

Assignment #8: 递归

Updated 1315 GMT+8 Oct 21, 2025

2025 fall, Complied by 郭旭杰、化学与分子工程学院

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/M04147>

耗时：30min解决（上课讲了很大一部分了）

思路：基础的递归，经典问题。

代码

```
def mov_hanoi(n0, a0, b0):
    return f'{n0}:{a0}>{b0}'

def solv_hanoi(n0, a0, b0, c0):
    if n0 == 1:
        return [mov_hanoi(n0, a0, c0)]
    else:
        jit_sp_solv_hanoi = []
        for i in solv_hanoi(n0 - 1, a0, c0, b0):
            jit_sp_solv_hanoi.append(str(i))
```

```

jit_sp_solv_hanoi.append(mov_hanoi(n0, a0, c0))
for i in solv_hanoi(n0 - 1, b0, a0, c0):
    jit_sp_solv_hanoi.append(str(i))
return jit_sp_solv_hanoi

n, a, b, c = list(map(str, input().split()))
n_int = int(n)
js = solv_hanoi(n_int, a, b, c)
for j in js:
    print(j)

```

代码运行截图 (至少包含有"Accepted")

OpenJudge
题目ID, 标题, 描述
25n2500011906 信箱 账号

CS101 / 计算思维算法实践

题目
排名
状态
提问

M04147:汉诺塔问题(Tower of Hanoi)
查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65535kB
全局题号 8200
添加于 2025-03-13
提交次数 234
尝试人数 164
通过人数 162

描述

一、汉诺塔问题

有三根杆子A, B, C。A杆上有N个(N>1)穿孔圆盘，盘的尺寸由下到上依次变小。要求按下列规则将所有圆盘移至C杆：每次只能移动一个圆盘；大盘不能叠在小盘上面。提示：可将圆盘临时置于B杆，也可将从A杆移出的圆盘重新移回A杆，但都必须遵循上述两条规则。

问：如何移？最少要移动多少次？

汉诺塔示意图如下：

#	结果	时间
6	Accepted	2025-10-30
5	Accepted	2025-10-30
4	Wrong Answer	2025-10-30
3	Wrong Answer	2025-10-30
2	Compile Error	2025-10-30
1	Wrong Answer	2025-10-30



#50631995提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

def mov_hanoi(n0, a0, b0):
    return f'{n0} : {a0} -> {b0}'


def solv_hanoi(n0, a0, b0, c0):
    if n0 == 1:
        return [mov_hanoi(n0, a0, c0)]
    else:
        jit_sp_solv_hanoi = []
        for i in solv_hanoi(n0 - 1, a0, c0):
            jit_sp_solv_hanoi.append(str(i))
        jit_sp_solv_hanoi.append(mov_hanoi(n0, a0, c0))
        for i in solv_hanoi(n0 - 1, b0, a0, c0):
            jit_sp_solv_hanoi.append(str(i))
        return jit_sp_solv_hanoi


n, a, b, c = list(map(str, input().split()))
n_int = int(n)
js = solv_hanoi(n_int, a, b, c)
for j in js:
    print(j)

```

基本信息

#: 50631995
 题目: M04147
 提交人: 25n2500011906
 内存: 3580kB
 时间: 22ms
 语言: Python3
 提交时间: 2025-10-30 17:14:33

©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/M05585>

耗时: 整理思路耗时数日, 改代码做了我大半个晚上。

思路: 用到了dfs, 学起来有难度。

最终求助腾讯元宝, 提供了一个双矩阵映射的思路。我对其加以完善。

我自己的思路是使用三维数组, 但是最终没有成功。

代码

```

def kpj(listy):
    return ' '.join(listy)

Lne = int(input())
for _ in range(Lne):
    n = int(input())
    visited = [[False] * n for _ in range(n)]
    matrix = []

```

