Assignment #4: 排序、栈、队列和树

Updated 0005 GMT+8 March 11, 2024

2024 spring, Complied by 同学的姓名、院系

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：macOS Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

05902: 双端队列

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

思路：

模拟运行，简单，注意读题即可

代码

```
tcases=int(input())
ans_lst=[]
for _ in range(tcases):
    n=int(input())
    queue=[]
    for i in range(n):
```

```
        act=[int(x) for x in input().split()]
        type=act[0]
        if type==1:
            x=act[1]
            queue.append(x)
        if type==2:
            c=act[1]
            if c==0:
                queue.pop(0)
            else:
                queue.pop()
    ans_lst.append(queue)
for i in range(tcases):
    if ans_lst[i]:
        out_lst=ans_lst[i]
        for j in range(len(out_lst)):
            out_lst[j]=str(out_lst[j])
        print(' '.join(out_lst))
    else:
        print('NULL')
```

代码运行截图 （至少包含有"Accepted"）

#44252833提交状态 查看

状态: Accepted

源代码

```
tcases=int(input())
ans_lst=[]
for _ in range(tcases):
    n=int(input())
    queue=[]
    for i in range(n):
        act=[int(x) for x in input().split()]
        type=act[0]
        if type==1:
            x=act[1]
            queue.append(x)
        if type==2:
            c=act[1]
            if c==0:
                queue.pop(0)
            else:
                queue.pop()
    ans_lst.append(queue)
for i in range(tcases):
    if ans_lst[i]:
        out_lst=ans_lst[i]
        for j in range(len(out_lst)):
            out_lst[j]=str(out_lst[j])
        print(' '.join(out_lst))
    else:
        print('NULL')
```

基本信息

#: 4425
题目: 0590
提交人: zxk
内存: 4116
时间: 46ms
语言: Pytho
提交时间: 2024

02694: 波兰表达式

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

思路：

逆波兰的逆就是波兰……逆着来顺次入栈，遇到运算符就执行一次运算，最后只剩一个就是答案

代码

```
base_lst = [str(x) for x in input().split()]
def option(astr, num_1, num_2):
    if astr == '*':
        return float(num_1) * float(num_2)
    elif astr == '/':
        return float(num_1) / float(num_2)
    elif astr == '+':
        return float(num_1) + float(num_2)
    else:
        return float(num_1) - float(num_2)

op_lst = ['+', '-', '*', '/']
stack = []
for _ in range(len(base_lst)):
    p=base_lst.pop()
    if p not in op_lst:
        stack.append(p)
    else :
        num_1=stack.pop()
        num_2=stack.pop()
        ad=option(p,num_1,num_2)
        stack.append(ad)
answer = stack[0]
print("%.6f" % answer)
```

代码运行截图（至少包含有"Accepted"）

#44286410提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
base_lst = [str(x) for x in input().split()]
def option(astr, num_1, num_2):
    if astr == '*':
        return float(num_1) * float(num_2)
    elif astr == '/':
        return float(num_1) / float(num_2)
    elif astr == '+':
        return float(num_1) + float(num_2)
    else:
        return float(num_1) - float(num_2)

op_lst = ['+', '-', '*', '/']
stack = []
for _ in range(len(base_lst)):
    p=base_lst.pop()
    if p not in op_lst:
        stack.append(p)
    else :
        num_1=stack.pop()
        num_2=stack.pop()
        ad=option(p,num_1,num_2)
        stack.append(ad)
answer = stack[0]
print("%.6f" % answer)
```

基本信息

#: 44286410
题目: 02694
提交人: zxk
内存: 3632kB
时间: 24ms
语言: Python3
提交时间: 2024-03-18 16:32:46

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:

见调度场算法

核心是顺次读取字符时，究竟是否可以将其输出，需要看后边的运算符的等级是否低，如果低了就可以输出，对于括号其实更多是隔断作用，优先考虑括号内的计算

代码

```
def trans(astr):
    precedence={'+':1,'-':1,'*':2,'/':2}
    stack=[]
    postfix=[]
    number=''

    for char in astr:
        if char in list('1 2 3 4 5 6 7 8 9 0 .'.split()):
            number+=char
        else :
            if number:
                num=float(number)
                postfix.append(int(num) if num.is_integer() else num)
                number=''
            if char in list('+ - * /'.split()):
                while stack and stack[-1] in list('+ - * /'.split()) and
precedence[char]<=precedence[stack[-1]]:
                    postfix.append(stack.pop())
```

