

Assignment #6: 链表、栈和排序

Updated 2143 GMT+8 Oct 13, 2025

2025 fall, Compiled by 杨浩、化院

1. 1. 题目

1.1 E24588: 后序表达式求值

Stack, <http://cs101.openjudge.cn/practice/24588/>

用时: 5min

思路: 略

代码:

```
1  n=int(input())
2  for i in range(n):
3      stack=[]
4      l=input().split()
5      for j in l:
6          if j[0].isnumeric():
7              stack.append(float(j))
8          else:
9              if j=='+' :
10                 stack[-2]+=stack[-1]
11                 stack.pop()
12             elif j=='-' :
13                 stack[-2]-=stack[-1]
14                 stack.pop()
15             elif j=='*' :
16                 stack[-2]*=stack[-1]
17                 stack.pop()
18             elif j=='/' :
19                 stack[-2]/=stack[-1]
20                 stack.pop()
21     print(f'{stack[0]:.2f}')
```

Fence 1

代码运行截图 (至少包含有 "Accepted")

#50413041提交状态

[查看](#) [提交](#) [统计](#) [提问](#)状态: **Accepted**

源代码

```

##16:29
n=int(input())
for i in range(n):
    stack=[]
    l=input().split()
    for j in l:
        if j[0].isnumeric():
            stack.append(float(j))

```

基本信息

#: 50413041
 题目: 24588
 提交人: 25n2400011769
 内存: 3628kB
 时间: 23ms
 语言: Python3
 提交时间: 2025-10-17 16:33:25

Figure 1

1.2 M234.回文链表

linked list, <https://leetcode.cn/problems/palindrome-linked-list/>请用快慢指针实现 $O(1)$ 空间复杂度。

用时: 24min

思路:

- 快慢指针找到中间点，再把后半段反转

代码:

```

1  class Solution:
2      def isPalindrome(self, head: Optional[ListNode]) -> bool:
3          fast=head
4          slow=head
5          l=0
6          while fast.next:
7              fast=fast.next
8              slow=slow.next
9              l +=1
10             if fast.next:
11                 fast=fast.next
12         mid=slow
13         pr1=slow
14         pr2=slow.next
15         while pr2:
16             slow=pr2
17             pr2=slow.next
18             slow.next=pr1
19             pr1=slow
20         mid.next=None
21         while slow:
22             if head.val!=slow.val:
23                 return False
24             head=head.next
25             slow=slow.next
26         return True

```

Fence 2

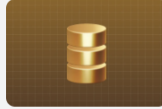
代码运行截图 (至少包含有 "Accepted")

通过 93 / 93 个通过的测试用例 用时: 24 m 1 s

AND-Y 提交于 2025.10.17 16:58

官方题解

写题解



高频 SQL 50 题

面试考点全覆盖应对高阶数据库面试



⌚ 执行用时分布



51 ms | 击败 21.11%

🌟 复杂度分析

📊 消耗内存分布

34.01 MB | 击败 89.54% 🏆

Figure 2

1.3 M27217: 有多少种合法的出栈顺序

<http://cs101.openjudge.cn/practice/27217/>

用时: 20min

思路:

- 动规

代码:

```
1  n=int(input())
2  dp=[0]*(n+1)
3  dp[0]=1
4  dp[1]=1
5  for i in range(1,n+1):
6      s=0
7      for j in range(0,i):
8          s+=dp[j]*dp[i-j-1]
9      dp[i]=s
10 print(dp[-1])
```

Fence 3

代码运行截图 (至少包含有 "Accepted")

#50339938提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
n=int(input())
dp=[0]*(n+1)
dp[0]=1
dp[1]=1
for i in range(1,n+1):
    s=0
    for j in range(0,i):
```

基本信息

#: 50339938
题目: 27217
提交人: 25n2400011769
内存: 3656kB
时间: 261ms
语言: Python3
提交时间: 2025-10-13 15:48:18

Figure 3

1.4 M24591: 中序表达式转后序表达式

<http://cs101.openjudge.cn/practice/24591/>

用时: 15min

思路:

- 栈存在运算符, 遇到 ')' 弹出直到 '('

代码

```
1  n=int(input())
2  for i in range(n):
3      l=input().strip()
4      num_stack=[]
5      num=''
6      dic={'+':1,'-':1,'*':2,'/':2,')':0}
7      for j in l:
8          if j.isnumeric():
9              num +=j
10             elif j=='.':
11                 num +=j
12             else:
13                 if num:
14                     num_stack.append(num)
15                     num=''
16                 num_stack.append(j)
17             if j[0].isnumeric():
18                 num_stack.append(num)
19             stack=[]
20             shuchu=[]
21             for j in num_stack:
22                 if j[0].isnumeric():
23                     shuchu.append(j)
24                 elif j=='(':
25                     stack.append('(')
26             else:
27                 while stack:
28                     if stack[-1]=='(':
29                         break
```