```
for i in range(n):
    row = []
    row_root = input()
    for rr0 in row_root:
        row.append(rr0)
    matrix.append(row)
red = 0
black = 0

def dfs_pm(i0, j0, tara):
    if i0 < 0 or i0 >= n or j0 < 0 or j0 >= n:
        return
    if visited[i0][j0]:
        return
    if matrix[i0][j0] != tara:
        return
    visited[i0][j0] = True
    dfs_pm(i0 + 1, j0, tara)
    dfs_pm(i0 - 1, j0, tara)
    dfs_pm(i0, j0 + 1, tara)
    dfs_pm(i0, j0 - 1, tara)

for i in range(n):
    for j in range(n):
        if matrix[i][j] == 'r' and not visited[i][j]:
            red += 1
            dfs_pm(i, j, 'r')
        elif matrix[i][j] == 'b' and not visited[i][j]:
            black += 1
            dfs_pm(i, j, 'b')

print(red, black)
```

代码运行截图 (至少包含有"Accepted")



CS101 / 计算思维算法实践

题目 排名 状态 提问

M05585:晶矿的个数

总时间限制: 1000ms 内存限制: 65536kB

描述

在某个区域发现了一些晶矿，已经探明这些晶矿总共有分为两类，为红晶矿和黑晶矿。现在要统计该区域内红晶矿和黑晶矿的个数。假设可以用二维地图 $m[][]$ 来描述该区域，若 $m[i][j]$ 为#表示该地点是非晶矿地点，若 $m[i][j]$ 为r表示该地点是红晶矿地点，若 $m[i][j]$ 为b表示该地点是黑晶矿地点。一个晶矿是由相同类型的并且上下左右相通的晶矿点组成。现在给你该区域的地图，求红晶矿和黑晶矿的个数。

输入

第一行为k，表示有k组测试输入。

每组第一行为n，表示该区域由 $n \times n$ 个地点组成， $3 \leq n \leq 30$

接下来n行，每行n个字符，表示该地点的类型。

输出

对每组测试数据输出一行，每行两个数字分别是红晶矿和黑晶矿的个数，一个空格隔开。

查看

提交

统计

提问

全局题号 5585

添加于 2025-03-13

提交次数 225

尝试人数 146

通过人数 137

你的提交记录

#	结果	时间
4	Accepted	2025-11-03
3	Wrong Answer	2025-10-31
2	Wrong Answer	2025-10-31
1	Wrong Answer	2025-10-31

状态: Accepted

源代码

```

def kpj(listy):
    return ' '.join(listy)

Lne = int(input())
for _ in range(Lne):
    n = int(input())
    visited = [[False] * n for _ in range(n)]
    matrix = []
    for i in range(n):
        row = []
        row_root = input()
        for rr0 in row_root:
            row.append(rr0)
        matrix.append(row)
    red = 0
    black = 0

    def dfs_pm(i0, j0, tara):
        if i0 < 0 or i0 >= n or j0 < 0 or j0 >= n:
            return
        if visited[i0][j0]:
            return
        if matrix[i0][j0] != tara:
            return
        visited[i0][j0] = True
        dfs_pm(i0 + 1, j0, tara)
        dfs_pm(i0 - 1, j0, tara)
        dfs_pm(i0, j0 + 1, tara)
        dfs_pm(i0, j0 - 1, tara)

        for i in range(n):
            for j in range(n):
                if matrix[i][j] == 'r' and not visited[i][j]:
                    red += 1
                    dfs_pm(i, j, 'r')
                elif matrix[i][j] == 'b' and not visited[i][j]:
                    black += 1
                    dfs_pm(i, j, 'b')

    print(red, black)

```

基本信息

#: 50678843
 题目: M05585
 提交人: 25n2500011906
 内存: 43308kB
 时间: 133ms
 语言: PyPy3
 提交时间: 2025-11-03 00:03:20

M02786: Pell数列

dfs, dp, <http://cs101.openjudge.cn/pctbook/M02786/>

耗时: 几天内不断优化, 终于完成。

思路: 注意学会利用“模”这个看似无用的信息以简化运算, 防止超时(TLE)和爆栈(RE)

不要尝试将递推暴力转化成通项, 带根号2, 计算机可能吃不消, PyCharm上面可以运行样例, 提交时会报错。

真正写出来能AC的代码其实并不长。

代码

