

assignment8

Assignment #8: 递归

Updated 13:15 GMT+8 Oct 21, 2025

2025 fall, Complied by 顾桂榕 基础医学院



顾桂榕 医学预科办

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

M04147汉诺塔问题(Tower of Hanoi)

dfs, <http://cs101.openjudge.cn/pctbook/Mo4147>

思路：

代码

```
def move_one(num:int,departure:str,terminal:str):
    print(f'{num}:{departure}→{terminal}')
def move(num:int,departure:str,stack:str,terminal:str):
    if num==1:
        move_one(num,departure,terminal)
    else:
        move(num-1,departure,terminal,stack)
        move_one(num,departure,terminal)
        move(num-1,stack,departure,terminal)
n,a,b,c = input().split()
move(int(n),a,b,c)
```

代码运行截图 (至少包含有"Accepted")

#50559362提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def move_one(num:int,departure:str,terminal:str):
    print(f'{num}:{departure}→{terminal}')
def move(num:int,departure:str,stack:str,terminal:str):
    if num==1:
        move_one(num,departure,terminal)
    else:
        move(num-1,departure,terminal,stack)
        move_one(num,departure,terminal)
        move(num-1,stack,departure,terminal)
n,a,b,c = input().split()
move(int(n),a,b,c)
```

基本信息

#: 50559362
题目: M04147
提交人: R.
内存: 3596kB
时间: 21ms
语言: Python3
提交时间: 2025-10-25 21:25:05

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

M05585: 晶矿的个数

matrices, dfs similar, <http://cs101.openjudge.cn/pctbook/Mo5585>

思路：

代码

代码运行截图 (至少包含有"Accepted")

M02786: Pell数列

dfs, dp, <http://cs101.openjudge.cn/pctbook/Mo2786/>

思路:

原来思路是检查左边和上边是否连通

if gem[x][y-1]!='r' and gem[x-1][y]!='r':

r+=1

在大多数情况下是对的

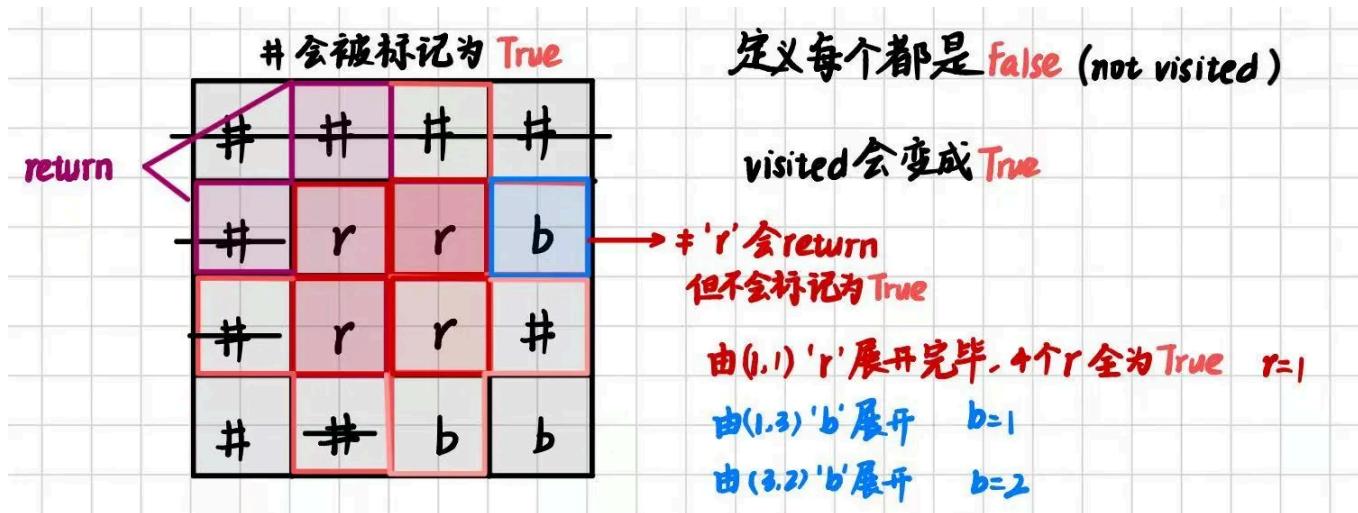
但是有反例:

r # r

r r r

r r r

遂学了深搜dfs



代码

```
k = int(input())
for _ in range(k):
    n = int(input())
    gem = []
    r = b = 0
    for i in range(n):
```

