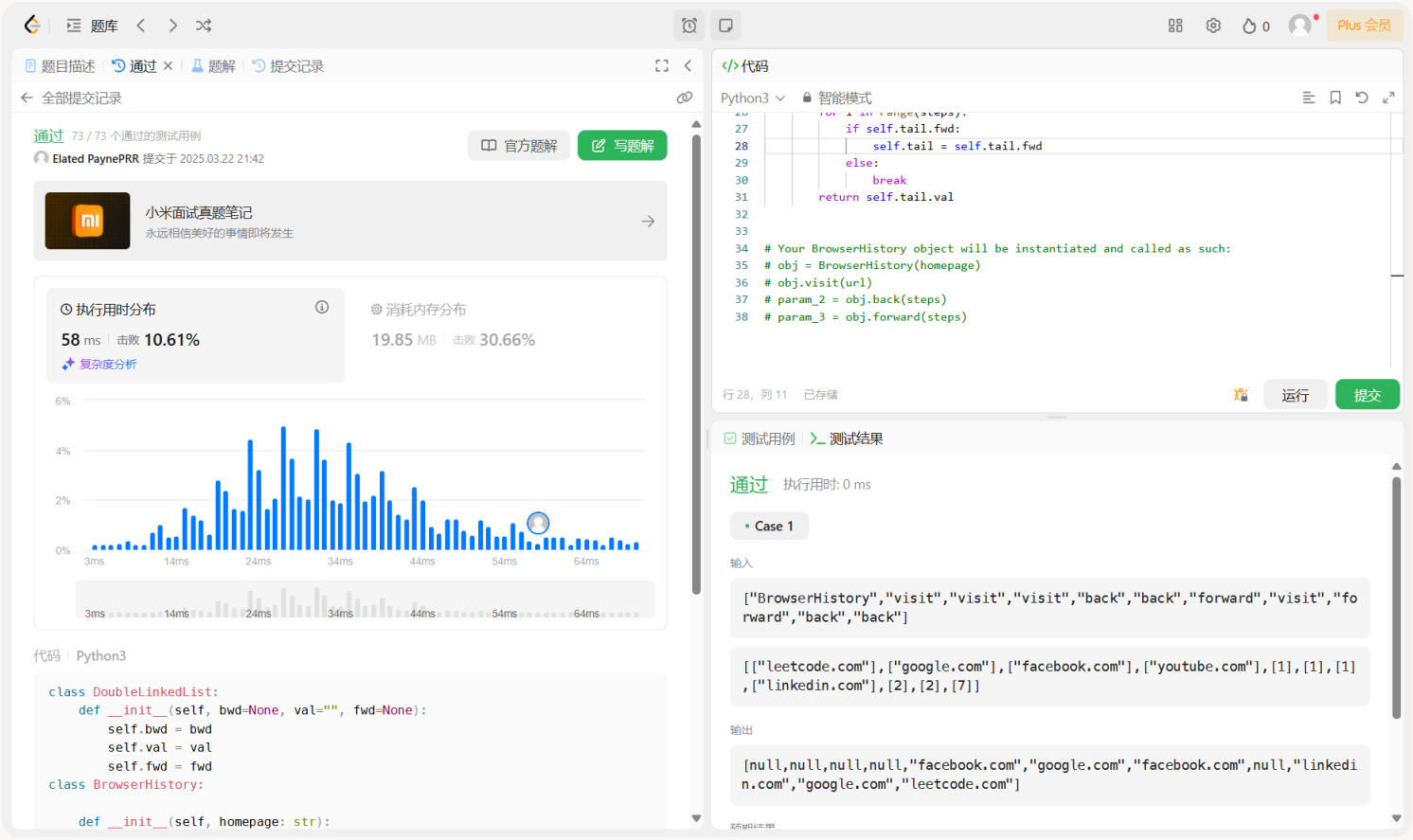
代码：

Python

```
# class BrowserHistory:

#     def __init__(self, homepage: str):
#         self.bwd = [homepage]
#         self.fwd = []

#     def visit(self, url: str) -> None:
#         self.bwd.append(url)
#         if self.fwd:
#             self.fwd = []

#     def back(self, steps: int) -> str:
#         steps = min(steps, len(self.bwd) - 1)
#         for i in range(steps):
#             self.fwd.append(self.bwd.pop())
```

```python
#         return self.bwd[-1]

#     def forward(self, steps: int) -> str:
#         steps = min(steps, len(self.fwd))
#         for i in range(steps):
#             self.bwd.append(self.fwd.pop())
#         return self.bwd[-1]


# # Your BrowserHistory object will be instantiated and called as such:
# # obj = BrowserHistory(homepage)
# # obj.visit(url)
# # param_2 = obj.back(steps)
# # param_3 = obj.forward(steps)
class DoubleLinkedList:
    def __init__(self, bwd=None, val="", fwd=None):
        self.bwd = bwd
        self.val = val
        self.fwd = fwd
class BrowserHistory:

    def __init__(self, homepage: str):
        self.head = DoubleLinkedList(val=homepage)
        self.tail = self.head

    def visit(self, url: str) -> None:
        self.tail.fwd = DoubleLinkedList(self.tail, url)
        self.tail = self.tail.fwd

    def back(self, steps: int) -> str:
        for i in range(steps):
            if self.tail != self.head:
                self.tail = self.tail.bwd
            else:
                break
        return self.tail.val


    def forward(self, steps: int) -> str:
        for i in range(steps):
            if self.tail.fwd:
                self.tail = self.tail.fwd
            else:
                break
        return self.tail.val


# Your BrowserHistory object will be instantiated and called as such:
# obj = BrowserHistory(homepage)
# obj.visit(url)
# param_2 = obj.back(steps)
# param_3 = obj.forward(steps)
```