```
pell = [0, 1]
for _ in range(1000000):
    pell.append((pell[-2] + 2 * pell[-1]) % 32767)
```

```
Lne = int(input())
for _ in range(Lne):
    print(pell[int(input())])
```

代码运行截图 (至少包含有"Accepted")

OpenJudge 题目ID, 标题, 描述 25n2500011906 信箱 账号

CS101 / 计算思维算法实践

题目 排名 状态 提问

M02786:Pell数列

查看 提交 统计 提问

总时间限制: 3000ms 内存限制: 65536kB

全局题号 1788
添加于 2025-03-13
提交次数 671
尝试人数 205
通过人数 199

描述

Pell数列 a_1, a_2, a_3, \dots 的定义是这样的, $a_1 = 1, a_2 = 2, \dots, a_n = 2 * a_{n-1} + a_{n-2}$ ($n > 2$)。给出一个正整数k, 要求Pell数列的第k项模上32767是多少。

输入

第1行是测试数据的组数n, 后面跟着n行输入。每组测试数据占1行, 包括一个正整数k ($1 \leq k < 1000000$)。

输出

n行, 每行输出对应一个输入。输出应是一个非负整数。

样例输入

```
2
1
8
```

样例输出

```
1
408
```

查看 提交 统计 提问

你的提交记录

#	结果	时间
16	Accepted	2025-11-02
15	Runtime Error	2025-11-02
14	Runtime Error	2025-11-02
13	Accepted	2025-11-02
12	Runtime Error	2025-10-31
11	Runtime Error	2025-10-31
10	Runtime Error	2025-10-31
9	Runtime Error	2025-10-31
8	Runtime Error	2025-10-31
7	Runtime Error	2025-10-31
6	Runtime Error	2025-10-31
5	Time Limit Exceeded	2025-10-28
4	Time Limit Exceeded	2025-10-28
3	Time Limit Exceeded	2025-09-18
2	Wrong Answer	2025-09-18
1	Compile Error	2025-09-18



#50676950提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
pell = [0, 1]
for _ in range(1000000):
    pell.append((pell[-2] + 2 * pell[-1]) % 32767)

Lne = int(input())
for _ in range(Lne):
    print(pell[int(input())])
```

基本信息

#: 50676950
 题目: M02786
 提交人: 25n2500011906
 内存: 106496kB
 时间: 171ms
 语言: PyPy3
 提交时间: 2025-11-02 21:14:02

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

M46.全排列

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

思路:

递归思路，不断由少一个元素的结果生成最终结果。

解题过程

先定义可调用的permute0，再敲定base(len(listy) == 1)，然后递归即可。

复杂度

- 时间复杂度: O($n \cdot n!$)
- 空间复杂度: O($n \cdot n!$)

代码

```
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        def permute0(listy):
            if len(listy) == 1:
                return([[listy[0]]])
            else:
                al_listy = []
                for j_in_listy in listy:
                    n_listy = [j_in_listy]
                    listy0 = listy[:]
                    listy0.remove(j_in_listy)
                    for k_in_listyf in permute0(listy0):
                        for k_in_listy in k_in_listyf:
                            n_listy.append(k_in_listy)
                al_listy.append(n_listy)
        return al_listy
```

```
n_listy = [j_in_listy]
return(al_listy)
```

```
return(permute0(nums))
```

代码运行截图 (至少包含有"Accepted")

The screenshot shows a Python code editor with the following code:

```
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        def permute0(listy):
            if len(listy) == 1:
                return([[listy[0]]])
            else:
                al_listy = []
                for j_in_listy in listy:
                    n_listy = [j_in_listy]
                    listy0 = listy[:]
                    listy0.remove(j_in_listy)
                    for k_in_listyf in permute0(listy0):
                        for k_in_listy in k_in_listyf:
                            n_listy.append(k_in_listy)
                            al_listy.append(n_listy)
                            n_listy = [j_in_listy]
                return(al_listy)
        return(permute0(nums))
```

The code defines a class `Solution` with a method `permute` that returns all permutations of a list `nums`. It uses a helper function `permute0` to generate permutations of a list `listy`. The logic involves removing an element from the list and then generating all permutations of the remaining elements, appending the removed element back to each permutation.

