**Assignment #10: dp & bfs**

2024 fall, Complied by 吕金浩，物理学院

**1. 题目**

**LuoguP1255 数楼梯**

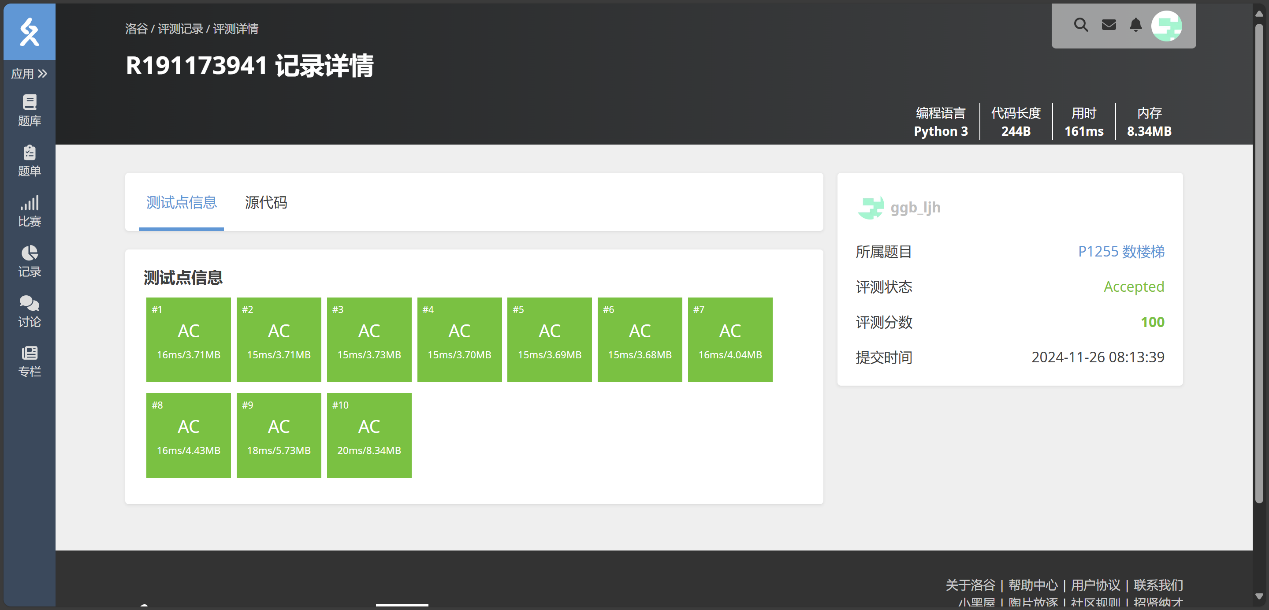
dp, bfs, <https://www.luogu.com.cn/problem/P1255>

思路：递归秒了……

代码：

import sys  
sys.setrecursionlimit(1<<30)  
from functools import lru\_cache  
@lru\_cache(maxsize=None)  
def stairs(n):  
 if n==1:  
 return 1  
 if n==2:  
 return 2  
 return stairs(n-1)+stairs(n-2)  
print(stairs(int(input())))

​



**27528: 跳台阶**

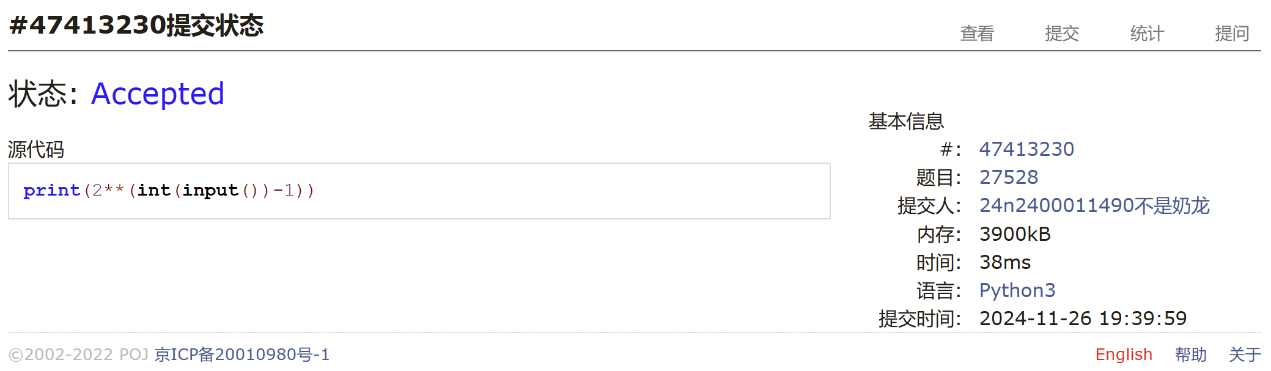
dp, <http://cs101.openjudge.cn/practice/27528/>

思路：What can I say?

