Assignment #A: 递归、田忌赛马

Updated 2355 GMT+8 Nov 4, 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

M018160: 最大连通域面积

dfs similar, http://cs101.openjudge.cn/pctbook/M18160

耗时：2h

思路：自己尝试写代码，结果根本无法达到计数连通域面积的目的，于是翻开《Book My Flight》查看题解，重点学习了深度优先搜索dfs的操作方法。

题解在找最大数的时候使用不断刷新的area与之前的优胜者sur(survivor)进行俄罗斯轮盘赌，谁更大谁才能生存(sur(vive))下来，直到决出最后的赢家。这是一种创新的正确思路，也将求最大值的底层逻辑暴露出来。但我不习惯这种巧妙而又粗暴的算法，还是将所有cnt(对应题解中的area)存储在一个列表中(很多情况下更为实用)，然后取列表元素的最大值。

代码

```
import sys

dir8 = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1, 0], [1, 1]]
cnt = 0
```

```
def dfs(x, y, tara):
    global cnt
    if matrix[x][y] == tara:
        matrix[x][y] = '.'
        cnt += 1
        for dirr in dir8:
            dfs(x + dirr[0], y + dirr[1], tara)
    else:
        cnt += 0


for _ in range(int(sys.stdin.readline())):
    n, m = map(int, sys.stdin.readline().split())
    ress = [0]

    matrix = [['.' for _ in range(m + 2)] for _ in range(n + 2)]
    for row in range(1, n + 1):
        matrix[row][1: -1] = sys.stdin.readline()

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if matrix[i][j] == 'W':
                cnt = 0
                dfs(i, j, 'W')
                ress.append(cnt)

    print(max(ress))
```

代码运行截图 （至少包含有"Accepted"）

M18160:最大连通域面积

总时间限制: 1000ms 内存限制: 65536kB

描述

一个棋盘上有棋子的地方用（'W'）表示，没有的地方用点来表示，现在要找出其中的最大连通区域，一个格子被视作和它周围八个格子都相邻。

现在需要 找出最大的连通区域的面积是多少，一个格子代表面积为1。

输入

输入的第一行是一个整数，表示一共有 T 组数据。
每组第一行包含两个整数N和M。
接下来的N行，每行有M个字符('W'或者'.')，表示格子的当前状态。字符之间没有空格。

输出

每组数据对应一行，输出最大的连通域的面积，不包含任何空格。

样例输入

```
2
2 2
W.
.W
10 12
W........WW.
.WWW.....WWW
....WW...WW.
.........WW.
.........W..
..W......W..
.W.W.....WW.
W.W.W.....W.
.W.W......W.
..W.......W.
```

全局题号 **18160**

添加于 **2025-03-13**

提交次数 **261**

尝试人数 **103**

通过人数 **100**

你的提交记录

#	结果	时间
23	Accepted	2025-11-12
22	Wrong Answer	2025-11-12
21	Runtime Error	2025-11-12
20	Runtime Error	2025-11-12
19	Runtime Error	2025-11-12
18	Runtime Error	2025-11-12
17	Runtime Error	2025-11-12
16	Runtime Error	2025-11-12
15	Runtime Error	2025-11-12
14	Runtime Error	2025-11-12
13	Wrong Answer	2025-11-12
12	Wrong Answer	2025-11-12
11	Runtime Error	2025-11-12
10	Runtime Error	2025-11-12
9	Runtime Error	2025-11-12
8	Runtime Error	2025-11-12
7	Runtime Error	2025-11-12
6	Compile Error	2025-11-12
5	Compile Error	2025-11-12
4	Compile Error	2025-11-12
3	Runtime Error	2025-11-11
2	Compile Error	2025-11-10

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

#50815051提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
import sys

dir8 = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1, 0], [1
cnt = 0


def dfs(x, y, tara):
    global cnt
    if matrix[x][y] == tara:
        matrix[x][y] = '.'
        cnt += 1
        for dirr in dir8:
            dfs(x + dirr[0], y + dirr[1], tara)
    else:
        cnt += 0


for _ in range(int(sys.stdin.readline())):
    n, m = map(int, sys.stdin.readline().split())
    ress = [0]

    matrix = [['.' for _ in range(m + 2)] for _ in range(n + 2)]
    for row in range(1, n + 1):
        matrix[row][1: -1] = sys.stdin.readline()

