Assignment #4: 排序、栈、队列和树

Updated 2031 GMT+8 March 11, 2024

2024 spring, Complied by 钟明衡 物理学院

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

操作系统：Windows_NT x64 10.0.19045

Python编程环境：Visual Studio Code 1.76.1

C/C++编程环境：Visual Studio Code 1.76.1

1. 题目

05902: 双端队列

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

思路：

用list模拟双端队列，入队直接用append即可，用一个指针来标记队首位置，先判断是否能出队，队尾出队直接pop，队首出队则指针+1

代码

```python
for _ in range(int(input())):
    l, i = [], 0
    for __ in range(int(input())):
        a, b = input().split()
        if a == '1':
            l.append(b)
        elif i < len(l):
            if b == '0':
                i += 1
            else:
                l.pop()
    print(' '.join(l[i:]) if i < len(l) else 'NULL')

```

代码运行截图

#44167538提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
for _ in range(int(input())):
    l, i = [], 0
    for __ in range(int(input())):
        a, b = input().split()
        if a == '1':
            l.append(b)
        elif i < len(l):
            if b == '0':
                i += 1
            else:
                l.pop()
    print(' '.join(l[i:]) if i < len(l) else 'NULL')
```

基本信息

#: 44167538
题目: 05902
提交人: 23n2300011505(12号娱乐选手)
内存: 3636kB
时间: 37ms
语言: Python3
提交时间: 2024-03-11 11:47:01

02694: 波兰表达式

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

思路:

如果出现一个运算符号，则寻找后面的两个波兰表达式进行运算，第一个表达式起始位置就是下一位，而第二个需要按顺序往下查找，如果是运算符号count就+1，否则-1，count=0时即找到第二个表达式的位置

最后输出在0位置开始的波兰表达式即可

代码

```python
s = []


def next(m):
    global s
```

```
    count = 1
    for i in range(m+1, len(s)):
        if count == 0:
            return i
        if s[i] == '+' or s[i] == '-' or s[i] == '*' or s[i] == '/':
            count += 1
        else:
            count -= 1


def poland(n):
    global s
    if s[n] == '+':
        return poland(n+1)+poland(next(n))
    elif s[n] == '-':
        return poland(n+1)-poland(next(n))
    elif s[n] == '*':
        return poland(n+1)*poland(next(n))
    elif s[n] == '/':
        return poland(n+1)/poland(next(n))
    else:
        return float(s[n])


s = input().split()
print('%.6f' % (poland(0)))

```

代码运行截图

#41616605提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 41616605
题目: 02694
提交人: 23n2300011505(12号娱乐选手)
内存: 3616kB
时间: 34ms
语言: Python3
提交时间: 2023-10-12 23:31:24

源代码

```
s = []


def next(m):
    global s
    count = 1
    for i in range(m+1, len(s)):
        if count == 0:
            return i
        if s[i] == '+' or s[i] == '-' or s[i] == '*' or s[i] == '/':
            count += 1
        else:
            count -= 1


def poland(n):
    global s
    if s[n] == '+':
        return poland(n+1)+poland(next(n))
    elif s[n] == '-':
        return poland(n+1)-poland(next(n))
    elif s[n] == '*':
        return poland(n+1)*poland(next(n))
    elif s[n] == '/':
        return poland(n+1)/poland(next(n))
    else:
        return float(s[n])


s = input().split()
print('%.6f' % (poland(0)))
```

24591: 中序表达式转后序表达式

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

思路:

首先是输入，将输入的字符串逐个判断，如果是数字（0~9，小数点）就继续判断，将整个数字储存，否则直接储存这个字符

接下来用栈来进行运算，栈A是最后答案的顺序，s是一个辅助栈，用来匹配括号以及处理运算符，具体操作如下:

s出栈的顺序就是运算符被执行的顺序，当出现右括号就将s的尾端全部弹出，直到找到对应的左括号

在同一个括号下，乘除比加减优先，同级运算左边的优先，优先运算优先从s中弹出进入A

代码

```
for _ in range(int(input())):
    t, l, s, A = '', [], [], []
    p = {'+': 0, '-': 0, '*': 1, '/': 1}
    for a in input().strip():
        if '0' <= a <= '9' or a == '.':
            t += a
```

```
        else:
            if t:
                l.append(t)
                t = ''
            l.append(a)
    if t:
        l.append(t)
    for a in l:
        try:
            b = float(a)
            A.append(a)
        except ValueError:
            if a == '(':
                s.append(a)
            elif a == ')':
                while s and s[-1] != '(':
                    A.append(s.pop())
                s.pop()
            else:
                while s and s[-1] != '(' and p[s[-1]] >= p[a]:
                    A.append(s.pop())
                s.append(a)
    while s:
        A.append(s.pop())
    print(' '.join(A))

```

代码运行截图

#44167598提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
for _ in range(int(input())):
    t, l, s, A = '', [], [], []
    p = {'+': 0, '-': 0, '*': 1, '/': 1}
    for a in input().strip():
        if '0' <= a <= '9' or a == '.':
            t += a
        else:
            if t:
                l.append(t)
                t = ''
            l.append(a)
    if t:
        l.append(t)
    for a in l:
        try:
            b = float(a)
            A.append(a)
        except ValueError:
            if a == '(':
                s.append(a)
            elif a == ')':
                while s and s[-1] != '(':
                    A.append(s.pop())
                s.pop()
            else:
                while s and s[-1] != '(' and p[s[-1]] >= p[a]:
                    A.append(s.pop())
                s.append(a)
    while s:
        A.append(s.pop())
    print(' '.join(A))
```

基本信息

#: 44167598
题目: 24591
提交人: 23n2300011505(12号娱乐选手)
内存: 3684kB
时间: 31ms
语言: Python3
提交时间: 2024-03-11 11:54:59

