Assignment #4: 排序、栈、队列和树

Updated 0005 GMT+8 March 11, 2024

2024 spring, Complied by 武昱达 23工院

编程环境

操作系统：Windows 11

Python编程环境：PyCharm 2023.1.4 (Professional Edition)

1. 题目

05902: 双端队列

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

思路：模拟

代码

```
from collections import deque
for _ in range(t:=int(input())):
    q=deque()
    for _ in range(n:=int(input())):
        type,num=map(int,input().split())
        if type==1:
            q.append(num)
        else:
            if num==0:
                q.popleft()
            else:
                q.pop()
    if q:
        print(*q)
    else:
        print("NULL")
```

代码运行截图 （至少包含有"Accepted"）

#44146914提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque
for _ in range(t:=int(input())):
    q=deque()
    for _ in range(n:=int(input())):
        type,num=map(int,input().split())
        if type==1:
            q.append(num)
        else:
            if num==0:
                q.popleft()
            else:
                q.pop()
    if q:
        print(*q)
    else:
        print("NULL")
```

基本信息

#: 44146914
题目: 05902
提交人: 23n2300011119 (武)
内存: 4052kB
时间: 41ms
语言: Python3
提交时间: 2024-03-10 10:17:45

02694: 波兰表达式

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

思路:

代码

```
def cal(x1,x2,operate):
    m,n=float(x1),float(x2)
    if operate=="+":return m+n
    if operate=="-":return m-n
    if operate=="*":return m*n
    if operate=="/":return m/n
#用一个空栈存放数字，遇到数字则入栈，
#遇到运算符则弹出两个数字执行运算。

raw,stack=list(map(str,input().split())),[]
operators=["+","-","*","/"]
for i in range(len(raw)-1,-1,-1):
    if raw[i] not in operators:stack.append(raw[i])
    else:stack.append(cal(stack.pop(),stack.pop(),raw[i]))
print("{:.6f}".format(stack[-1]))
```

代码运行截图 （至少包含有"Accepted"）

#44059079提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
def cal(x1,x2,operate):
    m,n=float(x1),float(x2)
    if operate=="+":return m+n
    if operate=="-":return m-n
    if operate=="*":return m*n
    if operate=="/":return m/n
#用一个空栈存放数字，遇到数字则入栈，
#遇到运算符则弹出两个数字执行运算。

raw,stack=list(map(str,input().split())),[]
operators=["+","-","*","/"]
for i in range(len(raw)-1,-1,-1):
    if raw[i] not in operators:stack.append(raw[i])
    else:stack.append(cal(stack.pop(),stack.pop(),raw[i]))
print("{:.6f}".format(stack[-1]))
```

基本信息

#: 44059079
题目: 02694
提交人: 23n2300011119 (武)
内存: 3612kB
时间: 21ms
语言: Python3
提交时间: 2024-03-03 19:53:04

English 帮助 关于

24591: 中序表达式转后序表达式

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

思路:

代码

```python
def infix_to_postfix(expression):
    def get_precedence(op):
        precedences = {'+': 1, '-': 1, '*': 2, '/': 2}
        return precedences[op] if op in precedences else 0

    def is_operator(c):
        return c in "+-*/"

    def is_number(c):
        return c.isdigit() or c == '.'

    output = []
    stack = []

    number_buffer = []

    def flush_number_buffer():
        if number_buffer:
            output.append(''.join(number_buffer))
            number_buffer.clear()

    for c in expression:
        if is_number(c):
            number_buffer.append(c)
        elif c == '(':
            flush_number_buffer()
            stack.append(c)
        elif c == ')':
            flush_number_buffer()
            while stack and stack[-1] != '(':
                output.append(stack.pop())
            stack.pop()  # popping '('
```

```
        elif is_operator(c):
            flush_number_buffer()
            while stack and get_precedence(c) <= get_precedence(stack[-1]):
                output.append(stack.pop())
            stack.append(c)

    flush_number_buffer()
    while stack:
        output.append(stack.pop())

    return ' '.join(output)

# Read number of expressions
n = int(input())

# Read each expression and convert it
for _ in range(n):
    infix_expr = input()
    postfix_expr = infix_to_postfix(infix_expr)
    print(postfix_expr)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
def infix_to_postfix(expression):
    def get_precedence(op):
        precedences = {'+': 1, '-': 1, '*': 2, '/': 2}
        return precedences[op] if op in precedences else 0

    def is_operator(c):
        return c in "+-*/"

    def is_number(c):
        return c.isdigit() or c == '.'

    output = []
    stack = []

    number_buffer = []

    def flush_number_buffer():
        if number_buffer:
            output.append(''.join(number_buffer))
            number_buffer.clear()

    for c in expression:
        if is_number(c):
            number_buffer.append(c)
        elif c == '(':
            flush_number_buffer()
```

