Assignment #5: 链表、栈、队列和归并排序

Updated 1348 GMT+8 Mar 17, 2025

2025 spring, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院

> **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：先创建一个虚拟头节点，再逐个比较两个链表的当前节点，并将较小的节点连接到新链表，最后直接拼接剩余节点并返回合并后的升序链表即可。

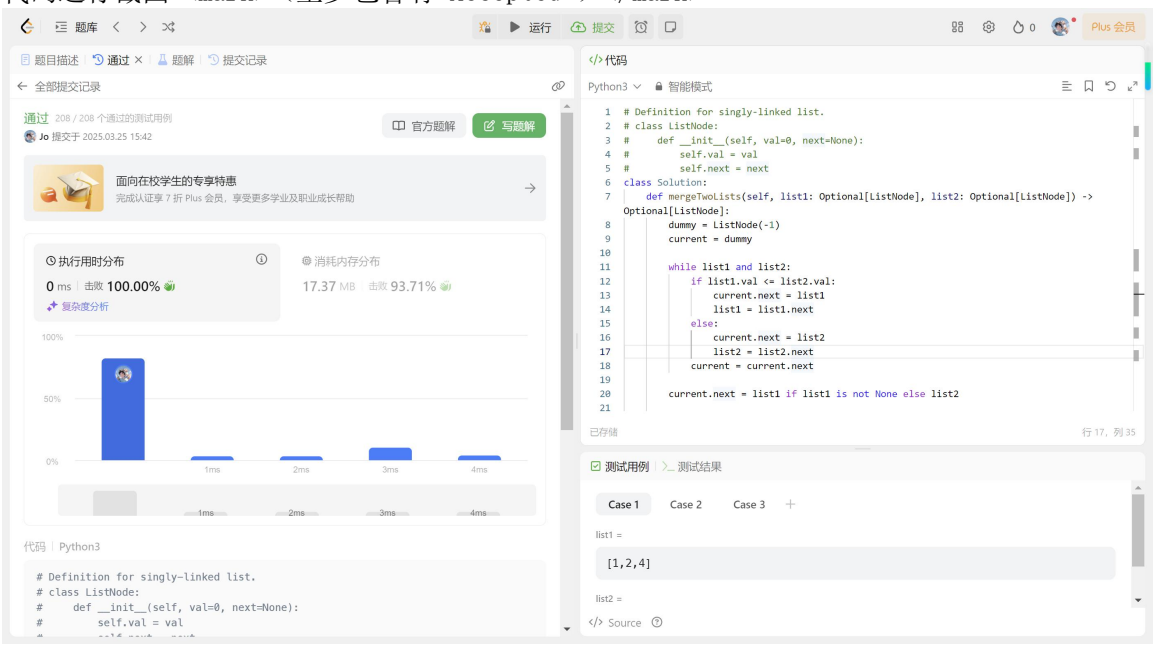
时间：10mins

代码：

```python
```

```
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) ->
Optional[ListNode]:
        dummy = ListNode(-1)
        current = dummy

        while list1 and list2:
            if list1.val <= list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next

        current.next = list1 if list1 is not None else list2

        return dummy.next
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



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：使用快慢指针法，亦即快指针每次走两步，慢指针每次走一步，当快指针到达末尾时，慢指针就在中点；接着，反转后半部分链表，并比较前半部分和反转后的后半部分，看看逐个节点值是否相同即可~

时间：15mins

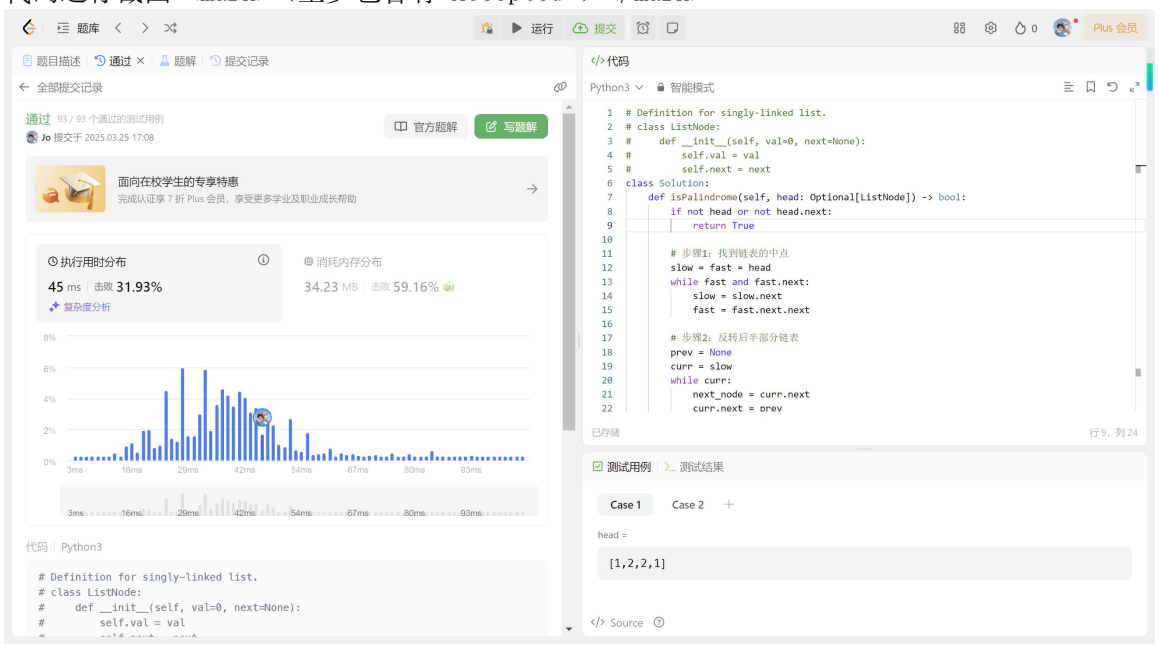
代码：

```python

```

```
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        if not head or not head.next:
            return True

        # 第一步，找到链表的中点
        slow = fast = head
        while fast and fast.next:
            slow = slow.next
            fast = fast.next.next

        # 第二步，反转后半部分链表
        prev = None
        curr = slow
        while curr:
            next_node = curr.next
            curr.next = prev
            prev = curr
            curr = next_node

        # 第三步，比较前半部分和反转后的后半部分
        p1 = head
        p2 = prev
        while p2:  # 只需要比较后半部分的长度即可
            if p1.val != p2.val:
                return False
            p1 = p1.next
            p2 = p2.next
```

```
        return True
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



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：用 visit 创建新节点或添加到数组末尾、清除当前节点之后的所有历史（断开链表或截断数组、更新当前指针到新节点；用 back/forward 移动当前指针（在链表里用 prev/next 即可），与此同时要确保不越界~