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if matrix[i][j] == 'W':
                cnt = 0
                dfs(i, j, 'W')
                ress.append(cnt)

    print(max(ress))
```

基本信息

#: 50815051
题目: M18160
提交人: 25n2500011906(Little(25n2500011906)
内存: 3916kB
时间: 91ms
语言: Python3
提交时间: 2025-11-12 22:04:31

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

sy134: 全排列III 中等

https://sunnywhy.com/sfbj/4/3/134

耗时：5min

思路：上次在LeetCode上面写过的全排列函数再排除重复数字情况即可。

代码

```
def klnj(listy):
    return '\n'.join(listy)


def kpj(listy):
```

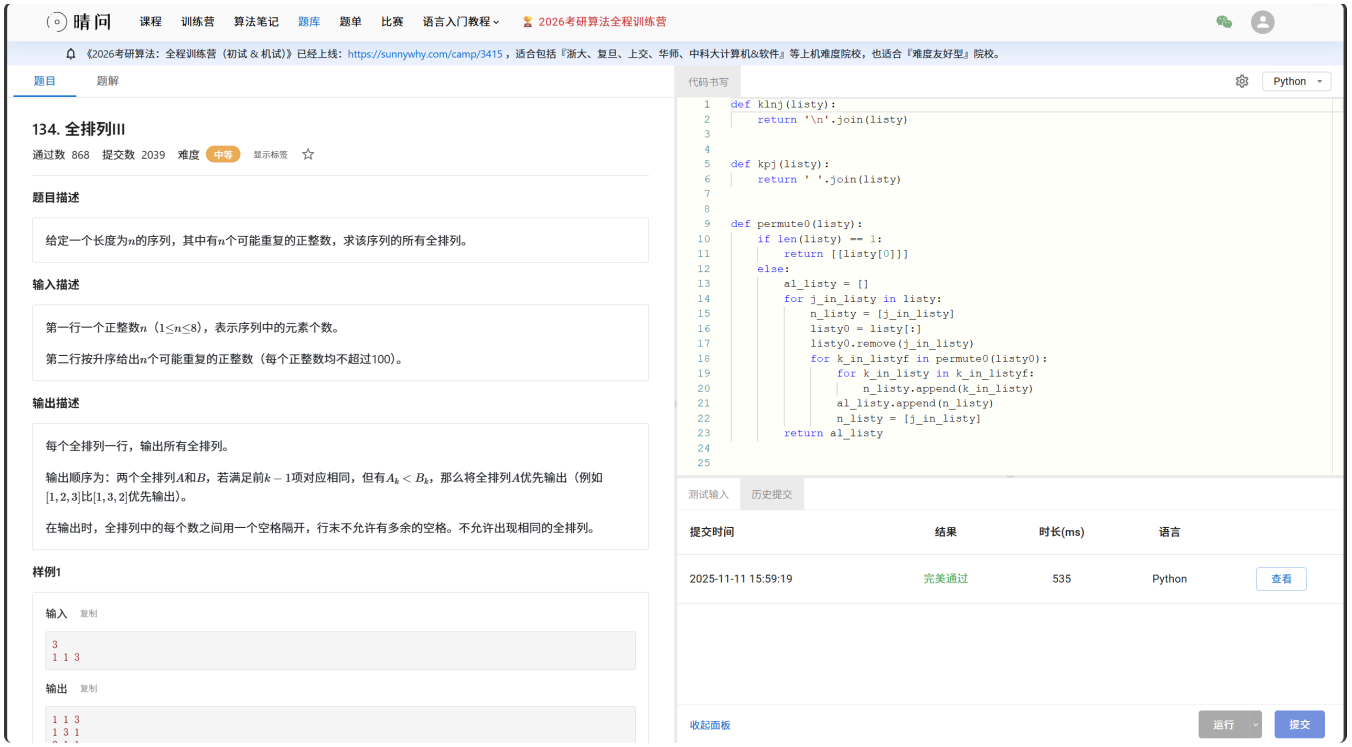
```
    return ' '.join(listy)