基本信息

#:	44071067
题目:	24591
提交人:	23n2300011119 (武)
内存:	3700kB
时间:	27ms
语言:	Python3
提交时间:	2024-03-04 22:16:34

22068: 合法出栈序列

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

思路:

代码

```
from collections import deque
import copy
raw=deque(input())
while True:
    try:
```

```
        case=deque(input())
        raw_temp=copy.deepcopy(raw)
        temp=[] # temp是一个空栈
        res=""
        if len(case)!=len(raw):
            print("NO")
            continue
        for i in case:
            if res.find(i)!=-1:
                continue
            if i in raw_temp:
                while raw_temp[0]!=i:
                    a=raw_temp.popleft()
                    temp.append(a)
                a=raw_temp.popleft()
                temp.append(a)
                b=temp.pop()
                res+=b
            if i in temp:
                while True:
                    c=temp.pop()
                    res+=c
                    if c==i:
                        break
        while temp:
            a=temp.pop()
            res+=a
        if res==""".join(case):
            print("YES")
        else:
            print("NO")
    except:
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
from collections import deque
import copy
raw=deque(input())
while True:
    try:
        case=deque(input())
        raw_temp=copy.deepcopy(raw)
        temp=[] # temp是一个空栈
        res=""
        if len(case)!=len(raw):
            print("NO")
            continue
        for i in case:
            if res.find(i)!=-1:
                continue
            if i in raw_temp:
                while raw_temp[0]!=i:
                    a=raw_temp.popleft()
                    temp.append(a)
                a=raw_temp.popleft()
                temp.append(a)
                b=temp.pop()
                res+=b
            if i in temp:
                while True:
                    c=temp.pop()
                    res+=c
```

基本信息

#:	44024737
题目:	22068
提交人:	23n2300011119 (武)
内存:	3720kB
时间:	30ms
语言:	Python3
提交时间:	2024-03-01 16:49:36

06646: 二叉树的深度

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

思路:

代码

```
class TreeNode:
    def __init__(self):
        self.left=None
        self.right=None
        self.parent=None
def TreeHeight(root):
    if root==None:
        return 0
    return max(TreeHeight(root.left),TreeHeight(root.right))+1

n=int(input())
nodes=[TreeNode() for _ in range(n)]

for node in range(n):
    left,right=map(int,input().split())
    if left!=-1:
        nodes[left-1].parent=nodes[node]
        nodes[node].left=nodes[left-1]
    if right!= -1:
        nodes[node].right=nodes[right-1]
        nodes[right-1].parent=nodes[node]
root=None
for node in nodes:
    if node.parent==None:
        root=node
        break
print(TreeHeight(root))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self):
        self.left=None
        self.right=None
        self.parent=None
def TreeHeight(root):
    if root==None:
        return 0
    return max(TreeHeight(root.left),TreeHeight(root.right))+1

n=int(input())
nodes=[TreeNode() for _ in range(n)]

for node in range(n):
    left,right=map(int,input().split())
    if left!=-1:
        nodes[left-1].parent=nodes[node]
        nodes[node].left=nodes[left-1]
    if right!= -1:
        nodes[node].right=nodes[right-1]
        nodes[right-1].parent=nodes[node]
root=None
for node in nodes:
    if node.parent==None:
        root=node
        break
print(TreeHeight(root))
```

基本信息

#: 44146714
题目: 06646
提交人: 23n2300011119 (武)
内存: 3628kB
时间: 24ms
语言: Python3
提交时间: 2024-03-10 10:09:25

02299: Ultra-QuickSort

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

思路:

如下。

代码

```
# 先善用搜索学习逆序数
# 相邻两个数交换，这种排法等价于冒泡排序
# 原问题等价于问冒泡排序最小交换次数
# 冒泡排序中每交换一次逆序对减少一个
# 排n次后逆序对为0，即排序次数=逆序数。

# 对于a,b两个子数组（已经有序），通过双指针把其中元素加入tmp列表中。因为升序所以加入较小的那个，如果较小的在b中，记为b*,则
# 此时a中的所有数字都和b*构成逆序对，逆序对总数+1

# 如果一个数组已经排完，那么他自身内部将不产生逆序对，同时内部的排序不影响他与其他数组产生的逆序对数。
# 因此，我们可以把一个数组拆分成两段，分别求左数组逆序数，右数组逆序数，两数组组合数的逆序数，相加即结果。
res=0
def Merge(a,start,mid,end):
    tmp,l,r,cnt=[],start,mid+1,0
    while l<=mid and r<=end:
        if a[l]<=a[r]:
            tmp.append(a[l])
            l+=1
        else:
            tmp.append(a[r])
            r+=1
            cnt+=mid+1-l
```

```python
    tmp.extend(a[l:mid+1])
    tmp.extend(a[r:end+1])
    for i in range(start,end+1):
        a[i]=tmp[i-start]
    return cnt

def MergeSort(a,start,end):
    global res
    if start==end:
        return
    mid=(start+end)//2
    MergeSort(a,start,mid)
    MergeSort(a,mid+1,end)
    res+=Merge(a,start,mid,end)


while (n:=int(input()))!=0:
    res,arr=0,[]
    for _ in range(n):
        arr.append(int(input()))
    MergeSort(arr,0,n-1)
    print(res)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44180625提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
# 先善用搜索学习逆序数
# 相邻两个数交换，这种排法等价于冒泡排序
# 原问题等价于问冒泡排序最小交换次数
# 冒泡排序中每交换一次逆序对减少一个
# 排n次后逆序对为0，即排序次数=逆序数。

# 对于a,b两个子数组（已经有序），通过双指针把其中元素加入tmp列表中。因为升序所以加
# 此时a中的所有数字都和b*构成逆序对，逆序对总数+1

# 如果一个数组已经排完，那么他自身内部将不产生逆序对，同时内部的排序不影响他与其他数
# 因此，我们可以把一个数组拆分成两段，分别求左数组逆序数，右数组逆序数，两数组组合数
res=0
def Merge(a,start,mid,end):
    tmp,l,r,cnt=[],start,mid+1,0
    while l<=mid and r<=end:
        if a[l]<=a[r]:
            tmp.append(a[l])
            l+=1
        else:
            tmp.append(a[r])
            r+=1
            cnt+=mid+1-l
    tmp.extend(a[l:mid+1])
    tmp.extend(a[r:end+1])
    for i in range(start,end+1):
        a[i]=tmp[i-start]
    return cnt

def MergeSort(a,start,end):
    global res
    if start==end:
        return
    mid=(start+end)//2
    MergeSort(a,start,mid)
    MergeSort(a,mid+1,end)
    res+=Merge(a,start,mid,end)


while (n:=int(input()))!=0:
    res,arr=0,[]
    for _ in range(n):
        arr.append(int(input()))
    MergeSort(arr,0,n-1)
    print(res)
```

基本信息

#:	44180625
题目:	02299
提交人:	23n2300011119 (武)
内存:	25092kB
时间:	4168ms
语言:	Python3
提交时间:	2024-03-12 11:28:50

English 帮助 关于

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。

Ultra-QuickSort很有难度，缝入了冒泡排序、归并排序，实践上主考递归，理论上主考稳定排序算法。
做了一上午，问题主要出现在：

1. python的引用问题

   对于可变类型a，函数无论是否以其为参数，无论是否声明全局变量，只要函数体内提到对a的操作，都会在内部修改这个a。除非换个名字（引用）拷贝一下。

2. 函数的返回值问题

   跟问题1纠缠在一起，很讨厌。对于该问题，我最终决定让主函数递归地改变全局变量res而不设置自身返回值，区别于题解。

3. 全局变量的设置问题

   对于可变对象，无所谓，见1；

   对于不可变对象，其定义必须出现在全局开头。否则compile error。

对递归的新理解：

递归有一种“信则灵”的性质。写递归陷入泥潭的主要原因是，在头脑中压入第二层甚至更多层递归，即深入地去考虑函数体。

解决办法：

1. 明确函数功能和函数操作，函数操作可以分为纵向操作和横向操作，横向操作是在同一层递归中实现的，而纵向操作是在深层递归中实现的。
2. 先解决纵向操作，即如果子问题已解决，母问题如何利用子问题的答案。
3. 再解决横向操作，即2中的“利用”的具体实现。
4. 最后解决边界问题，即答案定死的问题。

一旦以上四个问题得到解决，不需要人脑考虑深层递归，由数学归纳法即可。

最后：每日选做好难！！快跟不上了！！