```
gem.append(list(input().strip()))
visited = [[False]*n for _ in range(n)]

def dfs(x, y, mineral_type):
    if x < 0 or x ≥ n or y < 0 or y ≥ n:
        return
    if visited[x][y] or gem[x][y]≠mineral_type:
        return
    visited[x][y]=True
    directions = [(0,1),(0,-1),(1,0),(-1,0)]
    for dx,dy in directions:
        dfs(x+dx,y+dy,mineral_type)

for i in range(n):
    for j in range(n):
        if not visited[i][j]:
            if gem[i][j]=='r':
                r+=1
                dfs(i,j,'r')
            elif gem[i][j]=='b':
                b+=1
                dfs(i,j,'b')
print(r,b)
```

代码运行截图 (至少包含有"Accepted")

#50622376提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
k = int(input())
for _ in range(k):
    n = int(input())
    gem = []
    r = b = 0
    for i in range(n):
        gem.append(list(input().strip()))
    visited = [[False]*n for _ in range(n)]

    def dfs(x, y, mineral_type):
        if x < 0 or x >= n or y < 0 or y >= n:
            return
        if visited[x][y] or gem[x][y] != mineral_type:
            return
        visited[x][y] = True
        directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
        for dx, dy in directions:
            dfs(x+dx, y+dy, mineral_type)

    for i in range(n):
        for j in range(n):
            if not visited[i][j]:
                if gem[i][j] == 'r':
                    r += 1
                    dfs(i, j, 'r')
                elif gem[i][j] == 'b':
                    b += 1
                    dfs(i, j, 'b')
print(r, b)
```

基本信息

#: 50622376
题目: M05585
提交人: R.
内存: 3644kB
时间: 23ms
语言: Python3
提交时间: 2025-10-29 18:54:51

Windows 更新

重启以安装最新的 Windows
你的设备有新的更新可用。请稍等，我们将在你的使用时段自动重启。
或者，选择一个适合你的时间。



立即重启



选择一个时间

M46.全排列

backtracking, <https://leetcode.cn/problems/permutations/>

思路:

代码

```
line = input().strip()
num_list = []
for i in range(len(line)):
    if line[i].isdigit():
        num_list.append(int(line[i]))
def permute(num_list):
    def backtrack(used, temp):
        if len(temp) == len(num_list):
            res.append(temp[:])
            return
        for i in range(len(num_list)):
            if not used[i]:
                used[i] = True
                temp.append(num_list[i])
                backtrack(used, temp)
                temp.pop()
                used[i] = False
res = []
permute(num_list)
print(res)
```

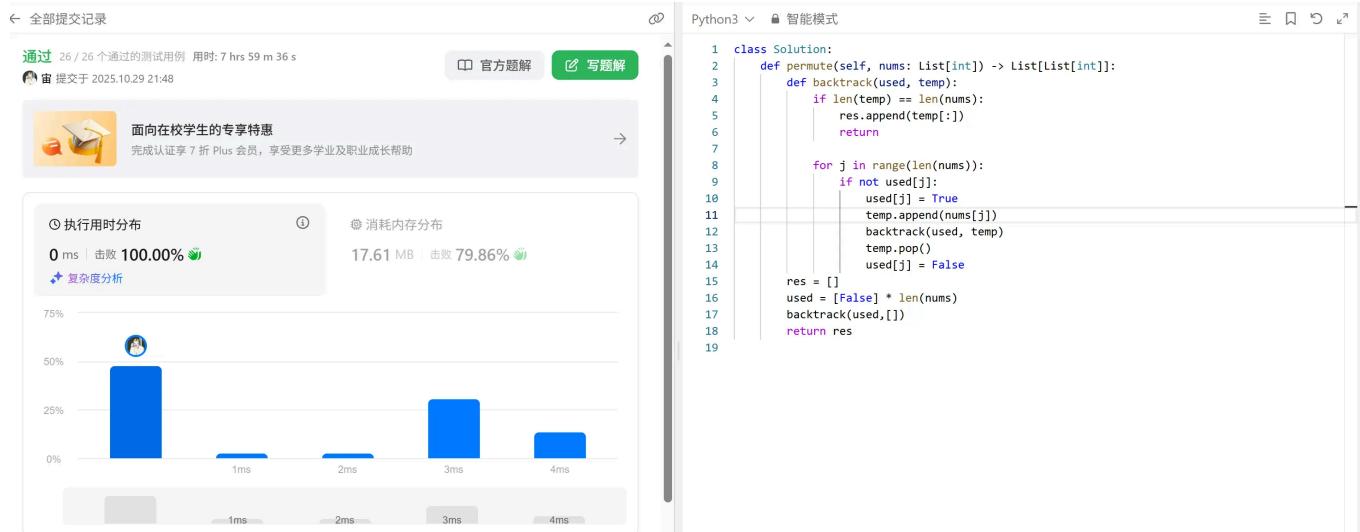
```

        for j in range(len(num_list)):
            if not used[j]:
                used[j] = True
                temp.append(num_list[j])
                backtrack(used, temp)
                temp.pop()
                used[j] = False

        res = []
        used = [False] * len(num_list)
        backtrack(used, [])
        return res
print(permute(num_list))

