

assignment A

Assignment #A : 递归、田忌赛马

Updated 2355 GMT+8 Nov 4, 2025

2025 fall, Comptled 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://typorao.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

M18160: 最大连通域面积

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

思路：

代码

```
test_cases = int(input())
for i in range(test_cases):
    N,M = map(int,input().split())
    matrix = []
    visited = [[False]*M for _ in range(N)]
    for j in range(N):
        matrix.append(list(input().strip()))

def dfs(x,y):
    if x < 0 or x >= N or y < 0 or y >= M:
        return 0
    if visited[x][y] or matrix[x][y] == '.':
        return 0
    visited[x][y] = True
    count = 1
    direction = [(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (1, -1), (-1, 1), (-1, -1)]
    for dx, dy in direction:
        count += dfs(x + dx, y + dy)
    return count

max_s = 0
for x in range(N):
    for y in range(M):
        count = 0
        if not visited[x][y] and matrix[x][y]=='W':
            current_size = dfs(x,y)
            max_s = max(max_s,current_size)
print(max_s)
```

代码运行截图 (至少包含有 "Accepted")

#50801524 提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
test_cases = int(input())
for i in range(test_cases):
    N, M = map(int, input().split())
    matrix = []
    visited = [[False]*M for _ in range(N)]
    for j in range(N):
        matrix.append(list(input().strip()))

    def dfs(x, y):
        if x < 0 or x >= N or y < 0 or y >= M:
            return 0
        if visited[x][y] or matrix[x][y] == '.':
            return 0
        visited[x][y] = True
        count = 1
        direction = [(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (1, -1),
                     for dx, dy in direction:
                        count += dfs(x + dx, y + dy)
        return count

    max_s = 0
    for x in range(N):
        for y in range(M):
            count = 0
            if not visited[x][y] and matrix[x][y] == 'W':
                current_size = dfs(x, y)
                max_s = max(max_s, current_size)
    print(max_s)
```

基本信息

#: 50801524
题目: M18160
提交人: R.
内存: 3680kB
时间: 111ms
语言: Python3
提交时间: 2025-11-11 23:59:07

sy134: 全排列III 中等

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

思路:

代码

```
n = int(input())
num_list = sorted(list(map(int, input().split())))
def permute(num_list):
    def backtrack(used, temp):
        if len(temp) == n:
            if temp not in res:
                res.append(temp[:])
            return
        for x in range(len(num_list)):
            if not used[x]:
                used[x] = True
                temp.append(num_list[x])
                backtrack(used, temp)
                temp.pop()
                used[x] = False
    res = []
    backtrack(used, temp)
```

```
        backtrack(used,temp)
        temp.pop()
        used[x]=False
    res = []
    used = [False]*n
    backtrack(used,[])
    return res
ans = permute(num_list)
for i in range(len(ans)):
    print(*ans[i])
```

代码运行截图 (至少包含有 "Accepted")

考研算法全程训练营



浙大、复旦、上交、华师、中科大计算机&软件』等上机

RoseRong

难度

代码书写

```
1 n = int(input())
2 num_list = sorted(list(map(int, input().split())))
3 def permute(num_list):
4     def backtrack(used, temp):
5         if len(temp) == n:
6             if temp not in res:
7                 res.append(temp)
8             return
9         for x in range(len(num_list)):
10            if not used[x]:
11                used[x] = True
12                temp.append(num_list[x])
13                backtrack(used, temp)
14                temp.pop()
15                used[x] = False
16    res = []
17    used = [False] * n
18    backtrack(used, [])
19    return res
```

个人主页

我的收藏

题目统计

深色主题

意见反馈

退出登录

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: 463 ms

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。

思路：

代码

```
n, k = map(int, input().split())
num_list = sorted(list(map(int, input().split())))
def combination(num_list):
    def traceback(used, temp, a):
        if len(temp) == k:
            if temp not in res:
                res.append(temp[:])
            return
        for x in range(a, n):
            if not used[x]:
                temp.append(num_list[x])
                used[x] = True
                a += 1
                traceback(used, temp, a)
                temp.pop()
                used[x] = False

    used = [False]*n
    res = []
    traceback(used, [], 0)
    return res
ans = combination(num_list)
for _ in range(len(ans)):
    print(*ans[_], sep=' ')
```

代码运行截图 (至少包含有 "Accepted")

🏆 2026考研算法全程训练营



RoseRong

适合包括『浙大、复旦、上交、华师、中科大计算机&软件』等上机

困难

代码书写

