Assignment #10: dp & bfs

Updated 2 GMT+8 Nov 25, 2024

2024 fall, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院

说明：

1）请把每个题目解题思路（可选），源码 Python, 或者 C++（已经在 Codeforces/Openjudge 上 AC），截图（包含 Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用 word）。AC 或者没有 AC，都请标上每个题目大致花费时间。

2）提交时候先提交 pdf 文件，再把 md 或者 doc 文件上传到右侧“作业评论”。Canvas 需要有同学清晰头像、提交文件有 pdf、"作业评论"区有上传的 md 或者 doc 附件。

3）如果不能在截止前提交作业，请写明原因。

1. 题目

LuoguP1255 数楼梯

dp, bfs, https://www.luogu.com.cn/problem/P1255

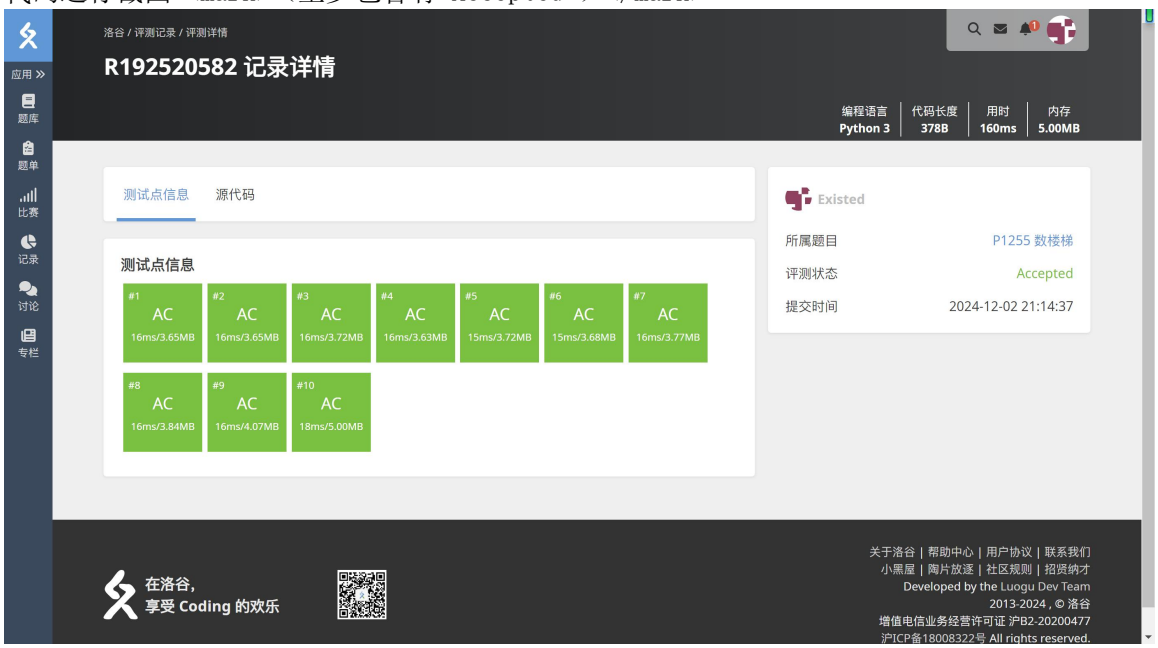
思路：这道题目原本可以用 dp 来做，但是因为我对 dp 比较熟悉了，而对 dfs 的模板还不太熟悉，于是用 dfs 写了一遍。
之前没遇到过递归爆栈的问题，这次遇到了（因为数据太大了），于是学会了如何用 sys 模块更改系统默认递归深度~