Below the code editor, there is a status bar showing "已存储" (Saved) and "行 1, 列 1" (Line 1, Column 1). The test results section shows "通过" (Passed) with a execution time of 0 ms, and checkboxes for Case 1, Case 2, and Case 3, all of which are checked. The input field contains `nums = [1,2,3]`, and the output field displays `[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`.

通过 26 / 26 个通过的测试用例

 LittleBeetroot 提交于 2025.11.03 00:34

 官方题解

 写题解



面向在校学生的专享特惠

完成认证享 7 折 Plus 会员，享受更多学业及职业成长帮助



① 执行用时分布

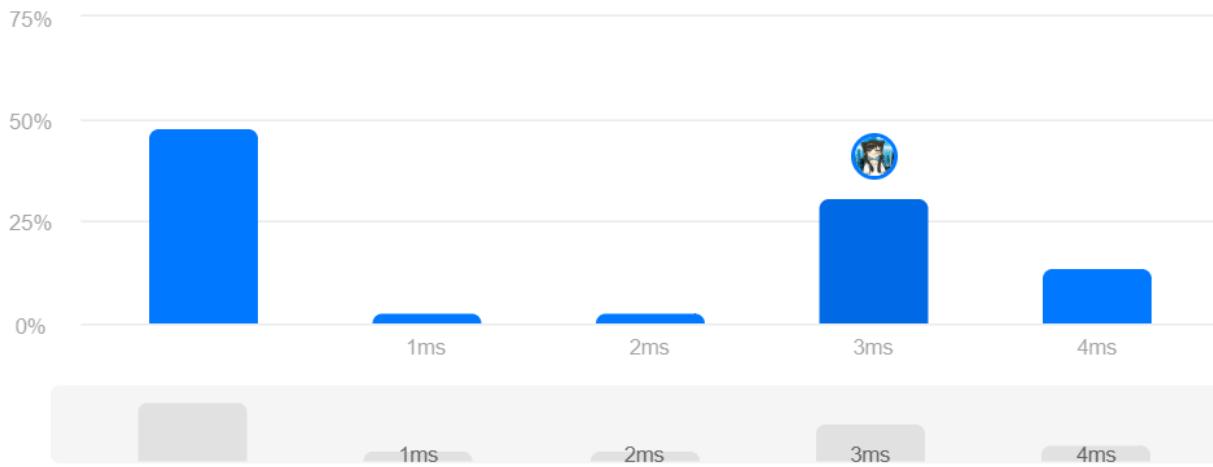
3 ms | 击败 47.15%

 复杂度分析



② 消耗内存分布

17.61 MB | 击败 80.94% 



① 执行用时分布

3 ms | 击败 47.15%

② 消耗内存分布

17.61 MB | 击败 80.94% 

 复杂度分析



+ 你最近一次提交运行时间超过了 47% 的用户

 发布题解



LittleBeetroot

46.全排列题解（适合Python新手）——LittleBeetroot 我创建的

Problem: 46. 全排列 思路 递归思路，不断由少一个元素的结果生成最终结果。解题过程 先定义可调用...

Python3

 0  2  0

T02754: 八皇后

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

耗时：做了整整一个下午。

思路：M46.全排列里面的函数可以搬过来用了。为简化运算，

最后应该逐个筛选正确选项(valid == True\blacklist == 0)

而不应该逐个排除错误选项(valid == False\blacklist != 0)。

blacklist==0表示合法这个是我个人的习惯，从两个月之前就这么写，当时还不能掌握bool型数据的用法，就用blacklist来替代。后来发现这种写法虽略耗时但却有可以表示更加复杂状态的优点，就一直这样用习惯了。

