Assignment #4: 排序、栈、队列和树

Updated 0005 GMT+8 March 11, 2024

2024 spring, Complied by 夏天，生命科学学院

说明：

1）The complete process to learn DSA from scratch can be broken into 4 parts:

Learn about Time complexities, learn the basics of individual Data Structures, learn the basics of Algorithms, and practice Problems.

2）请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

3）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、"作业评论"区有上传的md或者doc附件。

4）如果不能在截止前提交作业，请写明原因。

编程环境

（请改为同学的操作系统、编程环境等）

操作系统：Windows 10 家庭版

Python编程环境：Spyder python（3.11）

1. 题目

05902: 双端队列

http://cs101.openjudge.cn/practice/05902/

思路：按照题目要求使用deque即可

代码

```
from collections import deque
t=int(input())
for i in range(t):
    n=int(input())
    queue=deque()
    for ii in range(n):
        a,b=map(int,input().split())
        if a==1:
            queue.append(b)
        else:
            if queue:
                if b==0:
                    queue.popleft()
                else:
                    queue.pop()
            else:
                break
    if len(queue)!=0:
        print(' '.join(map(str,list(queue))))
    else:
        print('NULL')
```

代码运行截图（至少包含有"Accepted"）

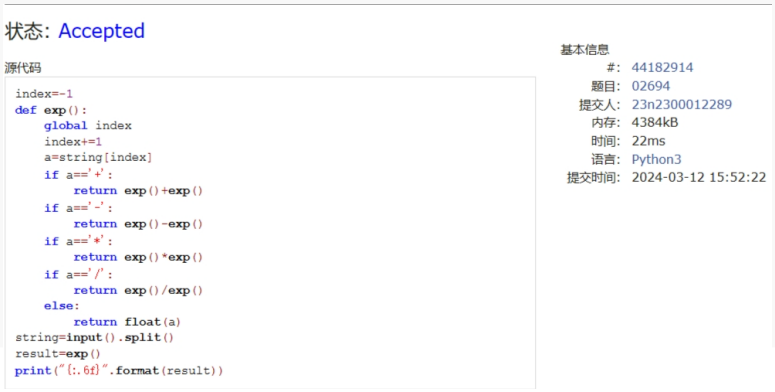
02694: 波兰表达式

http://cs101.openjudge.cn/practice/02694/

思路：递归，没遇到数就需要空着等后面的数填进去

代码

```
index=-1
def exp():
    global index
    index+=1
    a=string[index]
    if a=='+':
        return exp()+exp()
    if a=='-':
        return exp()-exp()
    if a=='*':
        return exp()*exp()
    if a=='/':
        return exp()/exp()
    else:
        return float(a)
string=input().split()
result=exp()
print("{:.6f}".format(result))
```



代码运行截图（至少包含有"Accepted"）

24591: 中序表达式转后序表达式

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

思路：见注释

代码

```
def middle_to_post(expression):
    priority={'+':1,'-':1,'*':2,'/':2} #定义运算优先级
    stack=[] #空栈，储存运算符号
    ans=[] #储存答案
    for i in expression:
        if i.isdigit() or '.' in i: #如果是数字，直接加入ans中
            ans.append(i)
        elif i=='(': #如果是（，直接加入stack中
            stack.append(i)
        elif i==')': #如果是），与最近的（匹配，之间的运算符号加入ans中
            while stack and stack[-1]!='(':
                ans.append(stack.pop())
            stack.pop() #将匹配的（弹出
        else: #如果是运算符号，比较当前符号与栈顶符号的优先级
            while stack and stack[-1]!='(' and priority[i]<=priority.get(stack[-1],0):
                ans.append(stack.pop())
            stack.append(i)
    while stack: #剩余符号全部弹出，加入ans
        ans.append(stack.pop())
    return ' '.join(ans)
n=int(input())
for ii in range(n):
    expression=input().replace('+', ' + ').replace('-', ' - ')\
                      .replace('*', ' * ').replace('/', ' / ')\
                      .replace('(', ' ( ').replace(')', ' ) ').split() #增加空格，便于分离
    print(middle_to_post(expression))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

22068: 合法出栈序列

http://cs101.openjudge.cn/practice/22068/

思路：见注释

代码