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


n = int(input())
js_0 = list(map(int, input().split()))
matt = permute0(js_0)
patt = []
for m in matt:
    if m not in patt:
        patt.append(m)
patt.sort()

for p in patt:
    ps = []
    for i in p:
        ps.append(str(i))
    print(kpj(ps))
```

代码运行截图 （至少包含有"Accepted"）



sy136: 组合II 中等

https://sunnywhy.com/sfbj/4/3/136

给定一个长度为的序列，其中有n个互不相同的正整数，再给定一个正整数k，求从序列中任选k个的所有可能结果。

耗时：30min

思路：先找出序列的所有子集，再从中筛选出所有长度为k的子集。

难点在找出所有子集的代码，要用到递归。

代码

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


def subsets(listy):
    res0 = [[]]
    for i0 in listy:
        res0 += [r0 + [i0] for r0 in res0]  # 每个元素扩展当前所有子集
    return res0


n, k = list(map(int, input().split()))
js = list(map(int, input().split()))

ress = []
for j in subsets(js):
```

```
    if len(j) == k:
        ress.append(j)
ress.sort()
resr = []
for r1 in ress:
    if r1 not in resr:
        resr.append([str(rr) for rr in r1])

for r2 in resr:
    print(kpj(r2))
```

代码运行截图 （至少包含有"Accepted"）

sy137: 组合III 中等

https://sunnywhy.com/sfbj/4/3/137

耗时：3min速通

思路：上一题的代码直接排除重复情况就可以了。

代码

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

```python
def subsets(listy):
    res0 = [[]]
    for i0 in listy:
        res0 += [r0 + [i0] for r0 in res0]  # 每个元素扩展当前所有子集
    return res0


n, k = list(map(int, input().split()))
js = list(map(int, input().split()))

ress = []
for j in subsets(js):
    if len(j) == k:
        ress.append(j)
ress.sort()
resr = []
for r1 in ress:
    if r1 not in resr:
        resr.append(r1)

ps = []
for r2 in resr:
    ps.append([str(i) for i in r2])

for r3 in ps:
    print(kpj(r3))
```

代码运行截图 （至少包含有"Accepted"）

M04123: 马走日

dfs, http://cs101.openjudge.cn/pctbook/M04123

耗时：2h

思路：自己做题苦思不得解，只好翻开《Book My Flight》看题解，边看边学，提交的时候竟然一直TLE，后来发现抄串行了，递归的适合套了2个dfs，导致时间复杂度应该是乘以2**n了。

然后我又探究了其他因素对代码时间的影响，发现把二维数组换成嵌套元组的列表可以在一定程度上缩短代码的运行时间（虽然时间复杂度不变），从而在一定程度上加速代码。

代码

```
dir_horse = [[-2, -1], [-2, 1], [-1, -2], [-1, 2], [1, -2], [1, 2], [2, -1], [2, 1]]

cnt = 0


def dfs(dep, x0, y0):
    if n * m == dep:
        global cnt
        cnt += 1
        return

    for r in range(len(dir_horse)):
        xt = x0 + dir_horse[r][0]
        yt = y0 + dir_horse[r][1]
        if chess[xt][yt] is False and 0 <= xt < n and 0 <= yt < m:
            chess[xt][yt] = True
            dfs(dep + 1, xt, yt)
            # 下面一行代码不容易理解，模拟一下可以发现，这行代码当且仅当所有方向都走不通时执行，其作用是退回到上一步，即回溯。
            chess[xt][yt] = False


for _ in range(int(input())):   # Lne表达的极简版
    n, m, x, y = list(map(int, input().split()))
    chess = [[False] * 10 for _ in range(10)]
    # False表示没有走过，由于棋盘的范围在函数中已经敲定，故chess的内容可以尽可能大一些。
    cnt = 0
    chess[x][y] = True    # 起点不能重复经过，记为True
    dfs(1, x, y)
    print(cnt)
```

代码运行截图 （至少包含有"Accepted"）

#50812382提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
dir_horse = [[-2, -1], [-2, 1], [-1, -2], [-1, 2], [1, -2], [1, 2], [2,

cnt = 0


def dfs(dep, x0, y0):
    if n * m == dep:
        global cnt
        cnt += 1
        return

    for r in range(len(dir_horse)):
        xt = x0 + dir_horse[r][0]
        yt = y0 + dir_horse[r][1]
        if chess[xt][yt] is False and 0 <= xt < n and 0 <= yt < m:
            chess[xt][yt] = True
            dfs(dep + 1, xt, yt)
            # 下面一行代码不容易理解，模拟一下可以发现，这行代码当且仅当所有方向都
            chess[xt][yt] = False


