

# Assignment #A: 递归回溯、🌲 (3/4)

Updated 2203 GMT+8 Nov 3, 2025

2025 fall, Compiled by 魏佳亮 2400011474

说明:

## 1. 解题与记录:

对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge, Codeforces, LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora <https://typoraio.cn> 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。

2. 提交安排：提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的本人头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。

3. 延迟提交：如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。

请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

## 1. 题目

### T51.N皇后

backtracking, <https://leetcode.cn/problems/n-queens/>

思路：dfs

代码：

```
class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        l=[]
        def solve(s):
            k=len(s)
            if k==n:
                l.append(s)
                return
            for i in range(n):
                p=True
                for j in range(len(s)):
                    if i==int(s[j]) or abs(k-j)==abs(i-int(s[j])):
```

```

        p=False
        break

    if p:
        solve(s+str(i))

solve('')
r=[]
for x in l:
    t=['' for _ in range(n)]
    for i in range(n):
        t[i]='.'*int(x[i])+ 'Q'+(n-int(x[i])-1)*'.'
    r.append(t)
return r

```

代码运行截图 (至少包含有"Accepted")



题目描述 通过 × 题解 提交记录

全部提交记录

通过 9 / 9 个通过的测试用例

Ama2... 提交于 2025.11.14 22:28

面向在校学生的专享特惠  
完成认证享 7 折 Plus 会员, 享受更多...

执行用时分布  
71 ms | 击败 6.08%  
复杂度分析

Python3 智能模式

```

1 class Solution:
2     def solveNQueens(self, n: int) -> List[List[str]]:
3         l=[]
4         def solve(s):
5             k=len(s)
6             if k==n:
7                 l.append(s)
8                 return
9             for i in range(n):
10                 p=True
11                 for j in range(len(s)):

```

已存储 行 15, 列 22

## M22275: 二叉搜索树的遍历

<http://cs101.openjudge.cn/practice/22275/>

思路: recursion

代码:

```

n=int(input())
l=list(map(int,input().split()))
def solve(l):
    if not l:
        return []
    k=len(l)
    for i in range(1,k):
        if l[i]>l[0]:
            k=i
            break
    return solve(l[1:k])+solve(l[k:])+[l[0]]
print(' '.join([str(x) for x in solve(l)]))

```

代码运行截图 (至少包含有"Accepted")

#50795054提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
n=int(input())
l=list(map(int,input().split()))
def solve(l):
    if not l:
        return []
    k=len(l)
    for i in range(1,k):
        if l[i]>l[0]:
            k=i
            break
    return solve(l[1:k])+solve(l[k:])+[l[0]]
print(' '.join([str(x) for x in solve(l)]))
```

基本信息

#: 50795054  
题目: 22275  
提交人: 24n2400011474  
内存: 3748kB  
时间: 22ms  
语言: Python3  
提交时间: 2025-11-11 17:00:30

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

## M25145: 猜二叉树（按层次遍历）

<http://cs101.openjudge.cn/practice/25145/>

思路: recursion

代码:

```
n=int(input())
for _ in range(n):
    a=input()
    b=input()
    d={}
    def solve(a,b):
        if len(a)<=1:
            return
        k=a.index(b[-1])
        if k==0 or k==len(a)-1:
            d[b[-1]]=b[-2:]
        else:
            d[b[-1]]=b[k-1],b[-2:]
        solve(a[:k],b[:k])
        solve(a[k+1:],b[k:-1])
    solve(a,b)
    s=[b[-1]]
    t=''
    while s:
        r=[]
        for x in s:
            if x:
```

```

        t+=x
        if x in d:
            r+=d[x]

    s=r
    print(t)

```

代码运行截图 (至少包含有"Accepted")

#50830896提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

n=int(input())
for _ in range(n):
    a=input()
    b=input()
    d={}
    def solve(a,b):
        if len(a)<=1:
            return
        k=a.index(b[-1])
        if k==0 or k==len(a)-1:
            d[b[-1]]=b[-2]
        else:
            d[b[-1]]=b[k-1],b[-2]
        solve(a[:k],b[:k])
        solve(a[k+1:],b[k:-1])
    solve(a,b)
    s=[b[-1]]
    t=''
    while s:
        r=[]
        for x in s:
            if x:
                t+=x
                if x in d:
                    r+=d[x]
        s=r
    print(t)

```

基本信息

#: 50830896  
 题目: 25145  
 提交人: 24n2400011474  
 内存: 3568kB  
 时间: 22ms  
 语言: Python3  
 提交时间: 2025-11-13 21:47:32

## T20576: printExp (逆波兰表达式建树)

<http://cs101.openjudge.cn/practice/20576/>

思路:

代码

```

import sys

# 1. 优先级 (Precedence)
P = {'or': 1, 'and': 2, 'not': 3}

```

```

try:
    T = input().split() # Tokens
    I = [0]             # Index (using a list as a mutable pointer)

# 2. 递归下降解析器 (函数式)
# 必须按 p_atom, p_not, p_and, p_or 的顺序定义
# 或者至少要让 p_or 能找到 p_and, p_and 找到 p_not...

def p_or():
    # p_or -> p_and ('or' p_and)*
    e = p_and()
    while I[0] < len(T) and T[I[0]] == 'or':
        I[0] += 1
        e = ('or', e, p_and())
    return e

def p_and():
    # p_and -> p_not ('and' p_not)*
    e = p_not()
    while I[0] < len(T) and T[I[0]] == 'and':
        I[0] += 1
        e = ('and', e, p_not())
    return e

def p_not():
    # p_not -> 'not' p_not | p_atom
    if I[0] < len(T) and T[I[0]] == 'not':
        I[0] += 1
        return ('not', p_not())
    return p_atom()

def p_atom():
    # p_atom -> 'True' | 'False' | '(' p_or ')'
    tok = T[I[0]]
    I[0] += 1
    if tok == '(':
        e = p_or()
        I[0] += 1 # 消耗 ')'
        return e
    return tok # 'True' or 'False'

# 3. 序列化函数 (toString)
def to_str(n, p=0): # n = node, p = parent_precedence
    if isinstance(n, str):
        return n # 'True' or 'False'

    op = n[0]
    pr = P[op] # my precedence

    if op == 'not':
        s = f"not {to_str(n[1], pr)}"
    else:

```

```

        # 处理左结合性
        rp = pr + 0.1 if op in ('or', 'and') else pr
        s = f"{to_str(n[1], pr)} {op} {to_str(n[2], rp)}"

    # 核心: 当我的优先级低于父级时, 才加括号
    return f"({s})" if pr < p else s

# 4. 执行
print(to_str(p_or()))

except:
    # 捕获 EOFError 和所有其他解析错误
    pass

```

代码运行截图 (至少包含有"Accepted")

#50842940提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

import sys

# 1. 优先级 (Precedence)
P = {'or': 1, 'and': 2, 'not': 3}

try:
    T = input().split() # Tokens

```

基本信息

#: 50842940  
 题目: 20576  
 提交人: 24n2400011474  
 内存: 3680kB  
 时间: 20ms  
 语言: Python3  
 提交时间: 2025-11-14 22:32:07

## T04080:Huffman编码树

greedy, <http://cs101.openjudge.cn/practice/04080/>

思路: greedy吧, 真没想到

代码

```

import heapq
n=int(input())
l=list(map(int,input().split()))
heapq.heapify(l)
r=0
while len(l)>1:
    a=heapq.heappop(l)
    b=heapq.heappop(l)
    r+=a+b
    heapq.heappush(l,a+b)
print(r)

```

代码运行截图 (至少包含有"Accepted")

#50831303提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```
import heapq
n=int(input())
l=list(map(int,input().split()))
heapq.heapify(l)
r=0
while len(l)>1:
    a=heapq.heappop(l)
    b=heapq.heappop(l)
    r+=a+b
    heapq.heappush(l,a+b)
print(r)
```

基本信息

#: 50831303  
题目: 04080  
提交人: 24n2400011474  
内存: 3624kB  
时间: 22ms  
语言: Python3  
提交时间: 2025-11-13 22:23:52

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

## M04078: 实现堆结构

<http://cs101.openjudge.cn/practice/04078/>

要求手搓堆实现。

思路：直接heapq实现

代码：

```
import heapq
n=int(input())
l=[]
heapq.heapify(l)
for _ in range(n):
    a=list(map(int,input().split()))
    if a[0]==1:
        heapq.heappush(l,a[1])
    else:
        print(heapq.heappop(l))
```

代码运行截图 (至少包含有"Accepted")

状态: **Accepted**

源代码

```
import heapq
n=int(input())
l=[]
heapq.heapify(l)
for _ in range(n):
    a=list(map(int,input().split()))
    if a[0]==1:
        heapq.heappush(l,a[1])
    else:
        print(heapq.heappop(l))
```

基本信息

#: 50830967

题目: 04078

提交人: 24n2400011474

内存: 4020kB

时间: 360ms

语言: Python3

提交时间: 2025-11-13 21:54:50

## 2. 学习总结和个人收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025fall每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

每日选做在跟着，波兰表达式摆烂了，估计考试出现这种题我也会放弃