1. 题目

LC21.合并两个有序链表

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

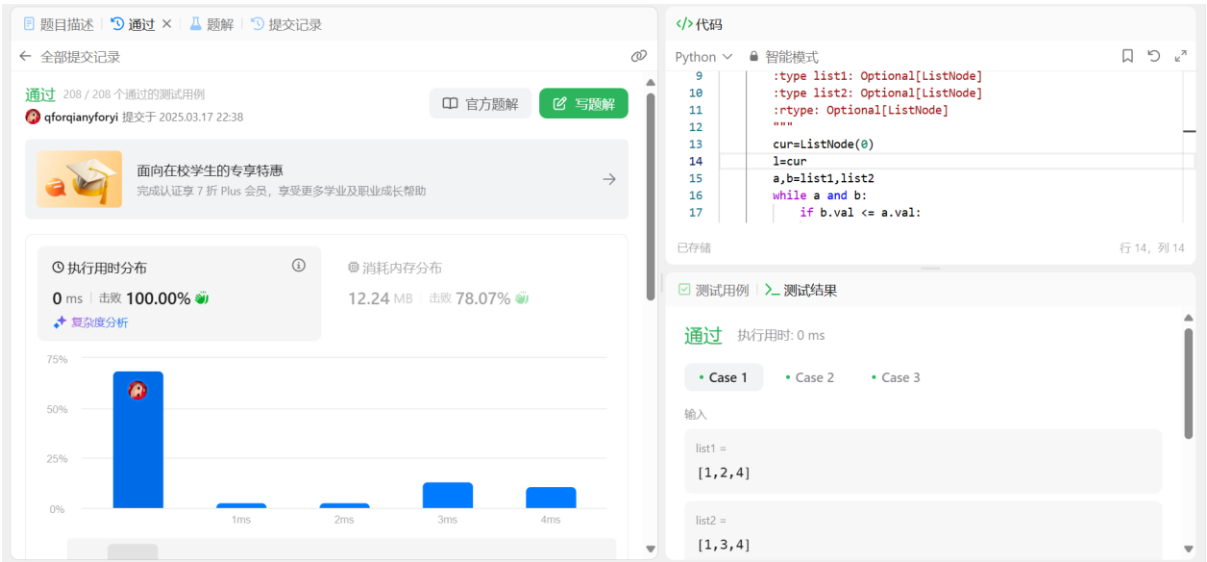
思路：初始化 cur 和 l 为 ListNode(0)，cur 作为指针，l 作为之后不断更新的链表的头节点；之后通过不断对比两个给定列表的值，并将有着较大值的链表更新为 cur.next，并将链表节点和指针移到下一个，最后当遍历完其中一个列表，另一列表中还有值的话就添加到链表中，最后输出以头节点储存的 l，但由于 l 是从 0 初始化开始，所以用.next 跳过。

代码：

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution(object):
    def mergeTwoLists(self, list1, list2):
        """
        :type list1: Optional[ListNode]
        :type list2: Optional[ListNode]
        :rtype: Optional[ListNode]
        """
        cur=ListNode(0)
        l=cur
        a,b=list1,list2
        while a and b:
```

```
        if b.val <= a.val:
            cur.next=b
            b=b.next
            cur=cur.next
        else:
            cur.next=a
            a=a.next
            cur=cur.next
    if a:
        cur.next=a
    if b:
        cur.next=b
    return l.next
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



大约用时：20 分钟

LC234.回文链表

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

<mark>请用快慢指针实现。</mark>

思路：在反转链表的基础上加上快慢指针（slow 每次移动一个节点，fast 每次移动两个节点；当链表长度为偶数时，slow 指针指向链表后半部分的第一个节点，当链表长度为奇数时，slow 指针指向链表的中间节点），找到中间并反转后面，可以避免原链表被完全更改，最后通过依次比较链表两端的元素判断链表是否为回文。

代码：

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution(object):
    def isPalindrome(self, head):
        """
        :type head: Optional[ListNode]
        :rtype: bool
        """
        slow=fast =head
        while fast and fast.next:
            slow=slow.next
            fast=fast.next.next

        tail=None
        cur=slow
```

```
    while cur:
        nexts=cur.next
        cur.next=tail
        tail=cur
        cur=nexts

    ori=head
    while ori and tail:
        if ori.val != tail.val:
            return False
        ori=ori.next
        tail=tail.next
    return True
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