代码：

```python

```
```
import sys
sys.setrecursionlimit(10**5)#设置递归深度限制（默认 1000）

n = int(input())
memo = [0]*(n+1)#一定要有一个负责储存记忆的列表，否则必 tle
```

```
def dfs(m):
    if memo[m]:#先检查是否已经计算过 m 时的情况
        return memo[m]

    elif m == 1:
        memo[m] = 1
    elif m == 2:
        memo[m] = 2
    else:
        memo[m] = dfs(m-1) + dfs(m-2)

    return memo[m]#统一返回给上一级函数

print(dfs(n))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



27528: 跳台阶

dp, http://cs101.openjudge.cn/practice/27528/

思路：作为数院的同学，这道题一眼就能看出来答案是 2^(n-1)（用小学就学过的隔板法），但是如果这样写作为答案就有点没意思了，于是我就效仿上道题几乎一模一样的方法，用 dfs 把这道题写了一遍，也很快就 ac 了~

代码：

```python

```

```
n = int(input())
memo = [0]*(n+1)

def dfs(m):
    if memo[m]:#先检查是否已经计算过 m 时的情况
        return memo[m]

    elif m == 1:
        memo[m] = 1
    else:
        memo[m] = sum(dfs(m-i) for i in range(1,m)) + 1

    return memo[m]#统一返回给上一级函数

print(dfs(n))
```

代码运行截图 ==（至少包含有"Accepted"）==

474D. Flowers

dp, https://codeforces.com/problemset/problem/474/D

思路：先推递推公式：记 an 为 n 个 flower 时的总方法数，bn 为 n 个 flowers 时以 W 结尾的方法数；找 bn 与 an 互相之间的关系：
显然 bn=an-k（特例：bk=1），而 an=bn+bn-1+...+bk+1
由此可看出，n 大于 k 时，an=an-k+...+a1+1（对应 bk）+1（对应全是 R）
用这个公式就可推出 an 了！

与此同时，第二步要求 a~b 项的和时，不要直接求和（显然会 tle）
要利用 an 的公式！不难推得，和=a(b+k)-a(a+k-1)！
（）为了让 b+k 仍在 dp 列表的范围内，需要将长度扩大至 2 倍的 10**5！）

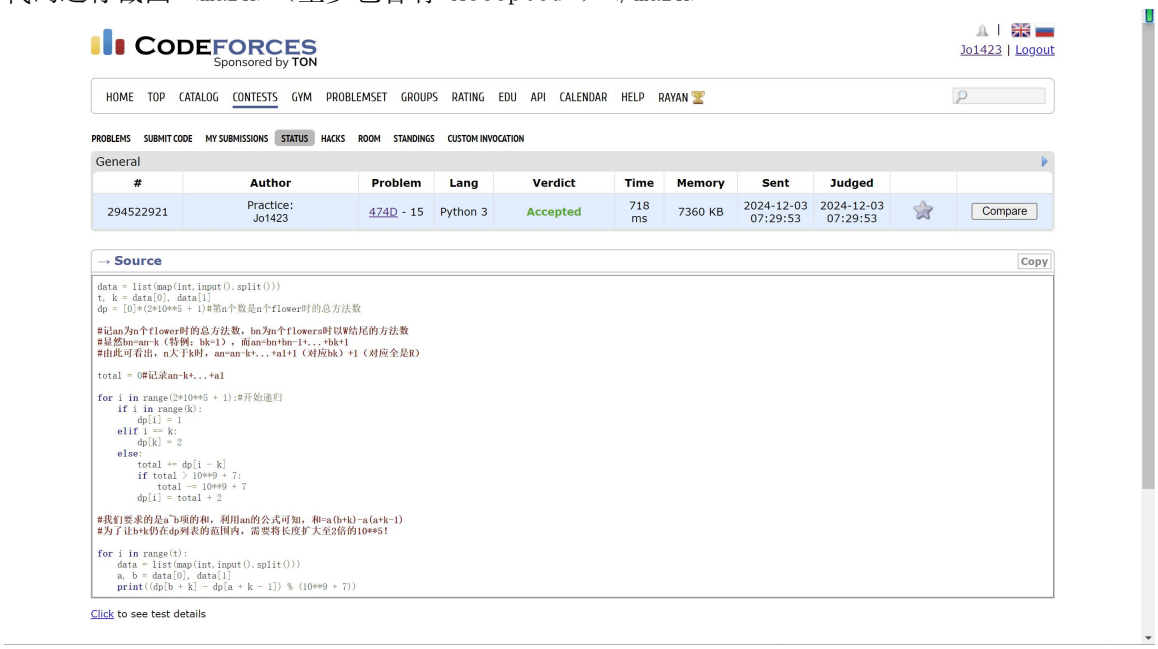
代码：

