Assignment #4: 排序、栈、队列和树

Updated 0005 GMT+8 March 11, 2024

2024 spring, Complied by 李佳霖，心理与认知科学学院

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：masOS Sonoma

Python编程环境：VScode

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：用列表实现双端队列类

代码

```
#
class deque():
    def __init__(self):
        self.items = []

    def addFront(self, item):
        self.items.insert(0, item)

    def addRear(self, item):
        self.items.append(item)

    def removeFront(self):
        return self.items.pop(0)

    def removeRear(self):
        return self.items.pop()

    def isEmpty(self):
        return self.items == []

    def size(self):
        return len(self.items)

t = int(input())
for i in range(t):
    n = int(input())
    d = deque()
    for j in range(n):
        type, value = map(int, input().split())
        if type == 1:
            d.addRear(value)
        elif type == 2:
            if value == 0:
                d.removeFront()
            else:
                d.removeRear()
    print('NULL' if d.isEmpty() else ' '.join(map(str, d.items)))
```

代码运行截图 （至少包含有"Accepted"）

#44253413提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class deque():
    def __init__(self):
        self.items = []

    def addFront(self, item):
        self.items.insert(0, item)
```

基本信息

#: 44253413
题目: 05902
提交人: 李佳霖2000013713
内存: 3664kB
时间: 45ms
语言: Python3
提交时间: 2024-03-16 19:19:55

02694: 波兰表达式

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

思路：和逆波兰表达式的唯一区别就是，遍历的时候是从后向前

代码

```
#
class Stack():
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def isEmpty(self):
        return self.items == []

    def size(self):
        return len(self.items)

a = list(input().split())
f = ['+', '-', '*', '/']
s = Stack()
for i in range(len(a)):
    if a[i] not in f:
        a[i] = float(a[i])
```

```
for i in range(len(a), 0, -1):
    if a[i-1] in f:
        if a[i-1] == '+':
            s.push(s.pop() + s.pop())
        elif a[i-1] == '-':
            s.push(s.pop() - s.pop())
        elif a[i-1] == '*':
            s.push(s.pop() * s.pop())
        elif a[i-1] == '/':
            s.push(s.pop() / s.pop())
    else:
        s.push(a[i-1])
print("{:.6f}".format(float(s.pop())))
```

代码运行截图 （至少包含有"Accepted"）

#44254565提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class Stack():
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)
```

基本信息

#: 44254565
题目: 02694
提交人: 李佳霖2000013713
内存: 3556kB
时间: 23ms
语言: Python3
提交时间: 2024-03-16 19:56:55

24591: 中序表达式转后序表达式

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

思路：参考CSDN上的流程实现

1. 创建用于保存运算符的空栈opStack，以及一个用于保存结果的空列表。
2. 使用字符串方法split将输入的中序表达式转换成一个列表。
3. 从左到右扫描标记列表。

- 如果标记是操作数，将其添加到结果列表的末尾。
- 如果标记是左括号，将其压入opStack栈中。
- 如果标记是右括号，反复从opStack栈中移除元素，直到移除对应的左括号。将从栈中取出的每一个运算符都添加到结果列表的末尾。
- 如果标记是运算符，将其压入opStack栈中。但是，在这之前，需要先从栈中取出优先级更高或相同的运算符，并将它们添加到结果列表的末尾。

4.当处理完输入表达式以后，检查opStack。将其中所有残留的运算符全部添加到结果列表的末尾。

代码

```
#
import re
class Stack():
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def isEmpty(self):
        return self.items == []

    def size(self):
        return len(self.items)

n = int(input())
f = ['+', '-', '*', '/', '(', ')']
operator = Stack()
for i in range(n):
    num = []
    expression = input()
    exp = re.findall(r'\d+\.\d+|\d+|[-+*/()]', expression) # expression
    for j in range(len(exp)):
```

```
        if exp[j] in f:
            if exp[j] == ')':
                while True:
                    op = operator.pop()
                    if op == '(':
                        break
                    num.append(op)
            else:
                if exp[j] == '+' or exp[j] == '-':
                    while not operator.isEmpty() and operator.items[-1] != '(':
                        num.append(operator.pop())
                elif exp[j] == '*' or exp[j] == '/':
                    while not operator.isEmpty() and (operator.items[-1] == '*' or
operator.items[-1] == '/'):
                        num.append(operator.pop())
                operator.push(exp[j])

        else:
            num.append(exp[j])

    while not operator.isEmpty():
        num.append(operator.pop())
    print(' '.join(num))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44255517提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import re
class Stack():
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)
```

基本信息

#:	44255517
题目:	24591
提交人:	李佳霖2000013713
内存:	3912kB
时间:	33ms
语言:	Python3
提交时间:	2024-03-16 20:28:05