大约用时：45 分钟

LC1472.设计浏览器历史记录

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

<mark>请用双链表实现。</mark>

思路：定义两个指针 prev 和 next 从两端遍历链表，每访问一个新页面，就把它加到链表里，前后页面通过指针连接，后退时通过 next 指针来跳到下一个页面，前进时通过 prev 指针将当前页面跳转到前一个页面。

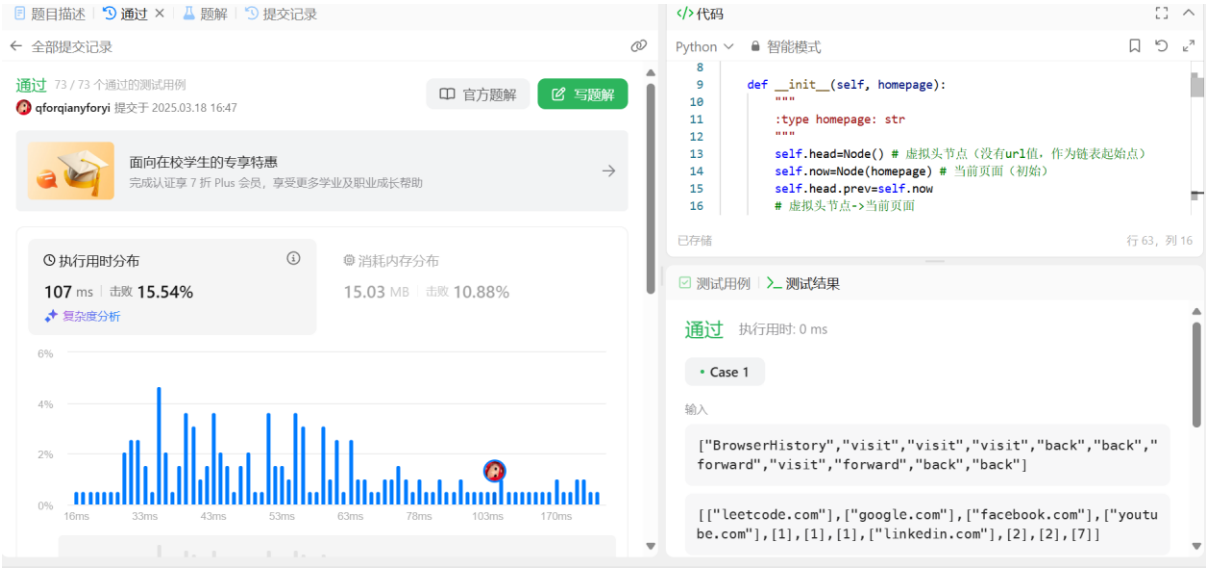
代码：

```
class Node:
    def __init__(self,val=None):
        self.val=val # 历史记录节点的值（url）
        self.prev=None  # 前一个页面的指针
        self.next=None  # 下一个页面的指针

class BrowserHistory(object):

    def __init__(self, homepage):
        """
        :type homepage: str
        """
        self.head=Node() # 虚拟头节点（没有 url 值，作为链表起始点）
        self.now=Node(homepage) # 当前页面（初始）
```

```
    self.head.prev=self.now
    # 虚拟头节点->当前页面
    self.head.next=self.now
     # 当前页面->虚拟头节点
    self.now.prev=self.head
    # 当前页面的前一个页面->头节点
    self.now.next=self.head
    # 当前页面的下一个页面->头节点
    # 循环双向链表

def visit(self, url):
    """
    :type url: str
    :rtype: None
    """
    node=Node(url) # 创建新访问页面的节点
    node.prev=self.head # 新节点 prev=当前页面
    node.next=self.now # 新节点 next=虚拟头节点