```python

```
```
data = list(map(int,input().split()))
t, k = data[0], data[1]
dp = [0]*(2*10**5 + 1)#第 n 个数是 n 个 flower 时的总方法数

#记 an 为 n 个 flower 时的总方法数，bn 为 n 个 flowers 时以 W 结尾的方法数
#显然 bn=an-k（特例：bk=1），而 an=bn+bn-1+...+bk+1
#由此可看出，n 大于 k 时，an=an-k+...+a1+1（对应 bk）+1（对应全是 R）

total = 0#记录 an-k+...+a1

for i in range(2*10**5 + 1):#开始递归
    if i in range(k):
        dp[i] = 1
    elif i == k:
        dp[k] = 2
    else:
        total += dp[i - k]
        if total > 10**9 + 7:
            total -= 10**9 + 7
        dp[i] = total + 2

#我们要求的是 a~b 项的和，利用 an 的公式可知，和=a(b+k)-a(a+k-1)
```

#为了让 b+k 仍在 dp 列表的范围内，需要将长度扩大至 2 倍的 10**5!

```
for i in range(t):
    data = list(map(int,input().split()))
    a, b = data[0], data[1]
    print((dp[b + k] - dp[a + k - 1]) % (10**9 + 7))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



LeetCode5.最长回文子串

dp, two pointers, string,
https://leetcode.cn/problems/longest-palindromic-substring/

思路：直接暴力遍历即可，注意要分奇数长度和偶数长度分情况讨论

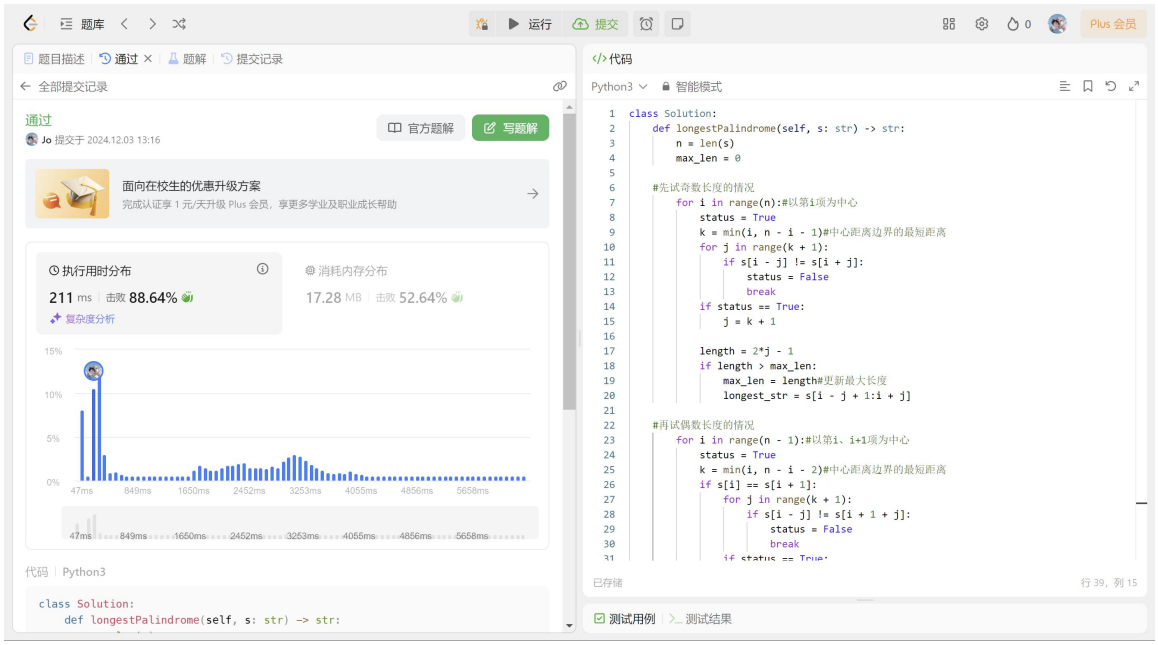
代码：

```python
```

```
class Solution:
    def longestPalindrome(self, s: str) -> str:
```

```
        n = len(s)
        max_len = 0

    #先试奇数长度的情况
        for i in range(n):#以第 i 项为中心
            status = True
            k = min(i, n - i - 1)#中心距离边界的最短距离
            for j in range(k + 1):
                if s[i - j] != s[i + j]:
                    status = False
                    break
            if status == True:
                j = k + 1

            length = 2*j - 1
            if length > max_len:
                max_len = length#更新最大长度
                longest_str = s[i - j + 1:i + j]

    #再试偶数长度的情况
        for i in range(n - 1):#以第 i、i+1 项为中心
            status = True
            k = min(i, n - i - 2)#中心距离边界的最短距离
            if s[i] == s[i + 1]:
                for j in range(k + 1):
                    if s[i - j] != s[i + 1 + j]:
                        status = False
                        break
                if status == True:
                    j = k + 1

                length = 2*j
                if length > max_len:
                    max_len = length#更新最大长度
                    longest_str = s[i - j + 1:i + j + 1]

        return(longest_str)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



12029: 水淹七军

bfs, dfs, http://cs101.openjudge.cn/practice/12029/

思路：先用 height 列表接收高度数据；可随时修改，视为当下的“表面高度”（验证一个点是否已经被淹时，若“表面高度”等于原高度，则说明这是陆地高度，未被淹；反之，则说明是水面高度，已经被淹）
随后，用 bfs 方法，建立 deque，每一步删去当前 deque 最靠前的点，并对该点处的水流的四个方向进行分析；若水流可以流动到某个点，则更新该点的“表面高度”为水流高度，并将该点加入到 deque 的结尾即可~

代码：