```
5         if len(temp)==k:
6             if temp not in res:
7                 res.append(temp[:])
8             return
9         for x in range(a,n):
10            if not used[x]:
11                temp.append(num_list[x])
12                used[x]=True
13                a+=1
14                traceback(used,temp,a)
15                temp.pop()
16                used[x]=False
17
18        used = [False]*n
19        res = []
20        traceback(used,[],0)
21        return res
22    ans = combination(num_list)
23    for i in range(len(ans)):
```

测试输入

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](#)

运行时长: 0 ms

sy137: 组合III 中等

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

思路：

代码

```
n,k = map(int,input().split())
num_list = sorted(list(map(int,input().split())))
def combination(num_list):
    def traceback(used,temp,a):
        if len(temp)==k:
            if temp not in res:
                res.append(temp[:])
            return
        for x in range(a,n):
            if not used[x]:
                temp.append(num_list[x])
                used[x]=True
                a+=1
                traceback(used,temp,a)
                temp.pop()
                used[x]=False

    used = [False]*n
    res = []
    traceback(used,[],0)
    return res
ans = combination(num_list)
for _ in range(len(ans)):
    print(*ans[_],sep=' ')
```

代码运行截图 (至少包含有 "Accepted")

126 考研算法全程训练营



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

```
1 n, k = map(int, input().split())
2 num_list = sorted(list(map(int, in
3 def combination(num_list):
4     def traceback(used, temp, a):
5         if len(temp) == k:
6             if temp not in res:
7                 res.append(temp[:])
8             return
9         for x in range(a, n):
10            if not used[x]:
11                temp.append(num_list[x])
12                used[x] = True
13                a += 1
14                traceback(used, temp, a)
15                temp.pop()
16                used[x] = False
17
18 used = [False] * 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

Mo4123: 马走日

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

思路：

Knight Tour

*main*函数：

```
def knight_tour():
    matrix
    directions
    count[0]=1
```

def is_valid:

```
def dfs(x,y,steps,matrix):
    if steps==m*n:
        count[0]+=1
        return
    for dx,dy in directions:
        nx,ny = dx+x,dy+y
        if is_valid(nx,ny):
            matrix[nx][ny]=1
            dfs(nx,ny,steps+1,matrix)
            matrix[nx][ny]=0
    matrix[x][y]=1
dfs(x,y,1,matrix)
print(count[0])
```

knight_tour()

代码

```
def horse_tour():
    test_cases = int(input())
    for i in range(test_cases):
        n, m, x, y = map(int, input().split())
        matrix = [[0] * m for _ in range(n)]
        directions = [(1,2),(1,-2),(-1,2),(-1,-2),(2,1),(2,-1),
                      (-2,1),(-2,-1)]
```

```
count = [0]

def is_valid(x,y):
    return 0 <= x < n and 0 <= y < m and matrix[x][y] == 0

def dfs(x,y,steps,matrix):
    if steps==m*n:
        count[0]+=1
        return

    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if is_valid(nx, ny):
            matrix[nx][ny] += 1
            dfs(nx, ny, steps + 1, matrix)
            matrix[nx][ny] -= 1

    matrix[x][y] = 1
    dfs(x,y,1,matrix)
    print(count[0])

horse_tour()
```

代码运行截图 (至少包含有 "Accepted")

#50853847 提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def horse_tour():
    test_cases = int(input())
    for i in range(test_cases):
        n, m, x, y = map(int, input().split())
        matrix = [[0] * m for _ in range(n)]
        directions = [(1, 2), (1, -2), (-1, 2), (-1, -2), (2, 1), (2, -1), (-2, 1), (-2, -1)]
        count = [0]

        def is_valid(x, y):
            return 0 <= x < n and 0 <= y < m and matrix[x][y] == 0

        def dfs(x, y, steps, matrix):
            if steps == m * n:
                count[0] += 1
                return

            for dx, dy in directions:
                nx, ny = x + dx, y + dy
                if is_valid(nx, ny):
                    matrix[nx][ny] += 1
                    dfs(nx, ny, steps + 1, matrix)
                    matrix[nx][ny] -= 1

            matrix[x][y] = 1
            dfs(x, y, 1, matrix)
            print(count[0])