    node.prev.next=node # 当前页面 next=新节点
    node.next.prev=node # 虚拟头节点 prev=新节点
    self.now = node # 更新当前页面为新页面

def back(self, steps):
    """
    :type steps: int
    :rtype: str
    """
```

```
        while self.now.next.val: # 向前遍历历史记录
            self.now=self.now.next
            steps -= 1 # 根据指定步数后退到之前的网页
            if steps == 0:
                break
        return self.now.val

    def forward(self, steps):
        """
        :type steps: int
        :rtype: str
        """
        while self.now.prev.val: # 向后遍历历史记录
            self.now=self.now.prev
            steps -= 1 # 根据指定步数向前移动
            if steps == 0:
                break
        return self.now.val
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



大约用时：1 小时 30 分钟

24591: 中序表达式转后序表达式

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

思路：定义一个判断+-*/优先级的 rule 函数，之后遍历整个字符串，判断是否为整数或浮点数，将其加入到 ans 列表中，之后判断括号和+-*/，遇到（压入栈，遇到+-*/根据优先级压入栈中，最后遇到）时弹出栈中的+-*/，直到遇到对应的（。

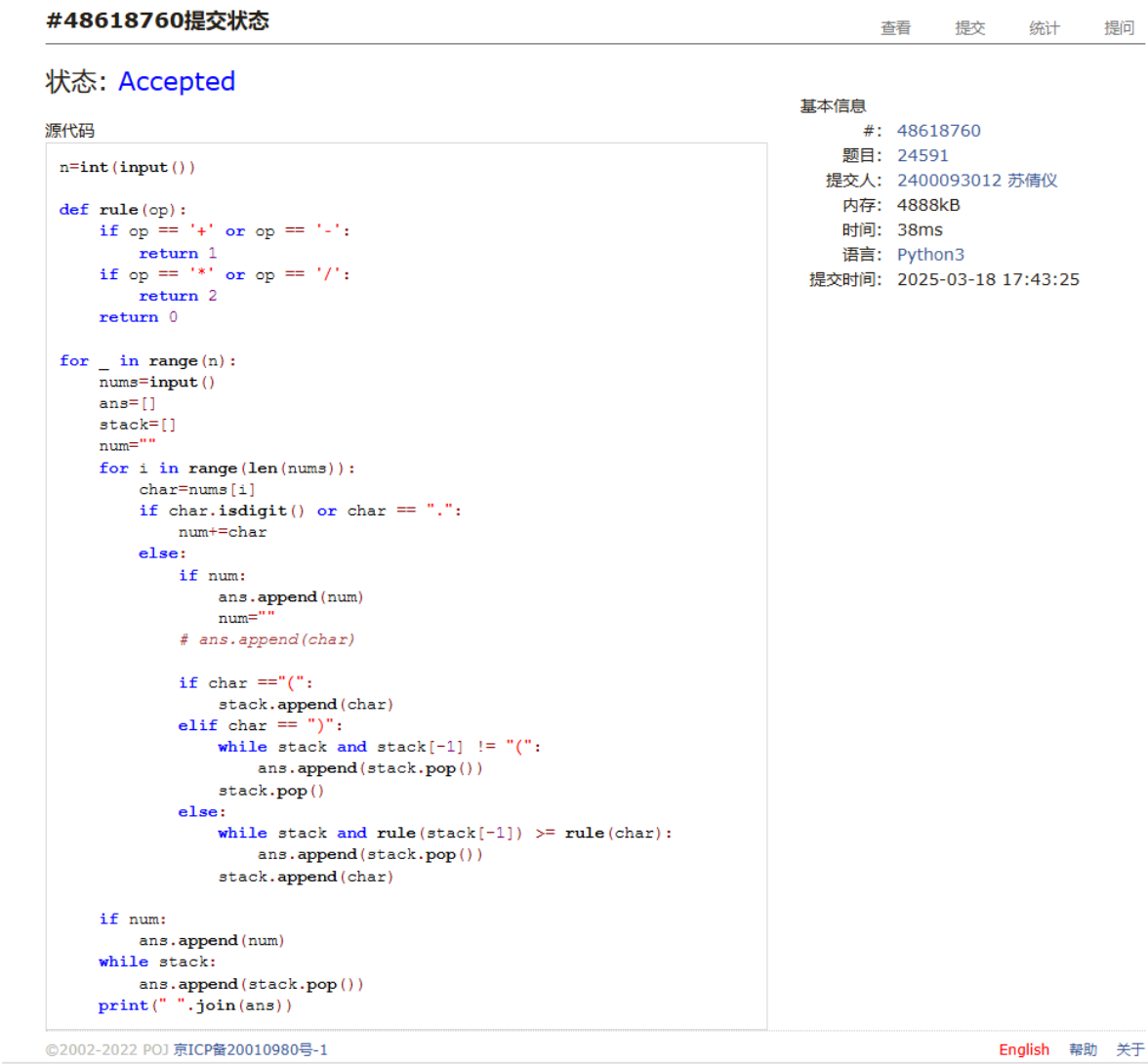
代码：