代码

```
def koj(listy):
    return ''.join(listy)

def _2sqr(x0, y0, x1, y1):
    if abs(x0 - x1) == abs(y0 - y1):
        return True
    else:
        return False

def permute0(listy):
    if len(listy) == 1:
        return [[listy[0]]]
    else:
        al_listy = []
        for j_in_listy in listy:
            n_listy = [j_in_listy]
            listy0 = listy[:]
            listy0.remove(j_in_listy)
            for k_in_listyf in permute0(listy0):
                for k_in_listy in k_in_listyf:
                    n_listy.append(k_in_listy)
```

```
al_listy.append(n_listy)
n_listy = [j_in_listy]
return al_listy

a1 = permute0([1, 2, 3, 4, 5, 6, 7, 8])
b1 = []
for a in a1:
    blacklist = 0
    for i in range(0, 7):
        if blacklist == 0:
            for j in range(i + 1, 8):
                if _2sqr(i + 1, int(a[i]), j + 1, int(a[j])):
                    blacklist = 1
                    break

    if blacklist == 0:
        b1.append(a)

res = []
for a in b1:
    aa = []
    for b in a:
        aa.append(str(b))
    res.append(aa)
res.sort()

Lne = int(input())
for _ in range(Lne):
    n = int(input())
    print(k0j(res[n - 1]))
```

代码运行截图 (至少包含有"Accepted")



T02754:八皇后

查看

提交

统计

提问

总时间限制: 1000ms 内存限制: 65536kB

描述

会下国际象棋的人都很清楚：皇后可以在横、竖、斜线上不限步数地吃掉其他棋子。如何将8个皇后放在棋盘上（有8 * 8个方格），使它们谁也不能被吃掉！这就是著名的八皇后问题。

对于某个满足要求的8皇后的摆放方法，定义一个皇后串 a 与之对应，即 $a=b_1b_2\dots b_8$ ，其中 b_i 为相应摆法中第 i 行皇后所处的列数。已经知道8皇后问题一共有92组解（即92个不同的皇后串）。

给出一个数 b ，要求输出第 b 个串。串的比较是这样的：皇后串 x 置于皇后串 y 之前，当且仅当将 x 视为整数时比 y 小。

输入

第1行是测试数据的组数 n ，后面跟着 n 行输入。每组测试数据占1行，包括一个正整数 b ($1 \leq b \leq 92$)

输出

输出有 n 行，每行输出对应一个输入。输出应是一个正整数，是对应于 b 的皇后串。

样例输入

```
2
1
92
```

样例输出

```
15863724
84136275
```

查看 提交 统计 提问



#50673071提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

def k0j(listy):
    return ''.join(listy)

def _2sqr(x0, y0, x1, y1):
    if abs(x0 - x1) == abs(y0 - y1):
        return True
    else:
        return False

def permute0(listy):
    if len(listy) == 1:
        return [[listy[0]]]
    else:
        al_listy = []
        for j_in_listy in listy:
            n_listy = [j_in_listy]
            listy0 = listy[:]
            listy0.remove(j_in_listy)
            for k_in_listyf in permute0(listy0):
                for k_in_listy in k_in_listyf:
                    n_listy.append(k_in_listy)
            al_listy.append(n_listy)
        n_listy = [j_in_listy]
    return al_listy

al = permute0([1, 2, 3, 4, 5, 6, 7, 8])
bl = []
for a in al:
    blacklist = 0
    for i in range(0, 7):
        if blacklist == 0:
            for j in range(i + 1, 8):
                if _2sqr(i + 1, int(a[i]), j + 1, int(a[j])):
```

基本信息

#: 50673071
 题目: T02754
 提交人: 25n2500011906
 内存: 10580kB
 时间: 329ms
 语言: Python3
 提交时间: 2025-11-02 18:29:20

T01958 Strange Towers of Hanoi

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

耗时: 20min速通。

思路: 其实有了M04147汉诺塔问题(Tower of Hanoi)的基础, 再仔细读题, 很轻松就能AC;

另一题29750:困难河内塔(只阅读了题目, 也是和汉诺塔相关问题, 并没有来得及做, 截至11月3日00:00无一人通过)是真的难, 以后有空再做。

代码

