

Assignment #6: 链表、栈和排序

Updated 2143 GMT+8 Oct 13, 2025

2025 fall, Compiled by 胡孝齐 物理学院

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本人头像, 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

E24588: 后序表达式求值

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

思路: 20min, 使用栈进行处理, 后序表达式计算过程可以归结为每次出现运算符时将栈中最后两个数进行相应的运算, 然后运算结果取代原先的两个数, 只要是合格的后序表达式, 最终栈中都只会剩下一个数, 即结果

代码:


```
n=int(input())
for _ in range(n):
    l=list(input().strip().split())
    stack=[]
    for c in l:
        if c=='/':
            b=stack.pop()
            a=stack.pop()
            num=a/b
            stack.append(num)
        elif c=='+':
            num=stack.pop()+stack.pop()
            stack.append(num)
        elif c=='-':
            num=-stack.pop()+stack.pop()
            stack.append(num)
```

```

elif c=='*':
    num=stack.pop()*stack.pop()
    stack.append(num)
else:
    stack.append(float(c))
print(f'{stack[0]:.2f}')

```

代码运行截图 (至少包含有"Accepted")


CS101 / 题库 (包括计概、数算题目)

[题目](#)
[排名](#)
[状态](#)
[提问](#)

#50417400提交状态

[查看](#)
[提交](#)
[统计](#)
[提问](#)

状态: Accepted

源代码

```

n=int(input())
for _ in range(n):
    l=list(input().strip().split())
    stack=[]
    for c in l:
        if c=='/':
            b=stack.pop()
            a=stack.pop()
            num=a/b
            stack.append(num)
        elif c=='+':
            num=stack.pop()+stack.pop()
            stack.append(num)
        elif c=='-':
            num=-stack.pop()+stack.pop()
            stack.append(num)
        elif c=='*':
            num=stack.pop()*stack.pop()
            stack.append(num)
        else:
            stack.append(float(c))
    print(f'{stack[0]:.2f}')

```

基本信息

#: 50417400
 题目: 24588
 提交人: 25n2400011320
 内存: 3580kB
 时间: 24ms
 语言: Python3
 提交时间: 2025-10-17 18:45:14

©2002-2022 POJ 京ICP备20010980号-1

[English](#)
[帮助](#)
[关于](#)

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。

思路：核心思路是考虑到单向链表没有指向前一个元素的指针，因而需要将后半部分链表反转后才能判断是否是回文链表，反转的方法是（可以用快慢指针，也可以不用）找到中间点，然后反转

代码：

```

class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        A=B=head
        while B and B.next:
            A=A.next
            B=B.next.next
        b=A
        c=None
        while b:
            a=b.next
            b.next=c
            c=b
            b=a
        i=head
        j=c

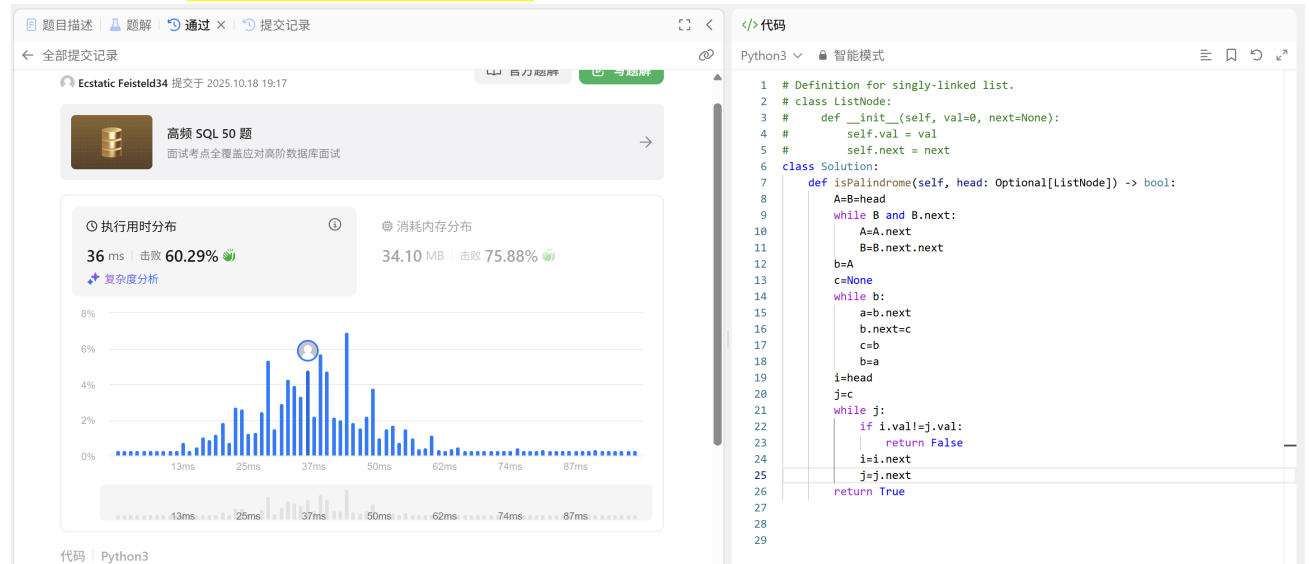
```

```

while j:
    if i.val!=j.val:
        return False
    i=i.next
    j=j.next
return True