```

代码运行截图 (至少包含有"Accepted")



T02754: 八皇后

dfs and similar, <http://cs101.openjudge.cn/pctbook/T02754>

思路：

原来想仿照全排列做一个 8×8 的matrix作为used

并且使用显性撤销pop的操作

其实可以用数组进行隐性覆盖原来的数值

这个时候并不那么直观

因为并没有所谓的x

进行is_safe判断的时候也只需return True or False即可

代码

```
from functools import lru_cache
@lru_cache(maxsize=None)
def eight_queens(index):
    def is_safe(board, row, col):
        for i in range(row):
            if col==board[i]:
                return False
            if abs(col-board[i])==abs(row-i):
                return False
        return True
    def backtrack(board, row):
        if row==8:
            res.append([x+1 for x in board])
            return
        for col in range(8):
            if is_safe(board, row, col):
                board[row]=col
                backtrack(board, row+1)
    res = []
    backtrack([0]*8, 0)
    return res[index]
n = int(input())
for _ in range(n):
    b = int(input())
    print(*eight_queens(b-1), sep='')
```

代码运行截图 (至少包含有"Accepted")

#50631498提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from functools import lru_cache
@lru_cache(maxsize=None)
def eight_queens(index):
    def is_safe(board, row, col):
        for i in range(row):
            if col==board[i]:
                return False
            if abs(col-board[i])==abs(row-i):
                return False
        return True
    def backtrack(board, row):
        if row==8:
            res.append([x+1 for x in board])
            return
        for col in range(8):
            if is_safe(board, row, col):
                board[row]=col
                backtrack(board, row+1)
    res = []
    backtrack([0]*8, 0)
    return res[index]
n = int(input())
for _ in range(n):
    b = int(input())
    print(*eight_queens(b-1), sep='')
```

基本信息

#: 50631498
题目: T02754
提交人: R.
内存: 3800kB
时间: 969ms
语言: Python3
提交时间: 2025-10-30 16:47:45

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

T01958 Strange Towers of Hanoi

<http://cs101.openjudge.cn/practice/01958/> 

思路:

代码

```
def normal_hanoi_tower(n):
    if n==1:
        return 1
    if n==2:
        return 3
    return normal_hanoi_tower(n-1)*2+1
def strange_hanoi_tower(n):
    ans_list = []
    if n==1:
        return 1
    if n==2:
        return 3
```

```

if n==3:
    return 5
for k in range(1,n):
    ans = normal_hanoi_tower(n-k)+strange_hanoi_tower(k)*2
    ans_list.append(ans)
min_ans = min(ans_list)
return min_ans
for i in range(1,13):
    print(strange_hanoi_tower(i))

```

代码运行截图 (至少包含有"Accepted")

#50632690提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```

def normal_hanoi_tower(n):
    if n==1:
        return 1
    if n==2:
        return 3
    return normal_hanoi_tower(n-1)*2+1
def strange_hanoi_tower(n):
    ans_list = []
    if n==1:
        return 1
    if n==2:
        return 3
    if n==3:
        return 5
    for k in range(1,n):
        ans = normal_hanoi_tower(n-k)+strange_hanoi_tower(k)*2
        ans_list.append(ans)
    min_ans = min(ans_list)
    return min_ans
for i in range(1,13):
    print(strange_hanoi_tower(i))

```

基本信息

#: 50632690
 题目: 01958
 提交人: R.
 内存: 3612kB
 时间: 26ms
 语言: Python3
 提交时间: 2025-10-30 17:57:56

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

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“计概2025fall每日选做”、CF、LeetCode、洛谷等网站题目。

回溯算法的核心思想是：

1. 向前探索（递归向下） - 做出选择，进入下一层决策
2. 向后回退（回溯） - 当发现当前路径不可能得到解时，撤销选择，返回上一层
3. backtrack搭配mark(used/visited/is_safe)作为judge食用

为什么叫"回溯"而不是"递归探索"

关键区别在于"撤销选择"的环节：

普通递归：

```
function(n) → function(n-1) → function(n-2) → ...
```

只是单向地向更深层次调用，没有"回头"的概念。

回溯算法：

```
选择A → 探索 → 发现不行 → **回退** → 选择B → 探索 → ...
```

↑ _____ 回溯 _____ ↓

斐波那契数列(recursion)