```
def is_possible_out_stack(x,test):
    stack=[]
    index=0
    if len(x)!=len(test):
        return 'NO'
    for i in test:
        if i in x:
            while not stack or stack[-1]!=i: #当栈是空的或者当前元素不是栈顶元素时
                if index==len(x):
                    return 'NO'
                stack.append(x[index]) #x中元素按序入栈
                index+=1
            stack.pop() #当前元素为栈顶元素，出栈
        else:
            return 'NO'
    if stack:
        return 'NO'
    else:
        return 'YES'
x=input().strip()
while True:
    try:
        test=input().strip()
        print(is_possible_out_stack(x,test))
    except EOFError:
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

06646: 二叉树的深度

http://cs101.openjudge.cn/practice/06646/

思路：参考了题解

代码

```
class TreeNode:
    def __init__(self,x):
        self.val=x
        self.left=None
        self.right=None
def build_tree(n,nodes):
    dic={i:TreeNode(i)for i in range(1,n+1)}
    for i,(left,right) in enumerate(nodes,start=1):
        if left!=-1:
            dic[i].left=dic[left]
        if right!=-1:
            dic[i].right=dic[right]
    return dic[1]
def max_depth(root):
    if root is None:
        return 0
    else:
        left_depth=max_depth(root.left)
        right_depth=max_depth(root.right)
        return max(left_depth,right_depth)+1
n=int(input())
nodes=[]
for _ in range(n):
    left,right=map(int,input().split())
    nodes.append((left,right))
root=build_tree(n,nodes)
print(max_depth(root))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

02299: Ultra-QuickSort

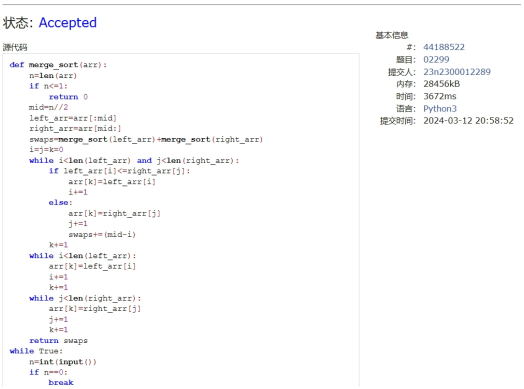
http://cs101.openjudge.cn/practice/02299/

思路：归并排序稍作修改（有点像线代/高代里排列的逆序数那种问题）

代码

```
    def merge_sort(arr):
        n=len(arr)
        if n<=1:
            return 0
        mid=n//2
        left_arr=arr[:mid]
        right_arr=arr[mid:]
        swaps=merge_sort(left_arr)+merge_sort(right_arr)
        i=j=k=0
        while i<len(left_arr) and j<len(right_arr):
            if left_arr[i]<=right_arr[j]:
                arr[k]=left_arr[i]
                i+=1
            else:
                arr[k]=right_arr[j]
                j+=1
                swaps+=(mid-i)
            k+=1
        while i<len(left_arr):
            arr[k]=left_arr[i]
            i+=1
            k+=1
        while j<len(right_arr):
            arr[k]=right_arr[j]
            j+=1
            k+=1
        return swaps
    while True:
        n=int(input())
        if n==0:
            break
        arr=[]
        for _ in range(n):
            arr.append(int(input()))
        print(merge_sort(arr))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。

做作业还是学到了挺多的新东西，比如栈中元素先入后出，借助class写树，归并排序（体现了二分的思想）

这么相比双端队列和波兰表达式算简单的了，接下来打算学一下类（class）的有关代码，然后尝试练习二叉树的代码