    horse_tour()
```

基本信息

#: 50853847
题目: M04123
提交人: R.
内存: 3684kB
时间: 3981ms
语言: Python3
提交时间: 2025-11-15 17:21:01

To2287: Tian Ji -- The Horse Racing

greedy, dfs <http://cs101.openjudge.cn/pctbook/To2287>

思路:

可能会出现的typo: (测试的时候发现)

10

11 10 9 8 7 6 5 4 3 2

9 8 7 6 5 4 3 2 1

primitive_version:

11=11, 田比不过王, 这时需要一个田的慢马来消耗王的快马

派出2, 划去11、2

此后一路畅通, 田全赢, money=8 200

但是有更优解:

11 10 9 8 7 6 5 4 3 2

11 9 8 7 6 5 4 3 2 1

我们让11=11实现平局, 剩下田全赢, money=9 200

因此“=”是一个不好处理的情况

复盘一下2个过程：

process_1 : 11>2,3>1,11>9,10>8,etc.

process_2 : 11=11,2>1,10>9,9>8,etc.

于是尝试考虑在这种情况下比较慢马，不轻易移动*start*的值

遂AC

else:

最快的马速度相等

if tianji_horse[t_end] > king_horse[k_end]:

田忌最慢的马能赢国王最慢的马

win += 1

t_end -= 1

k_end -= 1

赢不了会怎样

还是慢马来消耗快马

else:

用田忌最慢的马对齐王最快的马

这里是简化的代码

if tianji_horse[t_end] < king_horse[k_start]:

win -= 1

(实际上这时的状态有点意思：

我们已知王的快马等于田的快马，如果*tianji_horse[t_end]*

*king_horse[k_start]*不满足，那么绝对不可能大于，只能是等于

于是：*tianji_horse[t_end]=king_horse[k_start]=tianji_horse[t_end]*，这说明田忌的所有马都相等

话说回来，平局无需处理，一样的*code*遂合并)

t_end -= 1

k_start += 1

代码

```
while True:
    n = int(input())
    if n == 0:
        break
    else:
        tianji_horse = sorted(list(map(int, input().split())),
reverse=True)
```

```
king_horse = sorted(list(map(int, input().split())),
reverse=True)

win = 0
t_start, t_end = 0, n - 1 # 田忌指针
k_start, k_end = 0, n - 1 # 国王指针

for _ in range(n):
    if tianji_horse[t_start] > king_horse[k_start]:
        # 田忌最快的马能赢国王最快的马
        win += 1
        t_start += 1
        k_start += 1
    elif tianji_horse[t_start] < king_horse[k_start]:
        # 田忌最快的马输给国王最快的马，用最慢的马消耗
        win -= 1
        t_end -= 1
        k_start += 1
    else:
        # 最快的马速度相等
        if tianji_horse[t_end] > king_horse[k_end]:
            # 田忌最慢的马能赢国王最慢的马
            win += 1
            t_end -= 1
            k_end -= 1
        # 赢不了会怎样
        # 还是慢马来消耗快马
    else:
        # 用田忌最慢的马对齐王最快的马
        # 这里是简化的代码
        if tianji_horse[t_end] < king_horse[k_start]:
            win -= 1
        # 实际上这时
        t_end -= 1
        k_start += 1

print(win * 200)
```

代码运行截图 (至少包含有 "Accepted")

#50857669提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
while True:
    n = int(input())
    if n == 0:
        break
    else:
        tianji_horse = sorted(list(map(int, input().split()))), reverse=False
        king_horse = sorted(list(map(int, input().split()))), reverse=True

        win = 0
        t_start, t_end = 0, n-1 # 田忌指针
        k_start, k_end = 0, n-1 # 国王指针

        for _ in range(n):
            if tianji_horse[t_start] > king_horse[k_start]:
                # 田忌最快的马能赢国王最快的马
                win += 1
                t_start += 1
                k_start += 1
            elif tianji_horse[t_start] < king_horse[k_start]:
                # 田忌最快的马输给国王最快的马, 用最慢的马消耗
                win -= 1
                t_end -= 1
                k_start += 1
            else:
                # 最快的马速度相等
                if tianji_horse[t_end] > king_horse[k_end]:
                    # 田忌最慢的马能赢国王最慢的马
                    win += 1
                    t_end -= 1
                    k_end -= 1
                else:
                    # 用田忌最慢的马对齐王最快的马
                    if tianji_horse[t_end] < king_horse[k_start]:
                        win -= 1
                        t_end -= 1
                        k_start += 1

print(win * 200)
```

基本信息

#: 50857669
题目: T02287
提交人: R.
内存: 3788kB
时间: 54ms
语言: Python3
提交时间: 2025-11-15 21:39:27

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。

最近在恶补期中落下的内容，逐渐有了些思考。

下周会把mock exam和每日选做的题完成，加油！

状态: Accepted

源代码

```
a,b = map(int,input().split())
if b>a:
    a,b = b,a
while True:
    if a%b==0:
        print(b)
        break
    if a>b:
        a,b = b,a%b
```