代码运行截图 （至少包含有"Accepted"）



24591: 中序表达式转后序表达式

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

思路：

- 寒假写的
- 寒假的时候我好厉害，反正现在看不懂了，再看看吧

代码：

```python
prec = {"(": 1, "+": 2, "-": 2, "*": 3, "/": 3}
def infixToPostfix(infix):
    global prec
    op_stack = []
    ans = []
    for token in infix:
        try:
            float(token)
            ans.append(token)
        except ValueError:
            if token == "(":
                op_stack.append(token)
            elif token == ")":
                while (op := op_stack.pop()) != "(":
                    ans.append(op)
            else:
                while op_stack and prec[op_stack[-1]] >= prec[token]:
                    ans.append(op_stack.pop())
                op_stack.append(token)
    while op_stack:
        ans.append(op_stack.pop())
    return " ".join(map(str, ans))

def expToList(exp):
    global prec
```

```
    infix = []
    last = 0
    for i in range(len(exp)):
        if exp[i] in prec or exp[i] == ')':
            if exp[last:i]:
                infix.append(exp[last:i])
            infix.append(exp[i])
            last = i + 1
    if exp[last:]:
        infix.append(exp[last:])
    return infix


for i in range(int(input())):
    print(infixToPostfix(expToList(input().strip())))