```
n=int(input())


def rule(op):
    if op == '+' or op == '-':
        return 1
    if op == '*' or op == '/':
```

```
        return 2
    return 0


for _ in range(n):
    nums=input()
    ans=[]
    stack=[]
    num=""
    for i in range(len(nums)):
        char=nums[i]
        if char.isdigit() or char == ".":
            num+=char
        else:
            if num:
                ans.append(num)
                num=""
            # ans.append(char)


            if char =="(":
                stack.append(char)
            elif char == ")":
                while stack and stack[-1] != "(":
                    ans.append(stack.pop())
                stack.pop()
            else:
                while stack and rule(stack[-1]) >= rule(char):
                    ans.append(stack.pop())
                stack.append(char)
```

```
if num:
    ans.append(num)
while stack:
    ans.append(stack.pop())
print(" ".join(ans))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



#48618760提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n=int(input())

def rule(op):
    if op == '+' or op == '-':
        return 1
    if op == '*' or op == '/':
        return 2
    return 0

for _ in range(n):
    nums=input()
    ans=[]
    stack=[]
    num=""
    for i in range(len(nums)):
        char=nums[i]
        if char.isdigit() or char == ".":
            num+=char
        else:
            if num:
                ans.append(num)
                num=""
            # ans.append(char)

            if char =="(":
                stack.append(char)
            elif char == ")":
                while stack and stack[-1] != "(":
                    ans.append(stack.pop())
                stack.pop()
            else:
                while stack and rule(stack[-1]) >= rule(char):
                    ans.append(stack.pop())
                stack.append(char)

    if num:
        ans.append(num)
    while stack:
        ans.append(stack.pop())
    print(" ".join(ans))
```

基本信息

#: 48618760
题目: 24591
提交人: 2400093012 苏倩仪
内存: 4888kB
时间: 38ms
语言: Python3
提交时间: 2025-03-18 17:43:25

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

大约用时：45 分钟

03253: 约瑟夫问题 No.2

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

<mark>请用队列实现。</mark>

思路：生成一个 1 到 n+1 个人的队列，用 rotate(-p)（向左移动）指定从第 p 个人开始，之后当队列人数>1 时不断向左移动 m-1 个位置，让应该出圈的人排在第一位，用 popleft 将其出局，并依次将出局顺序存入 ans 列表中。

代码：

```
from collections import deque


while True:
    n,p,m=map(int,input().split())
    if (n,p,m) == (0,0,0):
        break


    queue=deque(range(1,n+1))


    ans=[]
    queue.rotate(-(p - 1))
    while len(queue) > 1:
        queue.rotate(-(m-1))
        ans.append(queue.popleft())
    if queue:
        ans.append(queue[0])
```

```
print(",".join(map(str,ans)))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#48619699提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

while True:
    n,p,m=map(int,input().split())
    if (n,p,m) == (0,0,0):
        break

    queue=deque(range(1,n+1))

    ans=[]
    queue.rotate(-(p - 1))
    while len(queue) > 1:
        queue.rotate(-(m-1))
        ans.append(queue.popleft())
    if queue:
        ans.append(queue[0])
    print(",".join(map(str,ans)))
```

基本信息

#: 48619699
题目: 03253
提交人: 2400093012 苏倩仪
内存: 3636kB
时间: 30ms
语言: Python3
提交时间: 2025-03-18 19:17:08

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

大约用时：20 分钟

20018: 蚂蚁王国的越野跑

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

思路：将每只蚂蚁的速度数组递归分割成更小的部分，直到每个部分只有一个元素（merge 函数），并合并这些部分，统计逆序对（merge_sort 函数），每次 merge 函数执行时计算当前部分的逆序对并通过 count 累加。

代码：