```

30         t=stack.pop()
31         if dic[j]<=dic[t]:
32             shuchu.append(t)
33             continue
34         else:
35             stack.append(t)
36             break
37         if j!=' ':
38             stack.append(j)
39         else:
40             stack.pop()
41     while stack:
42         shuchu.append(stack[-1])
43         stack.pop()
44     res=' '.join(shuchu)
45     print(res)
46

```

Fence 4

(至少包含有 "Accepted")

#50435109提交状态

[查看](#)
[提交](#)
[统计](#)
[提问](#)

状态: Accepted

源代码

```

##16:51
n=int(input())
for i in range(n):
    l=input().strip()
    num_stack=[]
    num=''
    for j in range(len(l)):
        if l[j]!=' ':
            num+=l[j]
        else:
            if num:
                num_stack.append(num)
                num=''
            else:
                continue
    if num:
        num_stack.append(num)
    res=' '.join(num_stack)
    print(res)

```

基本信息

#: 50435109
 题目: 24591
 提交人: 25n2400011769
 内存: 3716kB
 时间: 29ms
 语言: Python3
 提交时间: 2025-10-18 18:18:45

Figure 4

1.5 M02299: Ultra-QuickSort

merge sort, <http://cs101.openjudge.cn/practice/02299/>

用时:25min

思路:

- $i < j$ 且 $arr[i] > arr[j]$ 需要一次交换, 否则不需要交换

代码

```

1  def merge_count(alist):
2      left=0
3      right=len(alist)-1
4      mid=(left+right)//2
5      if right<=0:
6          return 0,alist
7      elif right==1:

```

```

8         if alist[0]>alist[1]:
9             return 1,[alist[1],alist[0]]
10        else:
11            return 0,alist
12    else:
13        left_count,left_arr=merge_count(alist[:mid])
14        right_count,right_arr=merge_count(alist[mid:])
15        i,j=0,0
16        arr=[]
17        count=0
18        while i<len(left_arr) and j<len(right_arr):
19            if left_arr[i]<=right_arr[j]:
20                arr.append(left_arr[i])
21                i +=1
22            else:
23                arr.append(right_arr[j])
24                j +=1
25                count +=len(left_arr)-i
26        arr.extend(left_arr[i:])
27        arr.extend(right_arr[j:])
28        return left_count+count+right_count,arr
29    while True:
30        t=int(input())
31        if t==0:
32            break
33        alist=[]
34        for i in range(t):
35            alist.append(int(input()))
36        print(merge_count(alist)[0])
37
38

```

Fence 5

(至少包含有 "Accepted")

1.6 M146.LRU 缓存

hash table, doubly-linked list, <https://leetcode.cn/problems/lru-cache/>

思路:

- 双端链表

代码:

```

1    class node:
2        def __init__(self,key):

```

```

3         self.next=None
4         self.last=None
5         self.key=key
6     class LRUCache:
7         def __init__(self, capacity: int):
8             self.cache={}
9             self.capacity=capacity
10            self.head=None
11            self.end=None
12        def get(self, key: int) -> int:
13            if key in self.cache:
14                if len(self.cache)==1:
15                    return self.cache[key][0]
16                pr=self.cache[key][1]
17                if pr.next==None:
18                    return self.cache[key][0]
19                if pr.key!=self.head.key:
20                    pr.last.next=pr.next
21            else:
22                self.head=pr.next
23                if pr.key!=self.end.key:
24                    pr.next.last=pr.last
25                pr.next=None
26                pr.last=self.end
27                self.end.next=pr
28                self.end=pr
29                return self.cache[key][0]
30            else:
31                return -1
32        def put(self, key: int, value: int) -> None:
33            if len(self.cache)==self.capacity and key not in
self.cache:
34                self.cache.pop(self.head.key)
35                self.head=self.head.next
36            if key in self.cache:
37                if len(self.cache)==1:
38                    self.cache[key]=(value,node(key))
39                    return
40                pr=self.cache[key][1]
41                if pr.next==None:
42                    self.cache[key]=(value,pr)
43                    return
44                if pr.key!=self.head.key:
45                    pr.last.next=pr.next
46            else:
47                self.head=pr.next
48                if pr.key!=self.end.key:
49                    pr.next.last=pr.last
50                pr.next=None
51                pr.last=self.end
52                self.end.next=pr

```

```
53         self.end=pr
54         self.cache[key]=(value,pr)
55     else:
56         self.cache[key]=(value,node(key))
57         if self.head==None:
58             self.head=self.cache[key][1]
59             self.end=self.cache[key][1]
60         else:
61             self.end.next=self.cache[key][1]
62             self.cache[key][1].last=self.end
63             self.end=self.end.next
```

Fence 6

代码运行截图 (至少包含有 "Accepted")

通过 24 / 24 个通过的测试用例

AND-Y 提交于 2025.10.09 18:20

官方题解

写题解

🕒 执行用时分布

228 ms | 击败 8.37%

📈 复杂度分析

💾 消耗内存分布

79.09 MB | 击败 5.06%

Figure 5

2. 2. 学习总结和个人收获

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 2025fall 每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

本周在Leetcode上练习了一些链表、栈、二叉树等数据结构的题，很多题目初见时都挺有难度的，掌握方法和技巧（如链表的快慢指针）后就大同小异了。