```

def hanoi_3(n0):
    return 2 ** n0 - 1
```

```

def hanoi_4(n0):
    if n0 == 1:
        return 1
    elif n0 == 2:
        return 3
    else:
        listy_hanoi_4 = []
        for k0 in range(1, n0):
            jit_hanoi_4 = 2 * hanoi_4(n0 - k0) + hanoi_3(k0)
            listy_hanoi_4.append(jit_hanoi_4)
        return min(listy_hanoi_4)

for i in range(1, 13):
    print(hanoi_4(i))

```

代码运行截图 (至少包含有"Accepted")

#	结果	时间
1	Accepted	2025-11-02



Figure 4: The standard (three) Towers of Hanoi.

The teacher points to the blackboard (Fig. 4) and says: "So here is the problem:

There are three towers: A, B and C.

There are n disks. The number n is constant while working the puzzle.

All disks are different in size.

The disks are initially stacked on tower A increasing in size from the top to the bottom.

The goal of the puzzle is to transfer all of the disks from tower A to tower C.

One disk at a time can be moved from the top of a tower either to an empty tower or to a tower with a larger disk on the top.

So your task is to write a program that calculates the smallest number of disk moves necessary to move all the disks from tower A to C."

Charlie: "This is incredibly boring—everybody knows that this can be solved using a simple recursion. I deny to code something as simple as this!"

The teacher sighs: "Well, Charlie, let's think about something for you to do: For you there is a fourth tower D. Calculate the smallest number of disk moves to move all the disks from tower A to tower D using all four towers."

Charlie looks irritated: "Urgh... Well, I don't know an optimal algorithm for four towers..."

OpenJudge

题目ID, 标题, 描述 25n2500011906 信箱 账号

CS101 / 题库 (包括计概、数算题目)

题目 排名 状态 提问

#50663545提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def hanoi_3(n0):
    return 2 ** n0 - 1

def hanoi_4(n0):
    if n0 == 1:
        return 1
    elif n0 == 2:
        return 3
    else:
        listy_hanoi_4 = []
        for k0 in range(1, n0):
            jit_hanoi_4 = 2 * hanoi_4(n0 - k0) + hanoi_3(k0)
            listy_hanoi_4.append(jit_hanoi_4)
        return min(listy_hanoi_4)

for i in range(1, 13):
    print(hanoi_4(i))
```

基本信息

#: 50663545
 题目: 01958
 提交人: 25n2500011906
 内存: 3548kB
 时间: 24ms
 语言: Python3
 提交时间: 2025-11-02 07:26:43

©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。

LeetCode第474场周赛，这是我第一次参加Leetcode上的周赛，结果仅用45分钟就AC3，得到12分(11:00才想起比赛的事去做题，11:45左右完成第三题)。

第 474 场周赛 排名

已结束

全国	全球	大模型	1713 153 人 AK!				
排名	用户名	得分	完成时间	题目1 (3)	题目2 (4)	题目3 (5)	题目4 (6)
387	我	12	01:22:34	00:44:46 ★5min ⏱	00:49:03 ⏱	01:12:34 ★5min ⏱	00:12:34 ⏱
1	Cranky 6agarin...	18	00:16:24	00:16:24 ⏱	00:16:12 ⏱	00:15:57 ⏱	00:15:38 ⏱
2	TsReaper 🌟	18	00:23:35	00:01:17 ⏱	00:04:28 ⏱	00:09:42 ⏱	00:18:35 ★5min ⏱
3	darrenhp-大龄...	18	00:29:21	00:01:31 ⏱	00:03:22 ⏱	00:09:40 ⏱	00:24:21 ★5min ⏱

找出缺失的元素

提交记录

989 / 989 个通过测试用例	状态: 通过
执行用时: 3 ms	提交时间: 13 小时前
内存消耗: 17.4 MB	

一次替换后的三元素最大乘积

提交记录

821 / 821 个通过测试用例

执行用时: 136 ms

内存消耗: 31.9 MB

状态: 通过

提交时间: 13 小时前

完成所有送货任务的最少时间

提交记录

1026 / 1026 个通过测试用例

执行用时: 235 ms

内存消耗: 17.5 MB

状态: 通过

提交时间: 12 小时前