```
                stack.append(char)
            elif char=='(':
                stack.append(char)
            elif char==')':
                while stack and stack[-1]!='(':
                    postfix.append(stack.pop())
                stack.pop()

    if number:
        num=float(number)
        postfix.append(int(num) if num.is_integer() else num)
    while stack:
        postfix.append(stack.pop())

    return ' '.join(str(x) for x in postfix)

n=int(input())
anslst=[]
for _ in range(n):
    anslst.append(trans(input()))
for x in anslst:
    print(x)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
def trans(astr):
    precedence={'+':1,'-':1,'*':2,'/':2}
    stack=[]
    postfix=[]
    number=''

    for char in astr:
        if char in list('1 2 3 4 5 6 7 8 9 0 .'.split()):
            number+=char
        else :
            if number:
                num=float(number)
                postfix.append(int(num) if num.is_integer() else num)
                number=''
            if char in list('+ - * /'.split()):
                while stack and stack[-1] in list('+ - * /'.split()) and
                    postfix.append(stack.pop())
                stack.append(char)
            elif char=='(':
                stack.append(char)
            elif char==')':
                while stack and stack[-1]!='(':
                    postfix.append(stack.pop())
                stack.pop()

    if number:
        num=float(number)
        postfix.append(int(num) if num.is_integer() else num)
    while stack:
        postfix.append(stack.pop())

    return ' '.join(str(x) for x in postfix)

n=int(input())
anslst=[]
for _ in range(n):
    anslst.append(trans(input()))
for x in anslst:
    print(x)
```

基本信息

#:	44289597
题目:	24591
提交人:	zxk
内存:	3800kB
时间:	32ms
语言:	Python3
提交时间:	2024-03-18 20:20:20

22068: 合法出栈序列

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

思路：

按照目标输出进行模拟，如果在源列表里就入栈，直到对应元素，如果不在，就必须是栈顶的元素，否则判断为NO，需要优先判断目标列表和源列表是否元素一致

代码

```
def legal(astr):
    alst=[]
    for i in astr:
        alst.append(i)
    blst=xlst.copy()
    t=blst.copy()
    m=alst.copy()
    t.sort()
    m.sort()
    if t!=m:
        return 'NO'
    stack=[]
    while alst:
        p=alst.pop(0)
        if p in blst:
            while blst[0]!=p:
                stack.append(blst.pop(0))
            blst.pop(0)
        if p in stack:
            if stack[-1]!=p:
                return 'NO'
            else :
                stack.pop()
    return 'YES'

x=input()
xlst=[]
for j in x:
    xlst.append(j)
anslst=[]
while True :
    try :
        anslst.append(legal(input()))
    except EOFError:
        break
for j in anslst:
    print(j)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

基本信息

#: 44295172
题目: 22068
提交人: zxk
内存: 3648kB
时间: 26ms
语言: Python3
提交时间: 2024-03-19 10:28:09

源代码

```
def legal(astr):
    alst=[]
    for i in astr:
        alst.append(i)
    blst=xlst.copy()
    t=blst.copy()
    m=alst.copy()
    t.sort()
    m.sort()
    if t!=m:
        return 'NO'
    stack=[]
    while alst:
        p=alst.pop(0)
        if p in blst:
            while blst[0]!=p:
                stack.append(blst.pop(0))
            blst.pop(0)
        if p in stack:
            if stack[-1]!=p:
                return 'NO'
            else :
                stack.pop()
    return 'YES'

x=input()
xlst=[]
for j in x:
    xlst.append(j)
anslst=[]
while True :
    try :
        anslst.append(legal(input()))
    except EOFError:
        break
for j in anslst:
    print(j)
```

06646: 二叉树的深度

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

思路:

建立一个节点的列表，定义左右节点编号，递归取左右最大的深度

代码

```
n=int(input())
node_lst=[]
for _ in range(n):
    node_lst.append([int(x) for x in input().split()])

class TreeNode:
    def __init__(self):
        self.left=None
        self.right=None

def tree_depth(node):
    if node is None:
        return 0
    left_depth=tree_depth(node.left)
```

```
    right_depth=tree_depth(node.right)
    return max(left_depth,right_depth)+1

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

for i in range(n):
    left_index,right_index=node_lst[i][0],node_lst[i][1]
    if left_index!=-1:
        nodes[i].left=nodes[left_index-1]
    if right_index!=-1:
        nodes[i].right=nodes[right_index-1]

root=nodes[0]
depth=tree_depth(root)
print(depth)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44216267提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n=int(input())
node_lst=[]
for _ in range(n):
    node_lst.append([int(x) for x in input().split()])

class TreeNode:
    def __init__(self):
        self.left=None
        self.right=None

def tree_depth(node):
    if node is None:
        return 0
    left_depth=tree_depth(node.left)
    right_depth=tree_depth(node.right)
    return max(left_depth,right_depth)+1

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

for i in range(n):
    left_index,right_index=node_lst[i][0],node_lst[i][1]
    if left_index!=-1:
        nodes[i].left=nodes[left_index-1]
    if right_index!=-1:
        nodes[i].right=nodes[right_index-1]

root=nodes[0]
depth=tree_depth(root)
print(depth)
```

基本信息

#: 44216267
题目: 06646
提交人: zxk
内存: 3656kB
时间: 26ms
语言: Python3
提交时间: 2024-03-14 18:44:34

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

02299: Ultra-QuickSort

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

思路:

归并排序mergesort，利用指针，global和分治递归，首先将原列表对半拆分，递归排每部分的序，对于这个函数来说，把变量改写为排好序的列表，然后移动整体列表的指针，对于左右列表，在左右第一个选小的放入，如果放左边的是没问题的，如果放右边的就需要挨个交换，换几次加几次，最后如果还有剩下的元素再依次放入，但是不存在交换

代码

```
count=0
def Ultra_QuickSort(alst):
    global count
    if len(alst)>1:
        mid=len(alst)//2
        left=alst[:mid]
        right=alst[mid:]

        Ultra_QuickSort(left)
        Ultra_QuickSort(right)

        L=R=p=0
        while len(left)>L and len(right)>R:
            if left[L]<=right[R]:
                alst[p]=left[L]
                L+=1
            else:
                alst[p]=right[R]
                R+=1
                count+=len(left)-L
            p+=1
        while len(left)>L:
            alst[p]=left[L]
            p+=1
            L+=1
        while len(right)>R:
            alst[p]=right[R]
            p+=1
            R+=1

data_lst=[]
while True :
    n = int(input())
    base_lst = []
    if n == 0 :
        break
    else :
        for i in range(n):
            base_lst.append(int(input()))
    data_lst.append(base_lst)
#导入数据为二维列表，每一行是一组数据
for data in data_lst:
    count=0
    Ultra_QuickSort(data)
    print(count)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态：Accepted

源代码

```
count=0
def Ultra_QuickSort(alst):
    global count
    if len(alst)>1:
        mid=len(alst)//2
        left=alst[:mid]
        right=alst[mid:]

        Ultra_QuickSort(left)
        Ultra_QuickSort(right)

        L=R=p=0
        while len(left)>L and len(right)>R:
            if left[L]<=right[R]:
                alst[p]=left[L]
                L+=1
            else:
                alst[p]=right[R]
                R+=1
                count+=len(left)-L
            p+=1
        while len(left)>L:
            alst[p]=left[L]
            p+=1
            L+=1
        while len(right)>R:
            alst[p]=right[R]
            p+=1
            R+=1

data_lst=[]
while True :
    n = int(input())
    base_lst = []
    if n == 0 :
        break
    else :
        for i in range(n):
            base_lst.append(int(input()))
    data lst.append(base lst)
```

基本信息

#:	44297778
题目:	02299
提交人:	zxk
内存:	32840kB
时间:	4297ms
语言:	Python3
提交时间:	2024-03-19 15:13:10

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。

学习了树的各种编码，感觉好麻烦啊，不过对类的用法有了一点感觉，更能理解了

1.双端队列送分

2.波兰表达式头一次做，先尝试了顺着入栈出栈，但是问题在于什么时候执行计算，判断起来非常麻烦，很难实现，各种报错，死循环等等；后来转念一想，逆波兰的逆不就是波兰吗？逆波兰简单，为啥不倒着入栈呢……所以一下就过了。感觉问题出在判据上，根据入栈的顺序，运算符和数字都可能在栈里，只有凑够两个数字才可以执行计算，但是计算结果入栈后栈当前的情况如何还需要再考虑，直到栈里最后时运算符+数字才能进行下一轮操作，一是时间复杂度高，二是确实难敲出来；逆波兰是只有数字在栈里，按运算符执行计算，只有一个判据，栈内元素种类单一，更适合线性结构体质，非常之高效啊！

3.通过中序转后序理解了调度场算法（到底怎么想到的，反正我是想不到，需要加到cheating paper！）这个调度场算法其实也是一定程度上对于一个人思路的复刻，比如我们在顺次判断如何转化为后序表达式时要以当前运算符优先级低于上一个运算符来判定输出，思路还是挺好理解的，只是转化成代码语言也太考验人了吧

4.合法出栈序列开始的时候想不出来怎么做，后来明白了其实他应该有的操作时固定的，并不需要选择，只需要顺次模拟，如果没法继续进行就返回NO即可，但是这个数据坑在部分序列不是合法序列

5.二叉树深度没啥说的，抄讲义

6.超快排看的老师的答案。最开始的思路是两两分开对换，然后如果交界处有问题就重新排序，这个方法太笨重了，有很多浪费的递归，于是写了cache，不过还是超时，然后转念一想用栈，思路是两个栈来回倒，入下一个在栈顶的前后，取决于大小，然后一直循环直到排序完成，但是显然依旧超时。这两个思路虽然都对，但是时间复杂度一致得长到离谱，用了将近一分钟才跑完，肯定是不合适吧！看了老师得各种指针，才明白其实这个和mergesort没区别，只是用逆序数相关的数学知识弥补了时间上的差距。首先，这个语法功底目前我是达不到的...尤其是对于输入输出的设计；其次，我也不清楚这个逆序数的数学原理，感觉对于一个新题我还是不希望这么做，更倾向于简单思路的模拟；再次，毕竟oj是AC至上；结论：开抄！

这次作业好难啊，其他作业也好难啊，怎么这么难啊……