```python
```
from collections import deque

def bfs(x, y):
    global height
    queue = deque([(x,y)])
    directions = [(-1,0),(1,0),(0,-1),(0,1)]

```
    while queue:
        x, y = queue.popleft()#对当前 deque 最靠前的点的水流分析
        for dir in directions:
            nx, ny = x + dir[0], y + dir[1]
            if nx in range(m) and ny in range(n) and height[nx][ny] < height[x][y]:
                height[nx][ny] = height[x][y]
                queue.append((nx,ny))

import sys
lines=list(sys.stdin.read().split())#一次性读取所有数据，并按空格划分开，合并为一个列表

k = int(lines[0])
idx = 1#目前检索到的行数
for i in range(k):
    height = []#记录高度，随时更改，可视为当下的“表面高度”
    #（若“表面高度”等于原高度，则说明这是陆地高度，未被淹；反之则说明是水面高度，已经被淹）
    m, n = map(int,lines[idx:idx + 2])
    idx += 2
    for j in range(m):
        data = list(map(int,lines[idx:idx + n]))
        height.append(data)
        idx += n

    data = list(map(int,lines[idx:idx + 2]))
    idx += 2
    a, b = data[0] - 1, data[1] - 1#记录司令部位置
    t = height[a][b]
    p = int(lines[idx])
    idx += 1
    for j in range(p):
        data = list(map(int,lines[idx:idx + 2]))
        idx += 2
        x, y = data[0] - 1, data[1] - 1
        bfs(x, y)

    if t < height[a][b]:
        print("Yes")
    else:print("No")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