for _ in range(int(input())):    # Lne表达的极简版
    n, m, x, y = list(map(int, input().split()))
    chess = [[False] * 10 for _ in range(10)]
    # False表示没有走过，由于棋盘的范围在函数中已经敲定，故chess的内容可以尽可能大
    cnt = 0
    chess[x][y] = True     # 起点不能重复经过，记为True
    dfs(1, x, y)
    print(cnt)
```

基本信息

#: 50812382
题目: M04123
提交人: 25n2500011906(Little(25n2500011906)
内存: 3640kB
时间: 3678ms
语言: Python3
提交时间: 2025-11-12 20:00:12

©2002-2022 POJ 京ICP备20010980号-1

T02287: Tian Ji -- The Horse Racing

greedy, dfs http://cs101.openjudge.cn/pctbook/T02287

耗时：1h

思路：这一题融合了中国从古到今军事理论的精华，不是一般人所能轻易解决的。

本来我是按照马的速度从大到小排序，想以此让田忌的马尽可能多赢，剩下的无论如何田忌都会输，以为这就是最优解，然后两次WA；接着又从小向大排序，不出意外地再次WA(**双指针**，**帕累托改进**，凡夫俗子的策略)。

直到我拼尽全力无法战胜时，仔细研读题解复述代码，才看到了兵法的伟大之处：

1.田忌采取**快马慢马两条战线齐头并进**的策略，有利于接下来的协调；

2.能战即战，不能战胜就做出**卡尔多改进**(我的做法之所以会WA是因为使用的是帕累托改进，不如卡尔多改进激进)，就是用自己手头上最慢的马当炮灰，迎战齐王手头上最快的马，妙哉，这样既能最大程度上**消灭敌人的有生力量**，又能最大限度**保存自己的有生力量**，用**局部的惨败**换取**整体局势的反转。**

3.等到再次优势在我时，全面发动反击，不断取得胜利，如此循环往复直至比赛结束。

总结：田忌赛马融合了*军事理论、传统文学、历史、数学、计算机、经济学、管理学*等多方面知识，是人类智慧的精华。尤其其中所蕴含的兵法*"知己知彼百战百胜"、"统筹规划齐头并进"、"舍小得失顾全大局"、"消灭敌人有生力量的同时尽可能保留自己的有生力量"*等在绝大多数**双方作战**（包括*军事战争、商战贸易战、信息战、心理战、双方对峙竞争、人工智能算法开发与升级*等）的情境都适用。

代码

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

    tj = list(map(int, input().split()))
    tj.sort()
    gw = list(map(int, input().split()))
    gw.sort()

    lt = 0
    rt = n - 1
    lg = 0
    rg = n - 1
    cnt = 0
    while lt <= rt:
        if tj[lt] > gw[lg]:
            cnt += 1
            lt += 1
            lg += 1
        elif tj[rt] > gw[rg]:
            cnt += 1
            rt -= 1
```

```
            rg -= 1
        else:
            if tj[lt] < gw[rg]:
                cnt -= 1

            lt += 1
            rg -= 1

    print(200 * cnt)
```

代码运行截图 （至少包含有"Accepted"）

2. 学习总结和收获

解决简单题的速度越来越快了。

对dfs深度优先搜索的理解还是不够深入，运用很不熟练，不能灵活运用，需要多加练习。

本次主要通过阅读理解题解来积累解题经验。学会了使用sys.stdin.readline()处理输入数据，使用global查找变量，更学习了基础的dfs思路，复习了递归算法。

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。

周四下午的模拟考试我在未做任何准备的情况之下参加了（做充分准备很可能也只能AC2）。我15:16才开始做题，相当于10min秒杀了前两道简单题；接下来四道题却难得离谱，我也没有练习过类似的题目，将近两个小时楞是没做出来任意一道题。还是要加强训练。

OpenJudge 题目ID, 标题, 描述 25n2500011906 信箱 账号

CS101 / 20251113 cs101 Mock Exam (hosted by TA) 已经结束

题目 排名 状态 统计 提问

比赛已经结束

2025-11-13 15:08:00 开始时间

2025-11-13 17:00:00 结束时间

2025年11月TA出的计概（2025fall-cs101: DS Algo DS-1班）课程模拟考试。
请独立完成，不能通讯，如：不能使用微信、邮件、QQ等工具。
考试期间，请同学只访问OJ，不能访问其他网站，不要查看OJ考试之前自己提交的代码。
考试过程中允许可以带10张A4纸大小的cheat sheet，以及空白草稿纸。
题目编号前面的大写字母，相应表明是 Easy/Medium/Tough 级别。
————-
登录别人的账号即视为违纪甚至作弊。把自己的账号密码告诉别人，被别人登录，也视为违纪甚至作弊。如果考前别人用过你的账号，请立即修改密码。