『研算法全程训练营



RoseRong

难度

个人主页

我的收藏

题目统计

深色主题

意见反馈

退出登录

代码书写

```
1 def fibonacci_sequence(n):  
2     if n==1 or n==2:  
3         return 1  
4     return fibonacci_sequence  
5 n = int(input())  
6 print(fibonacci_sequence(n))
```

测试输入

提交结果

历史提交

完美通过

查看题解

100% 数据通过测试 [详情](#)

运行时长: 0 ms

斐波那契数列(矩阵快速幂)

码书写

```
2 def matrix_multiply(A,B):
3     c11 = (A[0][0]*B[0][0]+A[0][1]*B[1][0])%
4     c12 = (A[0][0]*B[0][1]+A[0][1]*B[1][1])%
5     c21 = (A[1][0]*B[0][0]+A[1][1]*B[1][0])%
6     c22 = (A[1][0]*B[0][1]+A[1][1]*B[1][1])%
7     return [[c11,c12],[c21,c22]]
8 def fibonacci_matrix(n):
9     if n==0:
10        return 0
11    elif n==1:
12        return 1
13    else:
14        matrix = [[1,1],
15                  [1,0]]
16        return powered_matrix(n-1,matrix)
17 def powered_matrix(exponent,matrix):
18     base = matrix
19     result = [[1,0],
20               [0,1]]
21     while exponent>0:
```

|试输入

提交结果

历史提交

[个人主页](#)[我的收藏](#)[题目统计](#)[深色主题](#)[意见反馈](#)[退出登录](#)

完美通过

[查看题解](#)

100% 数据通过测试

[详情](#)

fibonacci-matrix

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix}$$

$$1024 = 2^{10}$$

normal 2^{10} 次计算 $O(n)$

powered-matrix 10 次计算 $O(\log n)$

$$1000000000_2$$

$$10^8 \rightarrow \log_2 10^8 \approx 60$$

$$F_{10} = 0 \quad F_{11} = 1 \quad F_{12} = 1$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix} = \begin{pmatrix} F_3 & F_2 \\ F_2 & F_1 \end{pmatrix}$$

$$\begin{pmatrix} F_3 & F_2 \\ F_2 & F_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^2$$

$$\begin{pmatrix} F_n & F_{n+1} \\ F_{n+1} & F_{n+2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$$

$$n=10 \text{ 例: } \begin{pmatrix} F_{10} & F_9 \\ F_9 & F_8 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^9 \text{ 算 } 9 \text{ 次不快速}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^9 =$$

result	base	base
$10 = 1010_2$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$
exponent = 9		

$$\textcircled{1} \quad \text{result} = (\text{result}, \text{base}) \quad \text{base} = (\text{base}, \text{base})^{\wedge} 2$$

$$\text{exponent} = 4$$

$$\textcircled{2} \quad \text{base} = (\text{base}, \text{base})^{\wedge} 4$$

$$\text{exponent} = 2$$

$$\textcircled{3} \quad \text{base} = (\text{base}, \text{base})^{\wedge} 8$$

$$\text{exponent} = 1$$

$$\textcircled{4} \quad \text{result} = (\text{result}, \text{base})^{\wedge} 9$$

$$\text{exponent} = 0, \text{ break}$$

$$\text{exponent}$$

$$15 = 1111_2 \quad 0+2+4+8=14$$

$$25 = 11001_2 \quad 0+8+16=24$$

螺旋序列

状态: Accepted

源代码

```
n = int(input())
res = [[0]*n for _ in range(n)]
bd_down=bd_right=n;bd_up=bd_left=0
x = y = temp = 0
while True:
    if temp == n ** 2:
        break
    if x == bd_up and y == bd_left:
        for y in range(bd_left, bd_right):
            temp += 1
            res[x][y] = temp
        y = bd_right
        bd_up += 1
        x = bd_up
    elif x == bd_up and y == bd_right:
        for x in range(bd_up, bd_down):
            temp += 1
            res[x][y - 1] = temp
        x = bd_down
        bd_right -= 1
        y = bd_right
    elif x == bd_down and y == bd_right:
        for y in range(bd_right-1, bd_left - 1, -1):
            temp += 1
            res[x - 1][y] = temp
        y = bd_left
        bd_down -= 1
        x = bd_down
    elif x == bd_down and y == bd_left:
        for x in range(bd_down-1, bd_up - 1, -1):
            temp += 1
            res[x][y] = temp
        x = bd_up
        bd_left += 1
        y = bd_left
    for _ in range(n):
        print(*res[_])
```

基本信息

#: 50565087
题目: M18106
提交人: R.
内存: 3672kB
时间: 21ms
语言: Python3
提交时间: 2025-10-26 12:27:31