

Assignment #5: 链表、栈、队列和归并排序

Updated 1348 GMT+8 Mar 17, 2025

2025 spring, Compiled 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:

```
1 # Definition for singly-linked list.
2 class ListNode:
3     def __init__(self, val=0, next=None):
4         self.val = val
5         self.next = next
6 class Solution:
7     def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) →
Optional[ListNode]:
8         p = ListNode()
9         ans = p
10        while list1 and list2:
11            if list1.val ≤ list2.val:
12                p.next = list1
13                list1 = list1.next
14            else:
15                p.next = list2
16                list2 = list2.next
17            p = p.next
18        while list1:
19            p.next = list1
20            p = p.next
21            list1 = list1.next
22        while list2:
23            p.next = list2
24            p = p.next
25            list2 = list2.next
26        return ans.next
```

Python

代码运行截图 (至少包含有"Accepted")

题目描述 通过 × 题解 提交记录

全部提交记录

通过 208 / 208 个通过的测试用例

Elated PaynePRR 提交于 2025.03.22 16:59

官方题解 与题解

面向在校学生的专享特惠

完成认证享 7 折 Plus 会员, 享受更多学业及职业成长帮助

执行用时分布 3 ms | 击败 15.82%

消耗内存分布 17.56 MB | 击败 42.61%

复杂度分析

代码 | Python3

```
# 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()
```

行 23, 列 22 已存储

运行 提交

测试用例 测试结果

通过 执行用时: 0 ms

Case 1 Case 2 Case 3 Case 4

输入

list1 = [1,2,4]

list2 = [1,3,4]

输出

[1,1,2,3,4,4]

预期结果

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:

```
1 # Definition for singly-linked list.
2 from typing import *
3 class ListNode:
4     def __init__(self, val=0, next=None):
5         self.val = val
6         self.next = next
7 class Solution:
8     def isPalindrome(self, head: Optional[ListNode]) -> bool:
9         p, q = head, head
10        if not p:
11            return head
12        while q.next and q.next.next:
13            p = p.next
14            q = q.next.next
15        q = p.next
16        p.next = None
17        p = head
18        bfr = q
19        if q and q.next:
20            q = q.next
21            bfr.next = None
22        while q.next:
23            aft = q.next
24            q.next = bfr
25            bfr = q
```

Python

```

26         q = aft
27         q.next = bfr
28     while q:
29         if p.val == q.val:
30             p = p.next
31             q = q.next
32         else:
33             return False
34     return True
35

```

代码运行截图 (至少包含有"Accepted")

The screenshot displays a LeetCode submission for a problem involving a singly-linked list. The left sidebar shows the problem description, submission status (Accepted), and execution time distribution. The right sidebar shows the code in Python3 and the test results.

Execution Time Distribution:

- 45 ms | 击败 31.76%
- 34.19 MB | 击败 61.81%

Code (Python3):

```

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

```

Test Results:

- 通过 执行用时: 0 ms
- Case 1, Case 2, Case 3

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:

```

1  # class BrowserHistory:
2
3  #     def __init__(self, homepage: str):
4  #         self.bwd = [homepage]
5  #         self.fwd = []
6
7  #     def visit(self, url: str) -> None:
8  #         self.bwd.append(url)
9  #         if self.fwd:
10 #             self.fwd = []
11
12 #     def back(self, steps: int) -> str:
13 #         steps = min(steps, len(self.bwd) - 1)
14 #         for i in range(steps):
15 #             self.fwd.append(self.bwd.pop())