基本信息

#: 50822758
 题目: E07592
 提交人: R.
 内存: 3584kB
 时间: 20ms
 语言: Python3
 提交时间: 2025-11-13 15:36:49

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

状态: Accepted

源代码

```
test_cases = int(input())
for j in range(test_cases):
    n = int(input())
    ans = (1+n)*n//2
    i = 0
    while True:
        if 2**i<=n<2** (i+1):
            break
        i+=1
    ans-= (2** (i+1)-1)*2
    print(ans)
```

基本信息

#: 50823821
 题目: E27273
 提交人: R.
 内存: 3608kB
 时间: 23ms
 语言: Python3
 提交时间: 2025-11-13 16:11:29

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

M29954 逃离紫罗兰监狱

是一道很有收获的题

学会了BFS的基本思路

源代码

```
from collections import deque
def escape():
    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
    visited = [[[False] * (K + 1) for _ in range(C)] for _ in range(R)]
    queue = deque()
    queue.append((s_x, s_y, K, 0))
    visited[s_x][s_y][K] = True
    while queue:
        x, y, k, steps = queue.popleft()
        if x == e_x and y == e_y:
            return steps
        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if 0 <= nx < R and 0 <= ny < C:
                if prison[nx][ny] == '.' or prison[nx][ny] == 'E':
                    if not visited[nx][ny][k]:
                        visited[nx][ny][k] = True
                        queue.append((nx, ny, k, steps + 1))
                elif prison[nx][ny] == '#' and k > 0:
                    if not visited[nx][ny][k - 1]:
                        visited[nx][ny][k - 1] = True
                        queue.append((nx, ny, k - 1, steps + 1))
    return -1
R, C, K = map(int, input().split())
prison = []
for i in range(R):
    prison.append(list(input()))
s_x, s_y, e_x, e_y = -1, -1, -1, -1
for x in range(R):
    for y in range(C):
        if prison[x][y] == 'S':
            s_x, s_y = x, y
        elif prison[x][y] == 'E':
            e_x, e_y = x, y
print(escape())
```

code:

```
from collections import deque
def escape():
    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
    visited = [[[False] * (K + 1) for _ in range(C)] for _ in range(R)]
    # A
    queue = deque()
    queue.append((s_x, s_y, K, 0))
    visited[s_x][s_y][K] = True
    while queue:
        x, y, k, steps = queue.popleft()
        if x == e_x and y == e_y:
            return steps
        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if 0 <= nx < R and 0 <= ny < C:
                if prison[nx][ny] == '.' or prison[nx][ny] == 'E':
```

至少归心

#: 50824978
题目: M29954
提交人: R.
内存: 5412kB
时间: 189ms
语言: Python3
提交时间: 2025-11-13 16:59:04

```

        if not visited[nx][ny][k]:
            visited[nx][ny][k] = True
            queue.append((nx, ny, k, steps + 1))
        elif prison[nx][ny] == '#' and k > 0:
            if not visited[nx][ny][k - 1]:
                visited[nx][ny][k - 1] = True
                queue.append((nx, ny, k - 1, steps + 1))

    return -1

R, C, K = map(int, input().split())
prison = []
for i in range(R):
    prison.append(list(input()))
s_x, s_y, e_x, e_y = -1, -1, -1, -1
for x in range(R):
    for y in range(C):
        if prison[x][y] == 'S':
            s_x, s_y = x, y
        elif prison[x][y] == 'E':
            e_x, e_y = x, y
print(escape())

```

A: 由于有 \mathcal{K} , 建立一个三维数组, $(\mathcal{K}+1)$ 表示状态数

或者理解 $(\mathcal{K}+1)$ 层, 当 $\mathcal{K}-1$ 时跌落一层 (因为当 \mathcal{K} 不同时从 (x_1, y_1) 到 (x_2, y_2) 的走法是不同的, 层数的跌落对于最后的答案并无影响)

B: *return* 的设计:

因为第一个*return*的一定是最小值, 因此直接*return*解除函数调用即可; 若没有最小值出现, 则*return(-1)*

C: 最短距离问题使用 \mathcal{BFS} (广度优先搜索), 而不是 \mathcal{DFS} (深搜)

\mathcal{BFS} 优先遍历周围的邻居, 再从邻居“扩散”, 有一条到终点的路径后输出答案停止

\mathcal{DFS} 每次向前一步, 直至终点, 这时再返回, 全部遍历后取最优解, 明显不如 \mathcal{BFS}