02802: 小游戏

bfs, http://cs101.openjudge.cn/practice/02802/

思路：太难了，想了一两个小时还是 wa，只能看题解了 qwq（读了两遍才完全理解）

代码：

```python

```

```
import heapq
num1=1
while True:
    w,h=map(int,input().split())
    if w==0 and h==0:
        break
    print(f"Board #{num1}:")
    martix=[[" "]*(w+2)]+[[" "]+list(input())+[" "] for _ in range(h)]+[[" "]*(w+2)]
    dir=[(0,1),(0,-1),(1,0),(-1,0)]
    num2=1
    while True:
        x1,y1,x2,y2=map(int,input().split())
        if x1==0 and x2==0 and y1==0 and y2==0:
```

```
            break
        queue,flag=[],False
        vis=set()
        heapq.heappush(queue,(0,x1,y1,-1))
        martix[y2][x2]=" "
        vis.add((-1,x1,y1))
        while queue:
            step,x,y,dirs=heapq.heappop(queue)
            if x==x2 and y==y2:
                flag=True
                break
            for i,(dx,dy) in enumerate(dir):
                px,py=x+dx,y+dy
                if 0<=px<=w+1 and 0<=py<=h+1 and (i,px,py) not in vis and martix[py][px]!="X":
                    vis.add((i,px,py))
                    heapq.heappush(queue,(step+(dirs!=i),px,py,i))
        if flag:
            print(f"Pair {num2}: {step} segments.")
        else:
            print(f"Pair {num2}: impossible.")
        martix[y2][x2]="X"
        num2+=1
    print()
    num1+=1
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

2. 学习总结和收获

<mark>如果作业题目简单，有否额外练习题目，比如：OJ“计概 2024fall 每日选做”、CF、LeetCode、洛谷等网站题目。</mark>

前两题难度都很简单，很快就能做出来~

第三题稍有难度，我一开始没想到递推公式，因为想直接推 an 的递推公式，但是想了好一会都没想出来；后来我把 bn（以 W 为结尾的方法数）也考虑进来了之后，发现 an 和 bn 可以互相之间很简单的表示，于是自然而然也就推出了 an 的公式了~总的来讲感觉难度不大，我不知道其他同学的方法是什么，反正我这个方法只要想到了 bn 就很容易得到 an 的公式啦

第四题我一开始没想到很简单的方法，直接暴力遍历，也能过（因为题目对 n 的规模限制的很小，只限制在 10**3 内，但是如果是 10**4 可能就不一定行了，因为暴力遍历是 o（n^2）），后来看了同学们的代码，方法都很巧妙，蛮有收获的~尤其是那个马拉车算法，感觉以我的脑回路不太容易想到，也算是开辟了一个新的脑回路吧哈哈哈

第五题我一开始用的是 dfs 的思路，想用 status = [[0]*n for _ in range(m)]来记录每个点的淹没状态，后来才发现遗漏了一个重要条件：每个被淹没的格子的水的高度！意识到这一点之后，我就发现貌似只能用 bfs 来做了。。知道了要用 bfs 来做之后就不难了，思路很容易想到；但我还在一直 wa，很痛苦，尤其是某个同学 ac 的代码和我的思路几乎一模一样，可是我就是没法 ac；经过整整 57 次的与 ac 代码的反复比对与修改，我才终于把自己的代码 ac。。（花了几乎整整一天时间）

随着最后通牒的道出，C国的总攻也开始了，由于C国在地形上的优势，C国总司令下令采用水攻，剿灭A国最后的有生力量。
地形图是一个从N*N的矩阵，矩阵上每一个点都对应着当前点的高度。C国总司令将选择若干个点进行放水。根据水往低处流的特性，水可以往四个方向的流动，被淹的地方的水面高度便和放水点的高度一样。然而，A国不是一马平川的，所以总会有地方是淹没不到的。你的任务很简单，判断一下A国司令部会不会被淹没掉。
我们将给你完整的地形图，然后给出A国司令部所在位置，给出C国将在哪几个点进行放水操作。你所需要的，就是给出A国司令部会不会被水淹。

输入

第一行：一个整数K，代表数据组数。
对于每一组数据：
第1行：符合题目描述的两个整数，M(0 < M <= 200)，N(0 < N <= 200)。
第2行至M+1行：每行N个数，以空格分开，代表这个矩阵上的各点的高度值H(0 <= H <= 1000)。
第M+2行：两个整数I(0 < I <= M)，J(0 < J <= N)，代表司令部所在位置。
第M+3行：一个整数P(0 < P <= M * N)，代表放水点个数。
第M+4行至M+P+4行：每行两个整数X(0 < X <= M)，Y(0 < Y <= N)，代表放水点。

输出

对于每组数据，输出一行，如果被淹则输出Yes，没有则输出No。

样例输入