```

代码运行截图 (至少包含有"Accepted")



M27217: 有多少种合法的出栈顺序

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

思路：本质上就是操作1和操作2的顺序不同导致结果不同，操作1和操作2的顺序需要满足栈中元素个数始终大于等于0即可。本题由于递归深度超过默认限制（1000），直接递归会runtime error 如果数学比较好可以直接得到卡特兰数的公式

代码：

```

import sys
sys.setrecursionlimit(2001)
n=int(input())
dic={}
def f(nums_in_stack,n1,n2):
    if nums_in_stack<0:
        return 0
    if n1==n2==n:
        return 1
    if n1>n or n2>n:
        return 0
    if (nums_in_stack,n1,n2) not in dic:
        dic[(nums_in_stack,n1,n2)]=f(nums_in_stack-1,n1,n2+1)+f(nums_in_stack+1,n1+1,n2)
    return dic[(nums_in_stack,n1,n2)]
print(f(0,0,0))

```

状态: Accepted

源代码

```
import sys
sys.setrecursionlimit(2001)
n=int(input())
dic={}
def f(nums_in_stack,n1,n2):
    if nums_in_stack<0:
        return 0
    if n1==n2==n:
        return 1
    if n1>n or n2>n:
        return 0
    if (nums_in_stack,n1,n2) not in dic:
        dic[(nums_in_stack,n1,n2)]=f(nums_in_stack-1,n1,n2+1)+f(nums_in_stack,n1,n2)
    return dic[(nums_in_stack,n1,n2)]
print(f(0,0,0))
```

基本信息

#: 50439099
题目: 27217
提交人: 25n2400011320
内存: 129472kB
时间: 646ms
语言: Python3
提交时间: 2025-10-18 21:37:15

M24591:中序表达式转后序表达式

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

思路：使用栈储存运算符和括号，括号内的单独视为一个表达式处理，处理结果则为一个整体。

代码

```
n=int(input().strip())
for _ in range(n):
    s=input().strip()
    operator=[]
    output=[]
    ops={'+' :0, '-' :0, '*' :1, '/' :1}
    i=0
    l=len(s)
    while i<l:
        if '0'<=s[i]<='9':
            num=''
            while i<l and ('0'<=s[i]<='9' or s[i]=='.'):
                num+=s[i]
                i+=1
            i-=1
            output.append(num)
        elif s[i]=='(':
            operator.append('(')
        elif s[i]==')':
            while operator and operator[-1]!='(':
                output.append(operator.pop())
            operator.pop()
        else:
            while operator and operator[-1] in ops and ops[operator[-1]]>=ops[s[i]]:
                output.append(operator.pop())
            operator.append(s[i])
            i+=1
    while operator:
        output.append(operator.pop())
    print(*output)
```

(至少包含有"Accepted")

#50470757提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n=int(input().strip())
for _ in range(n):
    s=input().strip()
    operator=[]
    output=[]
    ops={'+' :0, '-' :0, '*' :1, '/' :1}
    i=0
    l=len(s)
    while i<l:
        if '0'<=s[i]<='9':
            num=''
            while i<l and ('0'<=s[i]<='9' or s[i]=='.' ):
                num+=s[i]
                i+=1
            i-=1
            output.append(num)
        elif s[i]=='(':
            operator.append('(')
        elif s[i]==')':
            while operator and operator[-1]!='(':
                output.append(operator.pop())
            operator.pop()
        else:
            while operator and operator[-1] in ops and ops[operator[-1]]>ops[s[i]]:
                output.append(operator.pop())
            operator.append(s[i])
            i+=1
    while operator:
        output.append(operator.pop())
    print(*output)
```

基本信息

#: 50470757
题目: 24591
提交人: 25n2400011320
内存: 3676kB
时间: 27ms
语言: Python3
提交时间: 2025-10-20 19:31:52

M02299:Ultra-QuickSort

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

思路:

代码

(至少包含有"Accepted")

M146.LRU缓存

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

思路:

代码:

代码运行截图 (至少包含有"Accepted")

2. 学习总结和个人收获

练习了一些栈的应用