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==

罗瑞哲 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

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：

代码

```
#
t=int(input())
for i in range(t):
    lst=[]
    n=int(input())
    for j in range(n):
```

```
        tp,x=map(int,input().split())
        if tp==1:
            lst.append(x)
        if tp==2:
            if x==0:
                del lst[0]
            if x==1:
                del lst[-1]
    if len(lst)==0:
        print("NULL")
    else:
        print(" ".join(map(str,lst)))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44279804提交状态

状态: Accepted

源代码

```
t=int(input())
for i in range(t):
    lst=[]
    n=int(input())
    for j in range(n):
        tp,x=map(int,input().split())
        if tp==1:
            lst.append(x)
        if tp==2:
            if x==0:
                del lst[0]
            if x==1:
                del lst[-1]
    if len(lst)==0:
        print("NULL")
    else:
        print(" ".join(map(str,lst)))
```

02694: 波兰表达式

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

思路:

代码

```
#
expression = input().split()
```

```
stack = []
while expression:
    a = expression.pop(-1)
    if a in ['+', '-', '*', '/']:
        c = stack.pop(-1)
        d = stack.pop(-1)
        if a == '+':
            stack.append(c + d)
        elif a == '-':
            stack.append(c - d)
        elif a == '*':
            stack.append(c * d)
        else:
            stack.append(c / d)
    else:
        stack.append(float(a))

print("{:.6f}".format(stack[0]))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44289481提交状态

状态: Accepted

源代码

```
expression = input().split()
stack = []
while expression:
    a = expression.pop(-1)
    if a in ['+', '-', '*', '/']:
        c = stack.pop(-1)
        d = stack.pop(-1)
        if a == '+':
            stack.append(c + d)
        elif a == '-':
            stack.append(c - d)
        elif a == '*':
            stack.append(c * d)
        else:
            stack.append(c / d)
    else:
        stack.append(float(a))

print("{:.6f}".format(stack[0]))
```

24591: 中序表达式转后序表达式

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

思路:

代码

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

    for char in expression:
        if char.isnumeric() or char == '.':
            number += char
        else:
            if number:
                num = float(number)
                postfix.append(int(num) if num.is_integer() else num)
                number = ''
            if char in '+-*/':
                while stack and stack[-1] in '+-*/' 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())
for _ in range(n):
    expression = input()
    print(infix_to_postfix(expression))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44281117提交状态

状态: Accepted

源代码

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

    for char in expression:
        if char.isnumeric() or char == '.':
            number += char
        else:
            if number:
                num = float(number)
                postfix.append(int(num) if num.is_integer() else num)
                number = ''
            if char in '+-*/':
                while stack and stack[-1] in '+-*/' and precedence[char]
                    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())
for _ in range(n):
    expression = input()
    print(infix_to_postfix(expression))
```

22068: 合法出栈序列

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

思路:

代码

```
#
def is_valid_pop_sequence(origin, output):
    if len(origin) != len(output):
        return False
    stack = []
    bank = list(origin)
    for char in output:
        while (not stack or stack[-1] != char) and bank:
            stack.append(bank.pop(0))
        if not stack or stack[-1] != char:
            return False
        stack.pop()
    return True
origin = input().strip()
while True:
    try:
        output = input().strip()
        if is_valid_pop_sequence(origin, output):
            print('YES')
        else:
            print('NO')
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44289525提交状态

状态: Accepted

源代码

```
def is_valid_pop_sequence(origin, output):
    if len(origin) != len(output):
        return False
    stack = []
    bank = list(origin)
    for char in output:
        while (not stack or stack[-1] != char) and bank:
            stack.append(bank.pop(0))
        if not stack or stack[-1] != char:
            return False
        stack.pop()
    return True
origin = input().strip()
while True:
    try:
        output = input().strip()
        if is_valid_pop_sequence(origin, output):
            print('YES')
        else:
            print('NO')
    except EOFError:
        break
```

06646: 二叉树的深度

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

思路:

代码

```
#
class TreeNode:
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None
def build_tree(node_list):
    nodes = {i: TreeNode(i) for i in range(1, len(node_list) + 1)}
    for i, (left, right) in enumerate(node_list, 1):
        if left != -1:
            nodes[i].left = nodes[left]
        if right != -1:
            nodes[i].right = nodes[right]
    return nodes[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())
node_list = []
for _ in range(n):
    left, right = map(int, input().split())
    node_list.append((left, right))
root = build_tree(node_list)
depth = max_depth(root)
print(depth)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44289548提交状态

状态: Accepted

源代码

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

02299: Ultra-QuickSort

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

思路:

代码

```
#
def merge_sort(lst):
    if len(lst) <= 1:
        return lst, 0
    middle = len(lst) // 2
    left, inv_left = merge_sort(lst[:middle])
    right, inv_right = merge_sort(lst[middle:])
    merged, inv_merge = merge(left, right)
    return merged, inv_left + inv_right + inv_merge
def merge(left, right):
    merged = []
    inv_count = 0
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            merged.append(left[i])
            i += 1
        else:
            merged.append(right[j])
            j += 1
            inv_count += len(left) - i
    merged += left[i:]
    merged += right[j:]
    return merged, inv_count
while True:
    n = int(input())
    if n == 0:
        break
    lst = []
    for _ in range(n):
        lst.append(int(input()))
    _, inversions = merge_sort(lst)
    print(inversions)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44289588提交状态

状态: Accepted

源代码

```
def merge_sort(lst):
    if len(lst) <= 1:
        return lst, 0
    middle = len(lst) // 2
    left, inv_left = merge_sort(lst[:middle])
    right, inv_right = merge_sort(lst[middle:])
    merged, inv_merge = merge(left, right)
    return merged, inv_left + inv_right + inv_merge
def merge(left, right):
    merged = []
    inv_count = 0
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            merged.append(left[i])
            i += 1
        else:
            merged.append(right[j])
            j += 1
            inv_count += len(left) - i
    merged += left[i:]
    merged += right[j:]
    return merged, inv_count
while True:
    n = int(input())
    if n == 0:
        break
    lst = []
    for _ in range(n):
        lst.append(int(input()))
    _, inversions = merge_sort(lst)
    print(inversions)
```

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。==

我上学期计概学得不是很扎实，这学期要抓紧赶上，涉及到我不太了解的数据结构和算法我会尽快补学上。