```

代码运行截图 （至少包含有"Accepted"）

03253: 约瑟夫问题No.2

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

- 用环形链表做的

请用队列实现。

- 不是哥们，不能老这样吧

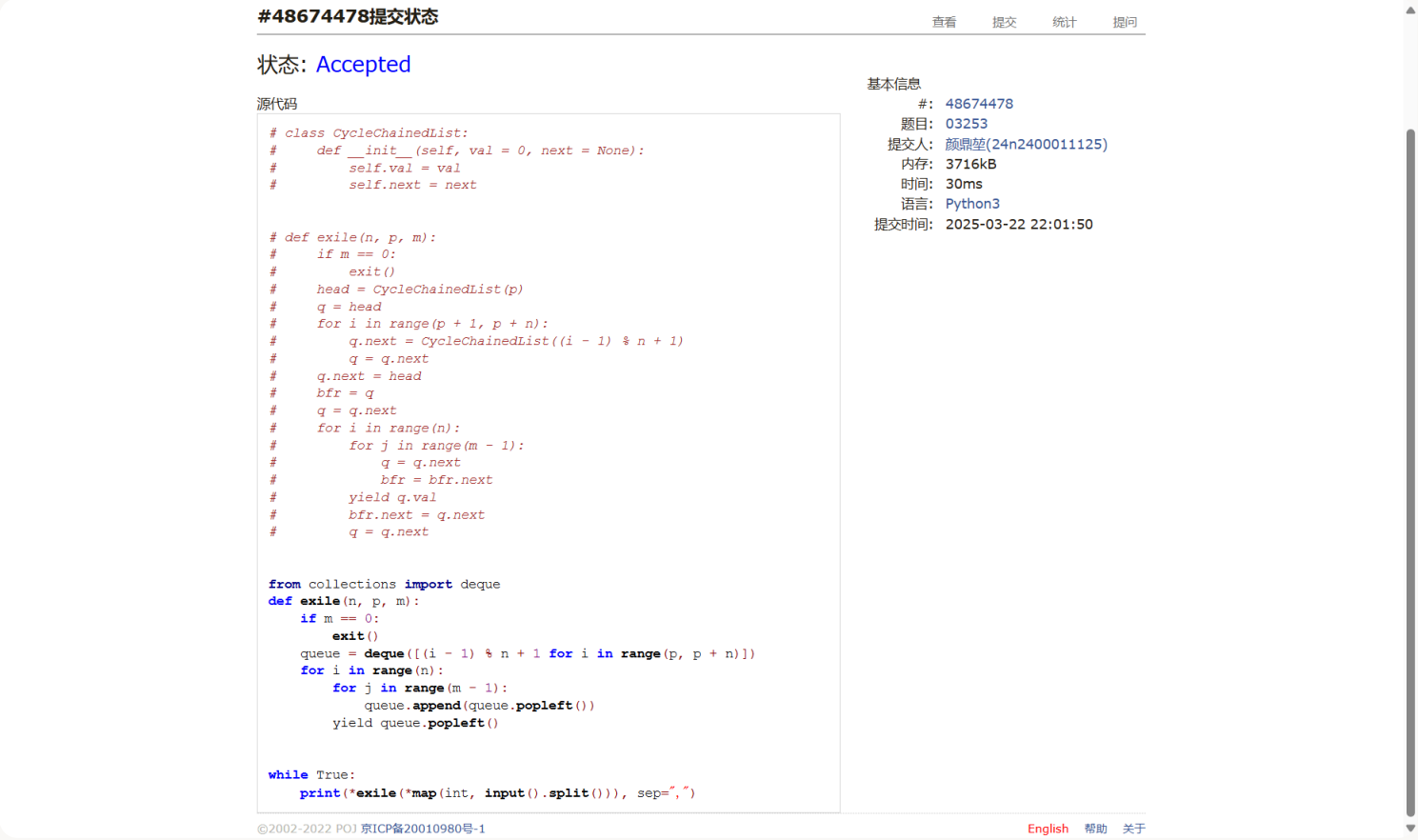
代码：

Python

```
# class CycleChainedList:
#     def __init__(self, val = 0, next = None):
#         self.val = val
#         self.next = next


# def exile(n, p, m):
#     if m == 0:
```

```python
#         exit()
#     head = CycleChainedList(p)
#     q = head
#     for i in range(p + 1, p + n):
#         q.next = CycleChainedList((i - 1) % n + 1)
#         q = q.next
#     q.next = head
#     bfr = q
#     q = q.next
#     for i in range(n):
#         for j in range(m - 1):
#             q = q.next
#             bfr = bfr.next
#         yield q.val
#         bfr.next = q.next
#         q = q.next


from collections import deque
def exile(n, p, m):
    if m == 0:
        exit()
    queue = deque([(i - 1) % n + 1 for i in range(p, p + n)])
    for i in range(n):
        for j in range(m - 1):
            queue.append(queue.popleft())
        yield queue.popleft()


while True:
    print(*exile(*map(int, input().split())), sep=",")

```

代码运行截图 （至少包含有"Accepted"）



#48674478提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
# class CycleChainedList:
#     def __init__(self, val = 0, next = None):
#         self.val = val
#         self.next = next


# def exile(n, p, m):
#     if m == 0:
#         exit()
#     head = CycleChainedList(p)
#     q = head
#     for i in range(p + 1, p + n):
#         q.next = CycleChainedList((i - 1) % n + 1)
#         q = q.next
#     q.next = head
#     bfr = q
#     q = q.next
#     for i in range(n):
#         for j in range(m - 1):
#             q = q.next
#             bfr = bfr.next
#         yield q.val
#         bfr.next = q.next
#         q = q.next


from collections import deque
def exile(n, p, m):
    if m == 0:
        exit()
    queue = deque([(i - 1) % n + 1 for i in range(p, p + n)])
    for i in range(n):
        for j in range(m - 1):
            queue.append(queue.popleft())
        yield queue.popleft()


while True:
    print(*exile(*map(int, input().split())), sep=",")
```

基本信息

#: 48674478
题目: 03253
提交人: 颜鼎堃(24n2400011125)
内存: 3716kB
时间: 30ms
语言: Python3
提交时间: 2025-03-22 22:01:50

English 帮助 关于

20018: 蚂蚁王国的越野跑

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

思路：

- 最开始想的是类似于导弹拦截的动态规划
- 后来发现过不了，然后发现就是求排列的逆序数
- 为了探究为什么过不了甚至写了一个随机生成的程序来比较

代码：

```python
from bisect import bisect_right
from random import randint
inv = 0
def MergeSort(arr):
    if len(arr) ⩽ 1:
        return arr
    mid = len(arr) // 2
    left = MergeSort(arr[:mid])
    right = MergeSort(arr[mid:])
    return merge(left, right)

def merge(left, right):
    global inv
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] ⩾ right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
            inv += len(left) - i
    result.extend(left[i:])
    result.extend(right[j:])
    return result

# ans1, ans2 = 0, 0
# while ans1 == ans2:
#     ans1, ans2 = 0, 0
#     N = 5
#     v = [randint(1, N) for i in range(N)]
#     print(*v, sep=" ")
#     surpass = [9 for i in range(N + 1)]
#     for ant in v:
#         surpass[bisect_right(surpass, ant)] = ant
#         ans1 += bisect_right(surpass, ant - 1)

#     MergeSort(v)
#     ans2 = inv
#     print(surpass, ans1, ans2)


N = int(input())
v = [int(input()) for i in range(N)]
MergeSort(v)
print(inv)
```

代码运行截图 （至少包含有"Accepted"）

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

这次作业交早一点，抽点时间再把每日选做写一写，终于没那么摆烂了

最后一题让我想起来一道物理题：平面上有四只蚂蚁从远处爬来，每一只都做匀速直线运动，显然它们最多两两相遇6次，现在已知发生了5次相遇，证明第6次相遇一定发生