

Assignment #4: 排序、栈、队列和树

Updated 0005 GMT+8 March 11, 2024

2024 spring, Compiled 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:

调用deque即可

代码

```
1 '''
2 2100017810 刘思瑞
3 '''
4 from queue import deque
5
6 def imple(a,b):
```

```
7     global d
8     if a ==1:
9         d.append(b)
10    else:
11        if b:
12            d.pop()
13        else:
14            d.popleft()
15
16    n = int(input())
17    for i in range(n):
18        num = int(input())
19        d = deque()
20        for j in range(num):
21            a,b = map(int,input().split())
22            imple(a,b)
23        if d:
24            for i in d:
25                print(i,end=' ')
26        else:
27            print('NULL',end=' ')
28    print()
```

代码运行截图

状态: Accepted

源代码

```
'''
2100017810 刘思瑞
'''

from queue import deque

def imple(a,b):
    global d
    if a ==1:
        d.append(b)
    else:
        if b:
            d.pop()
        else:
            d.popleft()

n = int(input())
for i in range(n):
    num = int(input())
    d = deque()
    for j in range(num):
        a,b = map(int,input().split())
        imple(a,b)
    if d:
        for i in d:
            print(i,end=' ')
    else:
        print('NULL',end=' ')
    print()
```

02694: 波兰表达式

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

思路:

前面是用递归写的, 直接贴了上学期的递归的程序, 后面第二部分用stack实现了, 确实正向的思维简化了许多

代码

```
1  '''
2  刘思瑞 2100017810
3  '''
4  def calcu(calculate,i,j):
5      global calcul1
6      if calculate[i+1] not in calcul1:
7          if calculate[i+2] not in calcul1:
8              calculate[i] =
str(eval(calculate[i+1]+calculate[i]+calculate[i+2]))
```

```
9         del calculate[i+1]
10        del calculate[i+1]
11        i = j[-1]
12        j = j[:-1]
13    else:
14        j.append(i)
15        i = i+2
16    else:
17        j.append(i)
18        i = i+1
19    return calculate, i, j
20
21
22    calcul1 = ['+', '-', '*', '/']
23    calculate = list(input().split())
24    i = 0
25    j = [0]
26    while True:
27        calculate, i ,j = calcu(calculate,i,j)
28        if len(calculate) == 1:
29            break
30    print('%.6f' % float(calculate[0]))
```

代码运行截图

状态: Accepted

源代码

```
'''
刘思瑞 2100017810
'''
def calcu( calculate, i, j ):
    global calcul
    if calculate[i+1] not in calcul:
        if calculate[i+2] not in calcul:
            calculate[i] = str(eval( calculate[i+1]+calculate[i]+calculate[i+2] ))
            del calculate[i+1]
            del calculate[i+1]
            i = j[-1]
            j = j[:-1]
        else:
            j.append(i)
            i = i+2
    else:
        j.append(i)
        i = i+1
    return calculate, i, j

calcul = ['+', '-', '*', '/']
calculate = list(input().split())
i = 0
j = [0]
while True:
    calculate, i, j = calcu( calculate, i, j )
    if len( calculate ) == 1:
        break
print( '%.6f' % float( calculate[0] ) )
```

代码

```
1 '''
2 2100017810 刘思瑞
3 '''
4 from queue import deque
5 def calcu():
6     global m,n
7     a = m.pop()
8     if a in ('+', '-', '*', '/'):
9         b = n.pop()
10        c = n.pop()
11        n.append(str(eval(b+a+c)))
12    else:
13        n.append(a)
14
15 s = list(input().split())
16 m = deque(s)
17 n = deque()
18 while True:
```

```

19     calcu()
20     if not m:
21         break
22 print('%.6f' %float(n[0]))

```

代码运行截图

状态: Accepted

源代码

```

'''
2100017810 刘思瑞
'''
from queue import deque
def calcu():
    global m,n
    a = m.pop()
    if a in ('+', '-', '*', '/'):
        b = n.pop()
        c = n.pop()
        n.append(str(eval(b+a+c)))
    else:
        n.append(a)

s = list(input().split())
m = deque(s)
n = deque()
while True:
    calcu()
    if not m:
        break
print('%.6f' %float(n[0]))

```

24591: 中序表达式转后序表达式

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

思路:

场调度算法

代码

```

1 def inp(s):
2     import re
3     s=re.split(r'([\(\)\+\-\*\\/])',s)
4     s=[item for item in s if item.strip()]
5     return s
6
7 num = int(input())

```

```

8  for j in range(num):
9      stack = []
10     output = []
11     dic = {'+':1, '-':1, '*':2, '/':2}
12     for i in inp(input()):
13         if not i in '+-*/()':
14             output.append(i)
15         else:
16             if i == '(':
17                 stack.append(i)
18             elif i == ')':
19                 while True:
20                     if stack[-1] == '(':
21                         stack.pop()
22                         break
23                 output.append(stack.pop())
24             else:
25                 if not stack or stack[-1] == '(' or dic[i] > dic[stack[-1]]:
26                     stack.append(i)
27                 else:
28                     while True:
29                         output.append(stack.pop())
30                         if not stack or stack[-1] == '(' or dic[i] >
dic[stack[-1]]:
31                             stack.append(i)
32                             break
33     while stack:
34         output.append(stack.pop())
35     print(' '.join(output))