22068: 合法出栈序列

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

思路：gw老师的ppt代码，相当于直接模拟了可能的出栈序列，首先定义了两个指针分别对应s1和s2。当栈顶元素和序列中的字符匹配时，则可以直接弹出，否则压入栈中，最后检查栈中反序的字符串是否和s2剩余的字符串匹不匹配即可。stack[::-1]这样的写法值得学习。

代码

```
#
def isPopSeq(s1,s2):#判断s2是不是s1经出入栈得到的出栈序列
  stack = []
  if len(s1) != len(s2):
    return False
  else:
    L = len(s1)
    stack.append(s1[0])
    p1,p2 = 1,0 #p1指向s1,p2指向s2
    while p1 < L:
      if len(stack) > 0 and stack[-1] == s2[p2]: #栈顶元素与s2[p2]匹配
        stack.pop()
        p2 += 1
      else:
        stack.append(s1[p1])
        p1 += 1
    return "".join(stack[::-1]) == s2[p2:] #判断栈中剩余元素与s2[p2:]是否匹配

s1 = input()
while True:
  try:
    s2 = input() #如果输入数据结束，再执行input()会产生异常
  except:
    break  #输入数据结束
  if isPopSeq(s1,s2):
    print("YES")
  else:
    print("NO")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44277951提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
def isPopSeq(s1,s2):#判断s2是不是s1经出入栈得到的出栈序列
        stack = []
        if len(s1) != len(s2):
                return False
        else:
                L = len(s1)
                stack.append(s1[0])
```

基本信息

#: 44277951
题目: 22068
提交人: 李佳霖2000013713
内存: 3612kB
时间: 26ms
语言: Python3
提交时间: 2024-03-17 20:27:50

06646: 二叉树的深度

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

思路：参照老师讲义，又自己复刻了一遍

代码

```
#
class TreeNode():
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None

def tree_depth(root):
    if not root:
        return 0
    left = tree_depth(root.left)
    right = tree_depth(root.right)
    return max(left, right) + 1

n = int(input())
nodes = [TreeNode(i) for i in range(n)]
for i in range(n):
    left, right = map(int, input().split())
    if left != -1:
        nodes[i].left = nodes[left-1]
    if right != -1:
        nodes[i].right = nodes[right-1]

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

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44279446提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode():
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None

def tree_depth(root):
```

基本信息

#: 44279446
题目: 06646
提交人: 李佳霖2000013713
内存: 3652kB
时间: 24ms
语言: Python3
提交时间: 2024-03-17 21:35:48

02299: Ultra-QuickSort

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

思路：归并排序。将序列拆分成一个个最小子单元序列，然后排序，最后得到排序完的序列。

代码

```
#
def merge_and_count(arr, left, mid, right):
    swaps = 0
    temp = [0] * (right - left + 1)
    i, j, k = left, mid + 1, 0 # i is the index for the left subarray, j is the index for the
right subarray, k is the index for the temp[] array

    # Merge the two parts into temp[]
    while i <= mid and j <= right:
        if arr[i] <= arr[j]:
            temp[k] = arr[i]
            i += 1
        else:
            temp[k] = arr[j]
            j += 1
            swaps += (mid - i + 1)  # Count the number of swaps
        k += 1

    # Add the remaining elements in left part
    while i <= mid:

        temp[k] = arr[i]
```

```
            temp[k] = arr[i]
            k += 1
            i += 1

        # Add the remaining elements in right part
        while j <= right:
            temp[k] = arr[j]
            k += 1
            j += 1

        # Copy back the merged elements to original array
        for i in range(left, right + 1):
            arr[i] = temp[i - left]

        return swaps

    def merge_sort(arr, left, right):
        swaps = 0
        if left < right:
            mid = (left + right) // 2
            swaps += merge_sort(arr, left, mid)
            swaps += merge_sort(arr, mid + 1, right)
            swaps += merge_and_count(arr, left, mid, right)
        return swaps

    while True:
        n = int(input())
        if n == 0:
            break
        sequence = [int(input()) for _ in range(n)]
        swaps = merge_sort(sequence, 0, n - 1)
        print(swaps)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44281132提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
def merge_and_count(arr, left, mid, right):
    swaps = 0
    temp = [0] * (right - left + 1)
    i, j, k = left, mid + 1, 0

    while i <= mid and j <= right:
        if arr[i] <= arr[j]:
```

基本信息

#:	44281132
题目:	02299
提交人:	李佳霖2000013713
内存:	26728kB
时间:	4032ms
语言:	Python3
提交时间:	2024-03-17 23:22:33

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。

相比于之前学计概经常需要一些非常灵光一现的想法的时候，感觉自己更适合学这种相对而言比较系统一点的数据结构这种东西，虽然需要掌握的知识很多很多，相比于群里的各位大佬还差的很远，但仍需要加油。

打算这周把排序问题系统性的过一遍，有时间的话把树结构也过一遍。