22068: 合法出栈序列

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

思路:

比某个元素先出栈的，一定要比它先入栈，用这个条件去判断就够了

注意，可能会出现出栈数量和入栈数量不同，以及出了不存在的元素的情况，要先排除

代码

```
S = input()
n = len(S)
d = {S[i]: i for i in range(n)}
while True:
    try:
        s = input()
    except EOFError:
        break
    f = len(s) != n
    for a in s:
        try:
```

```
            b = d[a]
        except KeyError:
            f = True
            break
    for i in range(n-1):
        if f:
            break
        a = d[s[i]]
        for j in range(i+1, n):
            b = d[s[j]]
            if b < d[s[i]]:
                if b < a:
                    a = b
                else:
                    f = True
                    break
    print('NO' if f else 'YES')

```

代码运行截图

#44076112提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
S = input()
n = len(S)
d = {S[i]: i for i in range(n)}
while True:
    try:
        s = input()
    except EOFError:
        break
    f = len(s) != n
    for a in s:
        try:
            b = d[a]
        except KeyError:
            f = True
            break
    for i in range(n-1):
        if f:
            break
        a = d[s[i]]
        for j in range(i+1, n):
            b = d[s[j]]
            if b < d[s[i]]:
                if b < a:
                    a = b
                else:
                    f = True
                    break
    print('NO' if f else 'YES')
```

基本信息

#: 44076112
题目: 22068
提交人: 23n2300011505(12号娱乐选手)
内存: 3600kB
时间: 33ms
语言: Python3
提交时间: 2024-03-05 15:16:23

06646: 二叉树的深度

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

思路：

用两个列表来存储每个节点左右子树的索引，判断深度用dfs进行先序遍历

代码

```
ans, l, r = 1, [-1], [-1]


def dfs(n, count):
    global ans, l, r
    if l[n] != -1:
        dfs(l[n], count+1)
    if r[n] != -1:
        dfs(r[n], count+1)
    ans = max(ans, count)


n = int(input())
for i in range(n):
    a, b = map(int, input().split())
    l.append(a)
    r.append(b)
dfs(1, 1)
print(ans)

```

代码运行截图

#44167630提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
ans, l, r = 1, [-1], [-1]


def dfs(n, count):
    global ans, l, r
    if l[n] != -1:
        dfs(l[n], count+1)
    if r[n] != -1:
        dfs(r[n], count+1)
    ans = max(ans, count)


n = int(input())
for i in range(n):
    a, b = map(int, input().split())
    l.append(a)
    r.append(b)
dfs(1, 1)
print(ans)
```

基本信息

#: 44167630
题目: 06646
提交人: 23n2300011505(12号娱乐选手)
内存: 3632kB
时间: 22ms
语言: Python3
提交时间: 2024-03-11 11:57:22

02299: Ultra-QuickSort

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

思路:

每次交换相邻的两个数，逆序对数正好减少1，可见结果就是原序列的逆序对数，可以用归并排序来求

代码

```python
def merge_sort(l):
    if len(l) <= 1:
        return l, 0
    mid = len(l) // 2
    left, left_count = merge_sort(l[:mid])
    right, right_count = merge_sort(l[mid:])
    l, merge_count = merge(left, right)
    return l, left_count + right_count + merge_count


def merge(left, right):
    merged = []
    left_index, right_index = 0, 0
    count = 0
    while left_index < len(left) and right_index < len(right):
        if left[left_index] <= right[right_index]:
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1
```

```
            count += len(left) - left_index
    merged += left[left_index:]+right[right_index:]
    return merged, count


while True:
    n = int(input())
    l = []
    if n:
        for i in range(n):
            l.append(int(input()))
    else:
        break
    l, ans = merge_sort(l)
    print(ans)

```

代码运行截图

#44173657提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def merge_sort(l):
    if len(l) <= 1:
        return l, 0
    mid = len(l) // 2
    left, left_count = merge_sort(l[:mid])
    right, right_count = merge_sort(l[mid:])
    l, merge_count = merge(left, right)
    return l, left_count + right_count + merge_count


def merge(left, right):
    merged = []
    left_index, right_index = 0, 0
    count = 0
    while left_index < len(left) and right_index < len(right):
        if left[left_index] <= right[right_index]:
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1
            count += len(left) - left_index
    merged += left[left_index:]+right[right_index:]
    return merged, count


while True:
    n = int(input())
    l = []
    if n:
        for i in range(n):
            l.append(int(input()))
    else:
        break
    l, ans = merge_sort(l)
    print(ans)
```

基本信息

#: 44173657
题目: 02299
提交人: 23n2300011505(12号娱乐选手)
内存: 32220kB
时间: 3915ms
语言: Python3
提交时间: 2024-03-11 19:57:18

2. 学习总结和收获

通过本次作业加深了对栈、树的理解

最后一题用到了归并排序（第一次用是上学期，CF上的一道题D. Yet Another Inversions Problem，题意比这个更复杂，但本质上还是算逆序对数），我对归并排序的理解如下：

归并排序就是把数组分成左右两半分别排好序（递归，用同样的方法），然后依次弹出左右数组中最小的元素（即两个数组首个元素较小的那个，优先左边）操作直到左右有一个空了即结束，这一步就叫做归并。如果右边打头的元素比左边的小，则左边还没输出的元素都比这个元素小，逆序对数要增加当前左边剩余元素个数。这样，在排好序的同时，能够计算原序列的逆序对数。

每次归并的时间复杂度为$O(n)$，而二分要进行$\log n$次，故归并排序的时间复杂度为$O(n \log n)$，相比正常会想到的$O(n^2)$方法要更快。

作业的最后一题可以作为归并排序的模板。