```
1
5 5
1 1 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 1 1
3 3
2
1 1
2 2
```

样例输出

```
No
```

提示

样例中左上角的位置是(1, 1),右上角的位置是(1, 5), 右下角的位置是(5, 5)

查看 提交 统计 提问

尝试人数 156
通过人数 138

你的提交记录

#	结果	时间
57	Accepted	2024-12-03
56	Wrong Answer	2024-12-03
55	Wrong Answer	2024-12-03
54	Accepted	2024-12-03
53	Wrong Answer	2024-12-03
52	Accepted	2024-12-03
51	Wrong Answer	2024-12-03
50	Accepted	2024-12-03
49	Runtime Error	2024-12-03
48	Wrong Answer	2024-12-03
47	Wrong Answer	2024-12-03
46	Wrong Answer	2024-12-03
45	Wrong Answer	2024-12-03
44	Accepted	2024-12-03
43	Accepted	2024-12-03
42	Accepted	2024-12-03
41	Wrong Answer	2024-12-03
40	Wrong Answer	2024-12-03
39	Accepted	2024-12-03
38	Runtime Error	2024-12-03
37	Compile Error	2024-12-03
36	Accepted	2024-12-03
35	Wrong Answer	2024-12-03
34	Wrong Answer	2024-12-03
33	Runtime Error	2024-12-03
32	Compile Error	2024-12-03
31	Wrong Answer	2024-12-03
30	Wrong Answer	2024-12-03
29	Runtime Error	2024-12-03
28	Compile Error	2024-12-03
27	Compile Error	2024-12-03
26	Wrong Answer	2024-12-03
25	Wrong Answer	2024-12-03
24	Accepted	2024-12-03
23	Accepted	2024-12-03
22	Runtime Error	2024-12-03
21	Wrong Answer	2024-12-03
20	Accepted	2024-12-03
19	Wrong Answer	2024-12-03
18	Runtime Error	2024-12-03
17	Runtime Error	2024-12-03
16	Runtime Error	2024-12-03
15	Compile Error	2024-12-03
14	Wrong Answer	2024-12-03
13	Wrong Answer	2024-12-03
12	Wrong Answer	2024-12-03
11	Wrong Answer	2024-12-03
10	Wrong Answer	2024-12-03
9	Wrong Answer	2024-12-03
8	Wrong Answer	2024-12-03
7	Wrong Answer	2024-12-03
6	Wrong Answer	2024-12-03
5	Compile Error	2024-12-03
4	Wrong Answer	2024-12-03
3	Wrong Answer	2024-12-03
2	Runtime Error	2024-12-03
1	Runtime Error	2024-12-03

而且，我总感觉这道题目的测试数据有点问题，因为我在比对我的代码和 ac 代码的过程中发现，有好几次都是我的代码与 ac 代码只有一丁点点差别（而且这个差别完全无关紧要），却一个 wa 一个 ac？！比如我直到现在都不知道为什么下面两个代码一个 wa 一个 ac：

#47539527提交状态

查看 提交 统计 提问

状态: Wrong Answer

源代码

```
from collections import deque

def bfs(x, y):
    global status
    status[x][y] = height[x][y]#标记被淹的格子的水面高度
    queue = deque([(x,y)])
    directions = [(-1,0),(1,0),(0,-1),(0,1)]

    while queue:
        x, y = queue.popleft()#对当前deque最靠前的点的水流分析
        for dir in directions:
            nx, ny = x + dir[0], y + dir[1]
            if nx in range(m) and ny in range(n) and status[nx][ny] == h
                # (如果这个邻居还未被淹且海拔低于这个点的水流，则把它淹了)
                    status[nx][ny] = status[x][y]
                    queue.append((nx,ny))
            elif nx in range(m) and ny in range(n) and status[nx][ny] !=
                # (如果这个邻居已经淹了但是当前水面的高度比这个点的水流低，则更新
                    status[nx][ny] = status[x][y]
                    queue.append((nx,ny))