```
n = int(input())
a=[]
```

```
for _ in range(n):
    m=int(input())
    a.append(m)
# a=[-x for x in a] # 把每个元素变成负数

count=0 # 计数
temp=[0]*n # 临时存放排序过程中的元素

def merge_sort(left,right,mid):
    global count # 允许函数修改全局变量 count
    index1=left # 左边子数组的起始索引（左半部分的指针）
    index2=mid+1 # 右边子数组的起始索引（右半部分的指针）
    for i in range(left,right+1):
        temp[i]=a[i] # 把当前部分的数组内容复制到临时数组 temp 中

    s=left # a 数组的索引指针
    while index1 <= mid and index2 <= right:
        if temp[index1] >= temp[index2]: # 没有逆序对
            a[s]=temp[index1] # 较大的元素放回原数组
            index1+=1
        else:
            count+=(mid - index1 + 1) # 有逆序对，统计逆序对的数量
            a[s]=temp[index2] # 较小的元素放回原数组
            index2+=1
        s+=1

    # 如果有剩余部分也放回原数组
```

```
    while index1 <= mid:
        a[s]=temp[index1]
        index1+=1
        s+=1

    while index2 <= right:
        a[s]=temp[index2]
        index2+=1
        s+=1

def merge(left,right):
    if left < right: # 直到 left>=right 停止
        mid=(left+right)//2 # 计数之间位置
        merge(left,mid) # 对左半部分递归排序
        merge(mid+1,right) # 对右半部分递归排序
        merge_sort(left,right,mid) # 合并

merge(0, n-1) # 从 0 到 n-1 进行归并排序
print(count)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#48622258提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n = int(input())
a=[]
for _ in range(n):
    m=int(input())
    a.append(m)
# a=[-x for x in a] # 把每个元素变成负数

count=0 # 计数
temp=[0]*n # 临时存放排序过程中的元素

def merge_sort(left,right,mid):
    global count # 允许函数修改全局变量count
    index1=left # 左边子数组的起始索引（左半部分的指针）
    index2=mid+1 # 右边子数组的起始索引（右半部分的指针）
    for i in range(left,right+1):
        temp[i]=a[i] # 把当前部分的数组内容复制到临时数组temp中

    s=left # a数组的索引指针
    while index1 <= mid and index2 <= right:
        if temp[index1] >= temp[index2]: # 没有逆序对
            a[s]=temp[index1] # 较大的元素放回原数组
            index1+=1
        else:
            count+=(mid - index1 + 1) # 有逆序对，统计逆序对的数量
            a[s]=temp[index2] # 较小的元素放回原数组
            index2+=1
        s+=1

    # 如果有剩余部分也放回原数组
    while index1 <= mid:
        a[s]=temp[index1]
        index1+=1
        s+=1

    while index2 <= right:
        a[s]=temp[index2]
        index2+=1
        s+=1

def merge(left,right):
    if left < right: # 直到left>=right停止
        mid=(left+right)//2 # 计数之间位置
        merge(left,mid) # 对左半部分递归排序
        merge(mid+1,right) # 对右半部分递归排序
        merge_sort(left,right,mid) # 合并

merge(0, n-1) # 从0到n-1进行归并排序
print(count)
```

基本信息

#: 48622258
题目: 20018
提交人: 2400093012 苏倩仪
内存: 8312kB
时间: 739ms
语言: Python3
提交时间: 2025-03-18 21:58:21

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

大约用时：2 小时

2. 学习总结和收获

<mark>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。</mark>

前两题的链表相比上一周比较能上手了，第二题在反转后原链表也被更改的问题上卡了很久，之后再看一次提示才知道要用快慢指针，也学到了这个用法。第三题和第五题感觉好难，第三题一开始用模拟方法做出来了才发现要双向链表，琢磨了好久最后还是参考了题解，最后仿照题解做了出来，但是还是觉得有点难理解（但是明白了原来还有双向链表这种思路）；而第五题也写了好久写不出，最后在 csdn 找到了题解，也参照题解修改了，尽量把题解的代码理解透了，但看了题解后我还是认为以我现在的能力应该无法自己一个人写出这些代码。。。第四题的后序表达式复习了去年稍微学过一些的计概知识。第五题学到新东西了！这题感觉让我很直观的感受到了队列的原理，理解和写起来也还不费力。