```

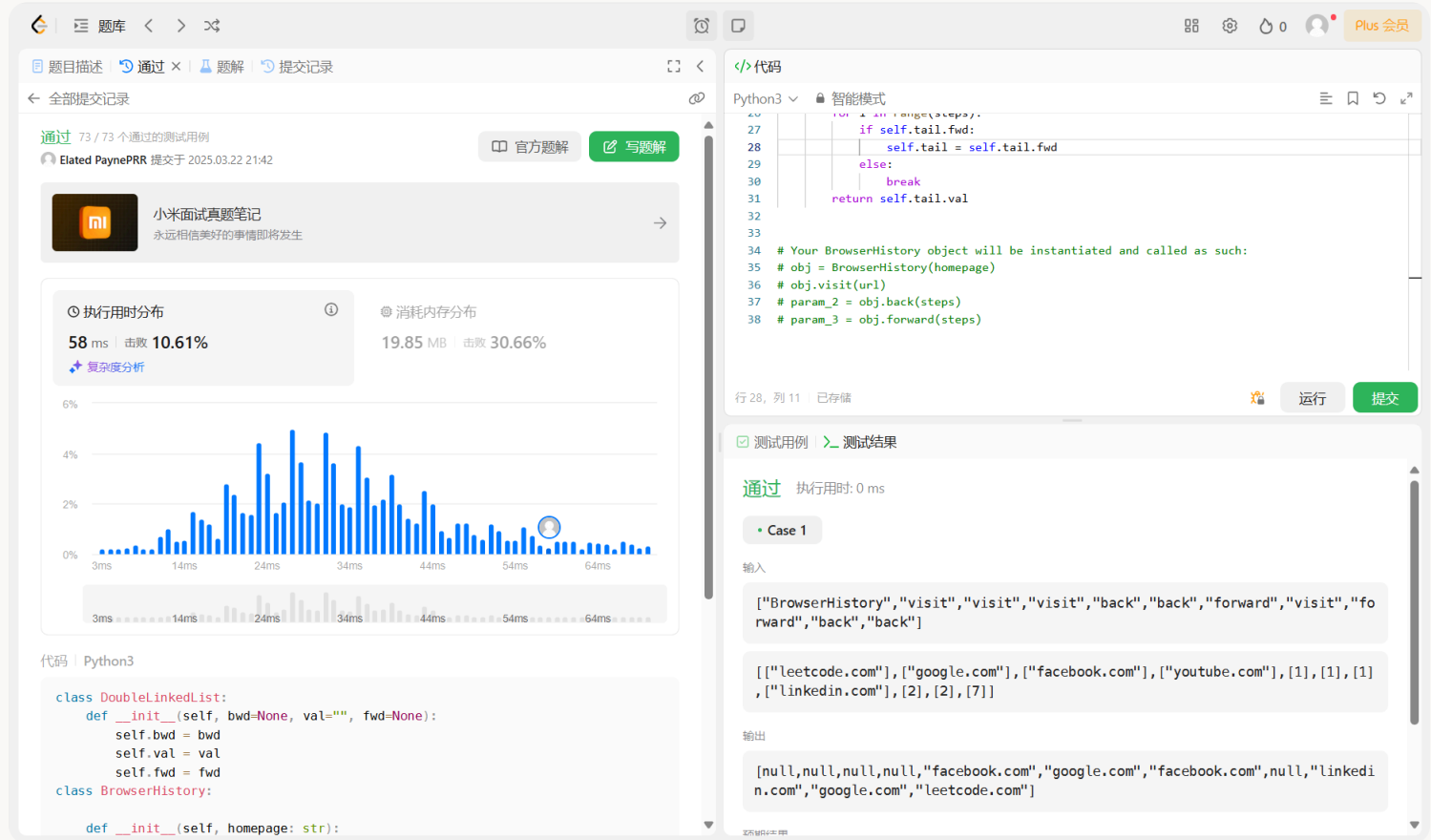
Python

```

16 #         return self.bwd[-1]
17
18 #     def forward(self, steps: int) → str:
19 #         steps = min(steps, len(self.fwd))
20 #         for i in range(steps):
21 #             self.bwd.append(self.fwd.pop())
22 #         return self.bwd[-1]
23
24
25 # # Your BrowserHistory object will be instantiated and called as such:
26 # # obj = BrowserHistory(homepage)
27 # # obj.visit(url)
28 # # param_2 = obj.back(steps)
29 # # param_3 = obj.forward(steps)
30 class DoubleLinkedList:
31     def __init__(self, bwd=None, val="", fwd=None):
32         self.bwd = bwd
33         self.val = val
34         self.fwd = fwd
35 class BrowserHistory:
36
37     def __init__(self, homepage: str):
38         self.head = DoubleLinkedList(val=homepage)
39         self.tail = self.head
40
41     def visit(self, url: str) → None:
42         self.tail.fwd = DoubleLinkedList(self.tail, url)
43         self.tail = self.tail.fwd
44
45     def back(self, steps: int) → str:
46         for i in range(steps):
47             if self.tail != self.head:
48                 self.tail = self.tail.bwd
49             else:
50                 break
51         return self.tail.val
52
53
54     def forward(self, steps: int) → str:
55         for i in range(steps):
56             if self.tail.fwd:
57                 self.tail = self.tail.fwd
58             else:
59                 break
60         return self.tail.val
61
62
63 # Your BrowserHistory object will be instantiated and called as such:
64 # obj = BrowserHistory(homepage)
65 # obj.visit(url)
66 # param_2 = obj.back(steps)
67 # 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:

```
1 prec = {": 1, "+: 2, "-: 2, "*: 3, "/: 3}
2 def infixToPostfix(infix):
3     global prec
4     op_stack = []
5     ans = []
6     for token in infix:
7         try:
8             float(token)
9             ans.append(token)
10        except ValueError:
11            if token == "(":
12                op_stack.append(token)
13            elif token == ")":
14                while (op := op_stack.pop()) != "(":
15                    ans.append(op)
16            else:
17                while op_stack and prec[op_stack[-1]] >= prec[token]:
18                    ans.append(op_stack.pop())
19                op_stack.append(token)
20        while op_stack:
21            ans.append(op_stack.pop())
22        return " ".join(map(str, ans))
23
24 def expToList(exp):
25     global prec
```

Python

代码运行截图 (至少包含有"Accepted")

OpenJudge

题目ID, 标题, 描述

24n2400011125

信箱

账号



CS101 / 题库 (包括计概、数算题目)

题目

排名

状态

提问

#48272446提交状态

查看

提交

统计

提问

状态: Accepted

源代码

```
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
```

基本信息

#: 48272446

题目: 24591

提交人: 颜鼎盛(24n2400011125)

内存: 3716kB

时间: 34ms

语言: Python3

提交时间: 2025-02-11 16:22:16

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：

```
1 # class CycleChainedList:
2 #     def __init__(self, val = 0, next = None):
3 #         self.val = val
4 #         self.next = next
5
6
7 # def exile(n, p, m):
8 #     if m == 0:
```

Python

```

9 #         exit()
10 #     head = CycleChainedList(p)
11 #     q = head
12 #     for i in range(p + 1, p + n):
13 #         q.next = CycleChainedList((i - 1) % n + 1)
14 #         q = q.next
15 #     q.next = head
16 #     bfr = q
17 #     q = q.next
18 #     for i in range(n):
19 #         for j in range(m - 1):
20 #             q = q.next
21 #             bfr = bfr.next
22 #         yield q.val
23 #         bfr.next = q.next
24 #         q = q.next
25
26
27 from collections import deque
28 def exile(n, p, m):
29     if m == 0:
30         exit()
31     queue = deque([(i - 1) % n + 1 for i in range(p, p + n)])
32     for i in range(n):
33         for j in range(m - 1):
34             queue.append(queue.popleft())
35         yield queue.popleft()
36
37
38 while True:
39     print(*exile(*map(int, input().split())), sep=",")
40

```

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

```

# 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

©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:

- 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:

```
1  from bisect import bisect_right
2  from random import randint
3  inv = 0
4  def MergeSort(arr):
5      if len(arr) ≤ 1:
6          return arr
7      mid = len(arr) // 2
8      left = MergeSort(arr[:mid])
9      right = MergeSort(arr[mid:])
10     return merge(left, right)
11
12 def merge(left, right):
13     global inv
14     result = []
15     i = j = 0
16     while i < len(left) and j < len(right):
17         if left[i] ≥ right[j]:
18             result.append(left[i])
19             i += 1
20         else:
21             result.append(right[j])
22             j += 1
23             inv += len(left) - i
24     result.extend(left[i:])
25     result.extend(right[j:])
26     return result
27
28 # ans1, ans2 = 0, 0
29 # while ans1 == ans2:
30 #     ans1, ans2 = 0, 0
31 #     N = 5
32 #     v = [randint(1, N) for i in range(N)]
33 #     print(*v, sep=" ")
34 #     surpass = [9 for i in range(N + 1)]
35 #     for ant in v:
36 #         surpass[bisect_right(surpass, ant)] = ant
37 #         ans1 += bisect_right(surpass, ant - 1)
38
39 #     MergeSort(v)
40 #     ans2 = inv
41 #     print(surpass, ans1, ans2)
42
43
44 N = int(input())
45 v = [int(input()) for i in range(N)]
46 MergeSort(v)
47 print(inv)
```

Python

OpenJudge

题目ID, 标题, 描述

24n2400011125 信箱 账号

CS101 / 题库 (包括计概、数算题目)

题目

排名

状态

提问

#48673968提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
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:
#     N = 5
#     v = [randint(1, N) for i in range(N)]
#     surpass = [1 << 64 for i in range(N + 1)]
#     for i in ...
```

基本信息

#: 48673968

题目: 20018

提交人: 颜鼎堃(24n2400011125)

内存: 11096kB

时间: 714ms

语言: Python3

提交时间: 2025-03-22 21:16:52

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