import sys
lines=list(sys.stdin.read().split())#一次性读取所有数据，并按空格划分开，合并为

k = int(lines[0])
idx = 1#目前检索到的行数
for i in range(k):
    height = []#记录高度，不可更改
    m, n = map(int,lines[idx:idx + 2])
    idx += 2
    for j in range(m):
        data = list(map(int,lines[idx:idx + n]))
        height.append(data)
        idx += n
    status = height[:]#复制一遍高度列表，可随时更改，视为当下的"表面高度"
    # (若"表面高度"等于原高度，则说明这是陆地高度，未被淹；反之则说明是水面高度，已经

    data = list(map(int,lines[idx:idx + 2]))
    idx += 2
    a, b = data[0] - 1, data[1] - 1#记录司令部位置
    p = int(lines[idx])
    idx += 1
    for j in range(p):
        data = list(map(int,lines[idx:idx + 2]))
        idx += 2
        x, y = data[0] - 1, data[1] - 1
        bfs(x, y)

    if height[a][b] < status[a][b]:
        print("Yes")
    else:print("No")
```

基本信息

#:	47539527
题目:	12029
提交人:	24n2300010821
内存:	6424kB
时间:	616ms
语言:	Python3
提交时间:	2024-12-03 19:59:39

English 帮助 关于

#47539499提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 47539499
题目: 12029
提交人: 24n2300010821
内存: 7556kB
时间: 616ms
语言: Python3
提交时间: 2024-12-03 19:58:33

源代码

```
from collections import deque

def bfs(x, y):
    global status
    status[x][y] = height[x][y]#标记被淹的格子的水面高度
    queue = deque([(x,y)])
    directions = [(-1,0),(1,0),(0,-1),(0,1)]

    while queue:
        x, y = queue.popleft()#对当前deque最靠前的点的水流分析
        for dir in directions:
            nx, ny = x + dir[0], y + dir[1]
            if nx in range(m) and ny in range(n) and status[nx][ny] == h
                # (如果这个邻居还未被淹且海拔低于这个点的水流，则把它淹了)
                    status[nx][ny] = status[x][y]
                    queue.append((nx,ny))
            elif nx in range(m) and ny in range(n) and status[nx][ny] !=
                # (如果这个邻居已经淹了但是当前水面的高度比这个点的水流低，则更新
                    status[nx][ny] = status[x][y]
                    queue.append((nx,ny))

import sys
lines=list(sys.stdin.read().split())#一次性读取所有数据，并按空格划分开，合并为

k = int(lines[0])
idx = 1#目前检索到的行数
for i in range(k):
    height = []#记录高度，不可更改
    m, n = map(int,lines[idx:idx + 2])
    idx += 2
    for j in range(m):
        data = list(map(int,lines[idx:idx + n]))
        height.append(data)
        idx += n
    status = height[:]#复制一遍高度列表，可随时更改，视为当下的"表面高度"
    # (若"表面高度"等于原高度，则说明这是陆地高度，未被淹；反之则说明是水面高度，已经

    data = list(map(int,lines[idx:idx + 2]))
    idx += 2
    a, b = data[0] - 1, data[1] - 1#记录司令部位置
    t = height[a][b]
    p = int(lines[idx])
    idx += 1
    for j in range(p):
        data = list(map(int,lines[idx:idx + 2]))
        idx += 2
        x, y = data[0] - 1, data[1] - 1
        bfs(x, y)

    if t < status[a][b]:
        print("Yes")
    else:print("No")
```


（这两个代码几乎完全一模一样，唯一的一个区别是：下面的代码把 t 记为了 height[a][b]，并在最后一步用 t 与 status[a][b]作对比；
而上面的代码在最后一步直接用 height[a][b]与 status[a][b]作对比；
但问题在于 height 列表根本没有改变过啊，不应该有任何关系啊）

第六题实在太难了，看了题解才能勉强理解。。感觉 bfs 掌握的还不是很熟练，还是得再消化一下，课后得再多练习几道类似的题目才行。。