代码：

print(2\*\*(int(input())-1))

​



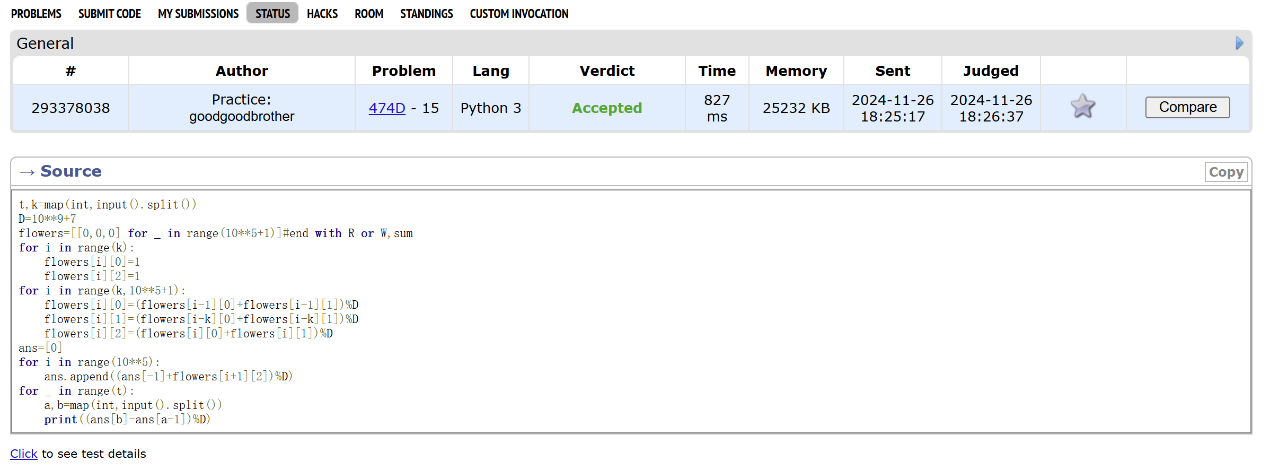
**474D. Flowers**

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

思路：一开始想的递推是首尾两端都可以加东西，想了差不多一个小时，然后突然意识到可以直接在尾端加1个R或k个W，就很快写出来了

代码：

t,k=map(int,input().split())  
D=10\*\*9+7  
flowers=[[0,0,0] for \_ in range(10\*\*5+1)]#end with R or W,sum  
for i in range(k):  
 flowers[i][0]=1  
 flowers[i][2]=1  
for i in range(k,10\*\*5+1):  
 flowers[i][0]=(flowers[i-1][0]+flowers[i-1][1])%D  
 flowers[i][1]=(flowers[i-k][0]+flowers[i-k][1])%D  
 flowers[i][2]=(flowers[i][0]+flowers[i][1])%D  
ans=[0]  
for i in range(10\*\*5):  
 ans.append((ans[-1]+flowers[i+1][2])%D)  
for \_ in range(t):  
 a,b=map(int,input().split())  
 print((ans[b]-ans[a-1])%D)



**LeetCode5.最长回文子串**

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

解法1

思路：找递推关系，判断每条字串是不是回文的

代码：

class Solution:

    def longestPalindrome(self, s: str) -> str:

        #def longestpali(s):

        lens=len(s)

        pali=[[False for \_ in range(lens)] for \_ in range(lens)]

        maxij=(0,0)

        for j in range(lens):

            for i in range(j,-1,-1):

                if j==i:

                    pali[i][j]=True

                elif j==i+1 and s[j]==s[i]:

                    pali[i][j]=True

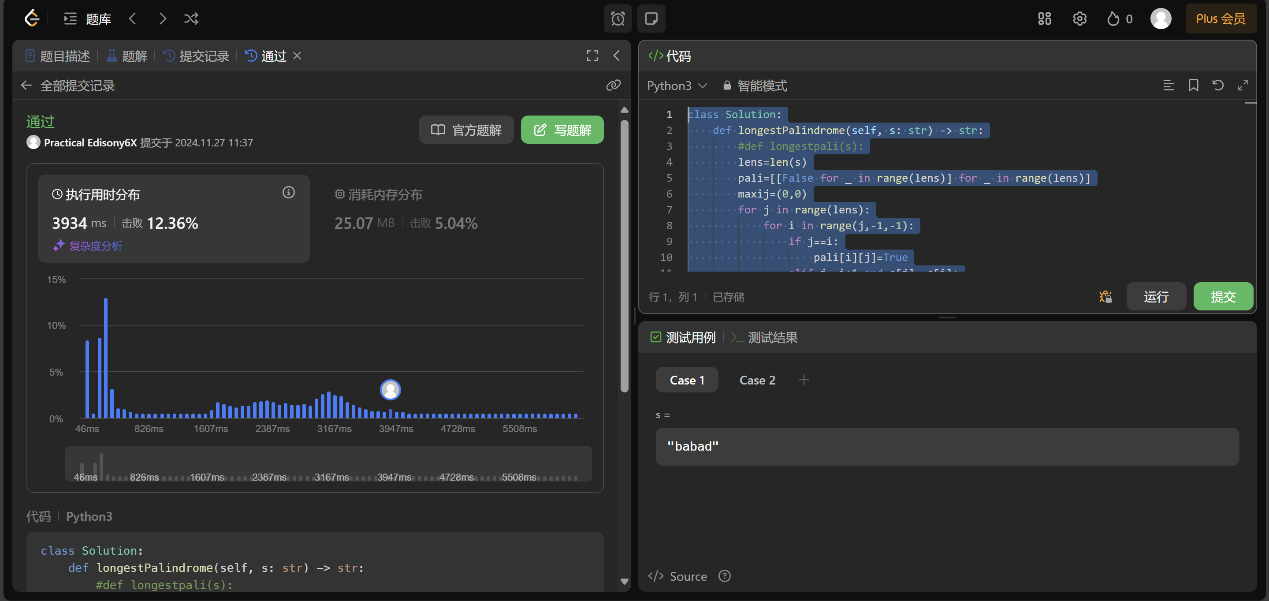
                else:

                    pali[i][j]=s[j]==s[i] and pali[i+1][j-1]

                if pali[i][j] and j-i>maxij[1]-maxij[0]:

                    maxij=(i,j)

        return s[maxij[0]:maxij[1]+1]



解法2

思路：看了答案之后，写出的我自己看得懂、写得出的双指针

代码：

class Solution:

    def longestPalindrome(self, s: str) -> str:

        #def longestpali(s):

        lens=len(s)

        ans=(0,0)

        def longest\_expand(x,y):

            while 0<=x<=y<lens and s[x]==s[y]:

                x-=1

                y+=1

            return x+1,y-1

        for i in range(lens):

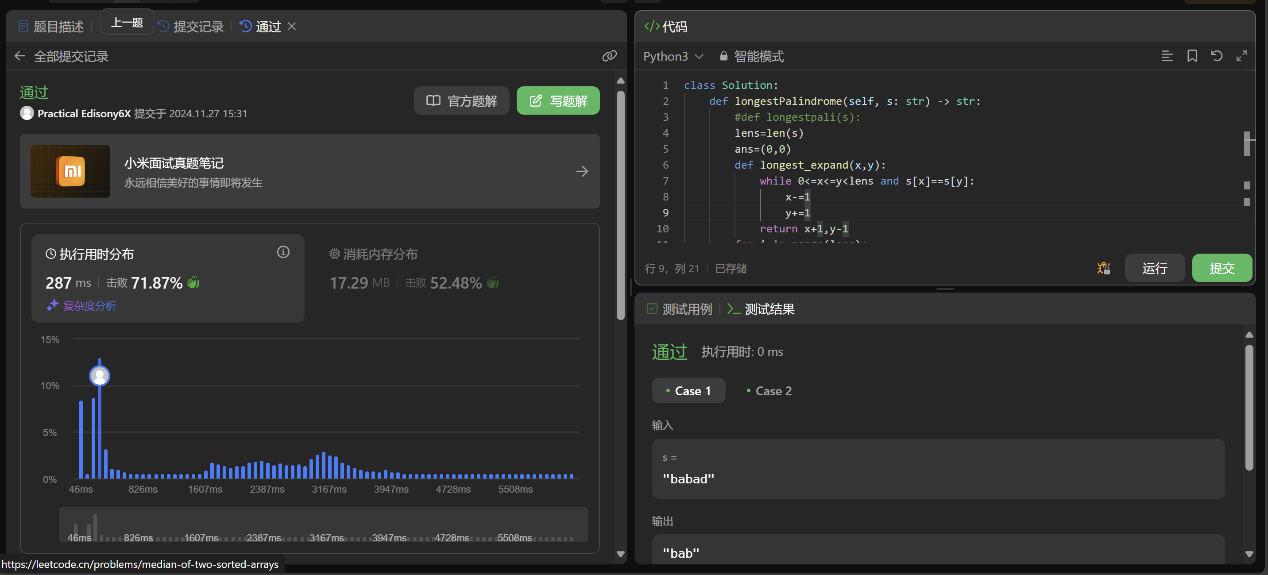
            for j in range(2):

                a = longest\_expand(i, i+j)

                if a[1] - a[0] > ans[1] - ans[0]:

                    ans = a

        return s[ans[0]:ans[1]+1]