```

代码运行截图

状态: Accepted

源代码

```
def inp(s):
    import re
    s=re.split(r'([\(\)\+\-\*\\/])',s)
    s=[item for item in s if item.strip()]
    return s

num = int(input())
for j in range(num):
    stack = []
    output = []
    dic = {'+':1,'-':1,'*':2,'/':2}
    for i in inp(input()):
        if not i in '+-*/()':
            output.append(i)
        else:
            if i == '(':
                stack.append(i)
            elif i == ')':
                while True:
                    if stack[-1] == '(':
                        stack.pop()
                        break
                output.append(stack.pop())
            else:
                if not stack or stack[-1]!='(' or dic[i] > dic[stack[-1]]:
                    stack.append(i)
                else:
                    while True:
                        output.append(stack.pop())
```

22068: 合法出栈序列

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

思路:

代码

```
1 '''
2 2100017810 刘思瑞
3 '''
4 def is_rational(s,m):
5     stack = []
6     if len(m) != len(s):
7         return False
8     while True:
9         if not m:
```



```

10         return True
11     yemp = m.pop()
12     while (not stack or stack[-1] != yemp) and s:
13         stack.append(s.pop(0))
14     if not stack or stack[-1] != yemp:
15         return False
16     stack.pop()
17
18 s = [i for i in input().strip()]
19 out = {True:'YES',False:'NO'}
20 while True:
21     try:
22         m = [i for i in input().strip()]
23         print(out[is_rational(s[::],m[::-1])])
24     except EOFError:
25         break

```

代码运行截图

状态: Accepted

源代码

```

'''
2100017810 刘思瑞
'''
def is_rational(s,m):
    stack = []
    if len(m) != len(s):
        return False
    while True:
        if not m:
            return True
        yemp = m.pop()
        while (not stack or stack[-1] != yemp) and s:
            stack.append(s.pop(0))
        if not stack or stack[-1] != yemp:
            return False
        stack.pop()

s = [i for i in input().strip()]
out = {True:'YES',False:'NO'}
while True:
    try:
        m = [i for i in input().strip()]
        print(out[is_rational(s[::],m[::-1])])
    except EOFError:
        break

```

06646: 二叉树的深度

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

思路:

递归, 第一个递归更简便, 但是写完第二个才想到

代码

```
1  '''
2  2100017810 刘思瑞
3  '''
4  from queue import deque
5  def search(tree,i):
6      if i == -2:
7          return 0
8      return max(search(tree,tree[i][0]),search(tree,tree[i][1]))+1
9  tree = []
10 n = int(input())
11 for i in range(n):
12     a,b = map(int,input().split())
13     tree.append([a-1,b-1])
14 print(search(tree,0))
```

代码运行截图

状态: Accepted

源代码

```
'''
2100017810 刘思瑞
'''

from queue import deque
def search(tree,i):
    if i == -2:
        return 0
    return max(search(tree,tree[i][0]),search(tree,tree[i][1]))+1
tree = []
n = int(input())
for i in range(n):
    a,b = map(int,input().split())
    tree.append([a-1,b-1])
print(search(tree,0))
```

代码

```
1  '''
2  2100017810 刘思瑞
3  '''
4  leng = []
```

```

5  def search(tree,lens,start):
6      global leng
7      if start == -2:
8          leng.append(lens)
9          return
10     for i in tree[start]:
11         search(tree,lens+1,i)
12     return
13
14 tree = []
15 n = int(input())
16 for i in range(n):
17     a,b = map(int,input().split())
18     tree.append([a-1,b-1])
19 search(tree,0,0)
20 print(max(leng))

```

代码运行截图

状态: **Accepted**

源代码

```

'''
2100017810 刘思瑞
'''
leng = []
def search(tree,lens,start):
    global leng
    if start == -2:
        leng.append(lens)
        return
    for i in tree[start]:
        search(tree,lens+1,i)
    return

tree = []
n = int(input())
for i in range(n):
    a,b = map(int,input().split())
    tree.append([a-1,b-1])
search(tree,0,0)
print(max(leng))

```

02299: Ultra-QuickSort

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

思路:

归并排序

代码

```

1  def mergeSort(arr):
2      import math
3      if(len(arr)<2):
4          return arr , 0
5      middle = math.floor(len(arr)/2)
6      left, inv_left = mergeSort(arr[:middle])
7      right, inv_right = mergeSort(arr[middle:])
8      merged, inv_merge = merge(left, right)
9      return merged, inv_left + inv_right + inv_merge
10 def merge(left,right):
11     result = []
12     count = 0
13     i,j = 0,0
14     while i < len(left) and j < len(right):
15         if left[i] <= right[j]:
16             result.append(left[i])
17             i += 1
18         else:
19             result.append(right[j])
20             j += 1
21             count += len(left) - i
22     result += left[i:]
23     result += right[j:]
24
25     return result, count
26
27 while True:
28     n = int(input())
29     if n == 0:
30         break
31     arr = []
32     for i in range(n):
33         arr.append(int(input()))
34     __, times = mergeSort(arr)
35     print(times)

```

代码运行截图

状态: Accepted

源代码

```
def mergeSort(arr):
    import math
    if len(arr) < 2:
        return arr , 0
    middle = math.floor(len(arr)/2)
    left, inv_left = mergeSort(arr[:middle])
    right, inv_right = mergeSort(arr[middle:])
    merged, inv_merge = merge(left, right)
    return merged, inv_left + inv_right + inv_merge
def merge(left, right):
    result = []
    count = 0
    i, j = 0, 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
            count += len(left) - i
    result += left[i:]
    result += right[j:]

    return result, count

while True:
    n = int(input())
    if n == 0:
```

2. 学习总结和收获

本周实验课太多，暂时没有做每日选做，以后补上