请把你的昵称改为 25nxxxxx, 后面部分是学号。http://cs101.openjudge.cn/mine
有同学昵称24n, 23n, ..., 19n开始也是可以的，学号别错，才能找到你的成绩。

	题目ID	标题	通过率	通过人数	尝试人数
✓	E07592	求最大公约数问题	100%	48	48
✓	E27273	简单的数学题	100%	46	46
	M28749	Top2招生工作	11%	1	9
	M29954	逃离紫罗兰监狱	65%	15	23
	T01941	The Sierpinski Fractal	92%	12	13
	T28702	合理的饭票设计	50%	1	2

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

32	25n2500011906(Little(25n2500011906)	2	00:51:23	00:13:25 (-1)	00:17:58

我又随手做了Openjudge和sunnywhy网站上面的一些简单题练习遇到不同简单题的处理方法思路。

✓	132	全排列I 递归 深度优先搜索	2104 / 4080	中等
✓	133	全排列II 递归 深度优先搜索	1452 / 2325	中等
✓	134	全排列III 递归 深度优先搜索	885 / 2074	中等
✓	135	组合I 递归 深度优先搜索	1325 / 2607	中等
✓	136	组合II 递归 深度优先搜索	1192 / 1792	中等
✓	137	组合III 递归 深度优先搜索	627 / 1217	中等

07743:计算矩阵边缘元素之和

查看 | 提交 | 统计 | 提问

总时间限制: 1000ms　内存限制: 65536kB

全局题号 **7743**

添加于 **2025-09-26**

提交次数 **1**

尝试人数 **1**

通过人数 **1**

你的提交记录

#	结果	时间
1	Accepted	2025-11-13

描述

输入一个整数矩阵，计算位于矩阵边缘的元素之和。所谓矩阵边缘的元素，就是第一行和最后一行的元素以及第一列和最后一列的元素。

输入

第一行分别为矩阵的行数m和列数n（m < 100，n < 100），两者之间以一个空格分开。
接下来输入的m行数据中，每行包含n个整数，整数之间以一个空格分开。

输出

输出对应矩阵的边缘元素和

样例输入

```
3 3
3 4 1
3 7 1
2 0 1
```

样例输出

```
15
```

E02913:加密技术

总时间限制: 1000ms 内存限制: 65536kB

描述

编制程序，将输入的一行字符加密解密。加密时，每个字符依次反复加上"4962873"中的数字，如果范围超过ASCII码的032(空格)~122('z'),则进行模运算。解密和加密的顺序相反。编制加密解密函数，打印各个过程的结果。

例如：如果是this is a book!
密文应该是：
't'+4，'h'+9，'i'+6，'s'+2，' '+8，'i'+7，'s'+3，' '+4，'a'+9，'
'+6，'b'+2，'o'+8，'o'+7，'k'+3，'!'+4

输入

输入一行字符串，其中包含了若干空格。

输出

对输入字符串进行加密，并输出加密结果。
再对输入字符串进行解密，并在换行后输出解密结果。

样例输入

```
aghi lrtq  haha
```

样例输出

```
epnk(suxz&"phke
aghi lrtq  haha
```


©2002-2022 POJ 京ICP备20010980号-1

E03248:最大公约数

总时间限制: 1000ms 内存限制: 65536kB

描述

给定两个正整数，求它们的最大公约数。

输入

有多组数据，每行为两个正整数，且不超过int可以表示的范围。

输出

行对应输出最大公约数。

样例输入

```
4 8
8 6
200 300
```

样例输出

```
4
2
100
```

提示

系统的测试文件中数据有很多组，因此同学们在程序里要写循环读取数据并判断是否读完文件的代码。
如果不知道如何处理，可以参考下面的两个模板：
C++这样写：

全局题号 2249
添加于 2025-03-31
提交次数 278
尝试人数 163
通过人数 157

你的提交记录

#	结果	时间
5	Accepted	2025-11-13
4	Runtime Error	2025-10-23
3	Wrong Answer	2025-10-23
2	Runtime Error	2025-10-23
1	Wrong Answer	2025-09-18