**12029: 水淹七军**

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

思路：RE了二十次，最后照着答案和老师的回答做出来的，另外用visited会TLE。

代码：

import sys  
sys.setrecursionlimit(1<<30)  
dx=[0,0,1,-1]  
dy=[1,-1,0,0]  
  
def dfs(x,y,h):  
 waterheight[x][y]=h  
  
 for i in range(4):  
 nx,ny=x+dx[i],y+dy[i]  
 if 0<=nx<m and 0<=ny<n:  
 if matrix[nx][ny]<h:  
 if waterheight[nx][ny]<h:  
 dfs(nx,ny,h)  
  
data=sys.stdin.read().split()  
k=int(data[0])  
idx=1  
  
for \_ in range(k):  
  
 m,n=map(int,data[idx:idx+2])  
 idx+=2  
 matrix=[]  
 for \_ in range(m):  
 matrix.append([int(x) for x in data[idx:idx+n]])  
 idx+=n  
 a,b=map(int,data[idx:idx+2])  
 idx+=2  
  
 tarx,tary=a-1,b-1  
 tarh=matrix[tarx][tary]  
 p=int(data[idx])  
 idx+=1  
 waters=[]  
 for \_ in range(p):  
 a,b=map(int,data[idx:idx+2])  
 idx+=2  
 waters.append((a-1,b-1,matrix[a-1][b-1]))  
  
 waterheight=[[0 for \_ in range(n)] for \_ in range(m) ]  
  
 for water in waters:  
 wx,wy,wh=water  
  
 if wh>tarh:  
 dfs(wx,wy,wh)  
 print('Yes' if waterheight[tarx][tary]>0 else 'No')



**02802: 小游戏**

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

思路：

比上一题舒服多了……这一题就只presentation error了一次，发现是输出空行搞错了之后，改了一下就AC了。

先将目标卡片设为空位，对线段数作bfs。对每一个k段线段能找到的空位，令其向四方向延展，找到k+1段线段能找到的空位。注意要适当允许重复现象，否则可能会导致路被堵死。

代码：

from collections import deque  
  
def bfs(sx,sy,tx,ty):  
 q=deque()  
 q.append((sx,sy,0))#x,y,step  
 while q:  
 x,y,step=q.popleft()  
 if x==tx and y==ty:  
 return step  
   
 for i in range(x+1,m+2):  
 if matrix[i][y]==' ' and ((i,y) not in inq or min\_steps[i][y]==step+1):#适当允许重复  
 min\_steps[i][y]=step+1  
 inq.add((i,y))  
 q.append((i,y,step+1))  
 else:  
 break  
 for i in range(x-1,-1,-1):  
 if matrix[i][y]==' ' and ((i,y) not in inq or min\_steps[i][y]==step+1):  
 min\_steps[i][y]=step+1  
 inq.add((i,y))  
 q.append((i,y,step+1))  
 else:  
 break  
 for i in range(y+1,n+2):  
 if matrix[x][i]==' ' and ((x,i) not in inq or min\_steps[x][i]==step+1):  
 min\_steps[x][i]=step+1  
 inq.add((x,i))  
 q.append((x,i,step+1))  
 else:  
 break  
 for i in range(y-1,-1,-1):  
 if matrix[x][i]==' ' and ((x,i) not in inq or min\_steps[x][i]==step+1):  
 min\_steps[x][i]=step+1  
 inq.add((x,i))  
 q.append((x,i,step+1))  
 else:  
 break  
  
 return -1  
  
board=0  
while True:  
 n,m=map(int,input().split())  
 if m==0 and n==0:  
 break  
 board+=1  
 if board>1:  
 print('')  
 print('Board #{}:'.format(board))  
 matrix=[[' ']\*(n+2)]+[list(' '+input()+' ') for \_ in range(m) ]+[[' ']\*(n+2)]  
   
 pair=0  
 while True:  
 y1,x1,y2,x2=map(int,input().split())  
 if y1==0 and x1==0 and y2==0 and x2==0:  
 break  
 pair+=1  
 inq={(x1,y1)}  
 min\_steps=[[float('inf') for \_ in range(n+2)] for \_ in range(m+2)]  
 min\_steps[x1][y1]=0  
 matrix[x1][y1]=' '  
 matrix[x2][y2]=' '  
 segment=bfs(x1,y1,x2,y2)  
 print('Pair {}: {} segments.'.format(pair,segment) if segment>0 else 'Pair {}: impossible.'.format(pair))  
 matrix[x1][y1] = 'X'  
 matrix[x2][y2] = 'X'



**2. 学习总结和收获**

我现在做dp主打一个递归，对于大部分遇到的dp题还是比较有用的，简化了对于边界条件的考虑。另外，感受到bfs和dfs的繁杂了，希望老师期末手下留情（