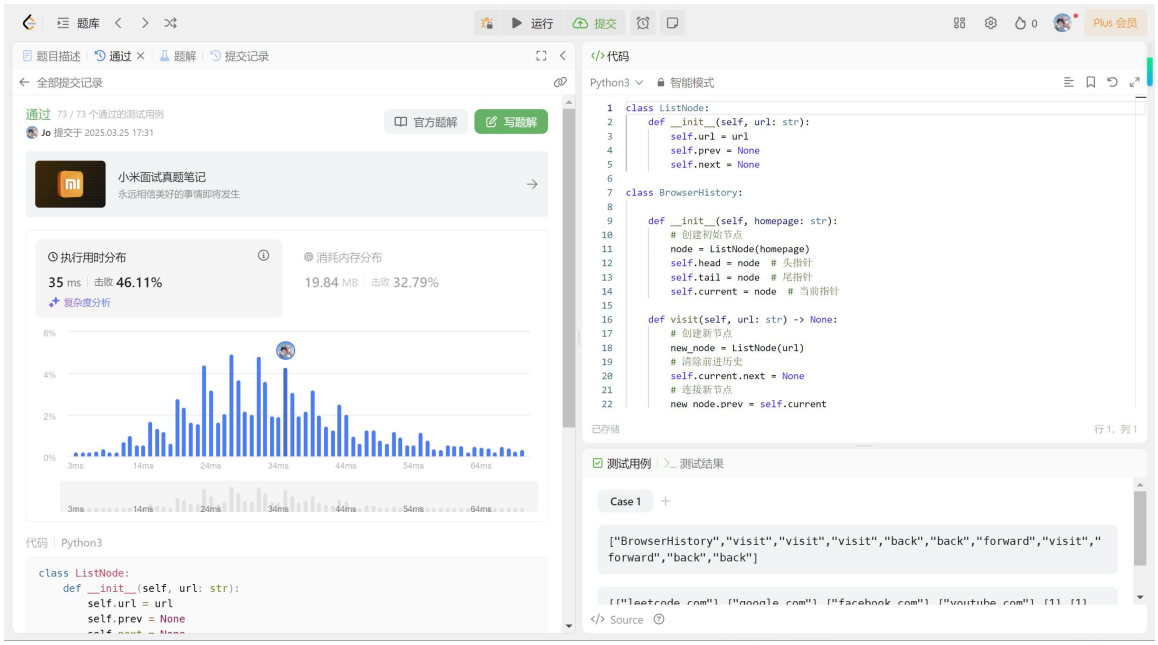
时间：30mins

代码：

```python

```

```
class ListNode:
    def __init__(self, url: str):
        self.url = url
        self.prev = None
        self.next = None

class BrowserHistory:

    def __init__(self, homepage: str):
        # 创建初始节点
        node = ListNode(homepage)
        self.head = node  # 头指针
        self.tail = node  # 尾指针
        self.current = node  # 当前指针

    def visit(self, url: str) -> None:
        # 创建新节点
        new_node = ListNode(url)
        # 清除前进历史
        self.current.next = None
        # 连接新节点
        new_node.prev = self.current
        self.current.next = new_node
        # 更新指针
        self.tail = new_node
        self.current = new_node

    def back(self, steps: int) -> str:
        while steps > 0 and self.current.prev:
            self.current = self.current.prev
            steps -= 1
        return self.current.url

    def forward(self, steps: int) -> str:
        while steps > 0 and self.current.next:
            self.current = self.current.next
            steps -= 1
        return self.current.url
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



24591: 中序表达式转后序表达式

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

思路：先使用 precedence 字典定义运算符的优先级，并创建 output 列表用来存储输出后的后序表达式、operator_stack 用来当成临时存储运算符的栈。
接着，开始处理表达式：对于数字，循环读取数字和小数点，直到遇到非数字或小数点字符，将完整数字加入输出列表；而对于括号，可以将左括号直接入栈，右括号则弹出栈内元素直到左括号，左括号弹出但不输出；对于运算符，先比较当前运算符与栈顶运算符的优先级，弹出栈顶优先级不低于当前运算符的运算符，并将当前运算符入栈即可~

时间：45mins

代码：

```python
```

```
def main(expression):
    precedence = {'+': 1, '-': 1, '*': 2, '/': 2}
```

```
    output = []
    operator_stack = []
    i = 0
    n = len(expression)
    while i < n:
        c = expression[i]
        if c.isdigit() or c == '.':
            # 提取完整的数字
            num = []
            while i < n and (expression[i].isdigit() or expression[i] == '.'):
                num.append(expression[i])
                i += 1
            output.append(''.join(num))
            continue
        elif c == '(':
            operator_stack.append(c)
        elif c == ')':
            while operator_stack and operator_stack[-1] != '(':
                output.append(operator_stack.pop())
            operator_stack.pop()  # 弹出左括号，不输出
        else:  # 运算符
            while (operator_stack and operator_stack[-1] != '(' and
                   precedence.get(operator_stack[-1], 0) >= precedence.get(c, 0)):
                output.append(operator_stack.pop())
            operator_stack.append(c)
        i += 1
    # 处理剩余的运算符
    while operator_stack:
        output.append(operator_stack.pop())
    return ' '.join(output)

n = int(input())for _ in range(n):
    expr = input().strip()
    print(main(expr))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

03253: 约瑟夫问题 No.2

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

<mark>请用队列实现。</mark>

代码：

```python

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

20018: 蚂蚁王国的越野跑

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

思路：

代码：

```python

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

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>

这次作业前两题的难度我个人感觉比较简单，属于比较基础的题目，基本上只要熟练单链表的基本性质，就能够比较快做出来。
第三题对于我来讲则稍微有点难度，因为我双链表使用的并不是很熟练（相关类型的题目还没做几道 qwq，还是练习的太少了），询问了 Deepseek 很多问题之后才把它勉强搞出来。
第四题的话我觉得就是栈的一个比较经典的应用。首先需要先用一个字典对运算符定优先级（加减小于乘除），然后接着再挨个处理括号和所有的运算符在栈内的操作。虽然做起来稍微有点套路化（感觉栈的很多题目思路都有点类似），但里面的细节还是比较多的，我也是 WA 了好几次最后才 AC，花了不少时间。