**Assignment #9: dfs, bfs, & dp**

2024 fall, Complied by 吕金浩，物理学院

**1. 题目**

**18160: 最大连通域面积**

dfs similar, <http://cs101.openjudge.cn/practice/18160>

代码：

dx=[1,-1,0,0,1,1,-1,-1]

dy=[0,0,1,-1,1,-1,1,-1]

ans=0

def dfs(x,y):

global ans

ans+=1

chessboard[x][y]='.'

for k in range(8):

newx=x+dx[k]

newy=y+dy[k]

if 0<=newx<n and 0<=newy<m and chessboard[newx][newy]=='W':

dfs(newx,newy)

#return res

for \_ in range(int(input())):

n,m=map(int,input().split())

chessboard=[]

for i in range(n):

chessboard.append(list(input()))

res=0

#print(chessboard)

for i in range(n):

for j in range(m):

if chessboard[i][j]=='W':

ans=0

dfs(i,j)

res=max(res,ans)

#ans=max(ans,dfs(i,j))

print(res)



**19930: 寻宝**

bfs, <http://cs101.openjudge.cn/practice/19930>

代码：

from collections import deque

dx=[1,-1,0,0]

dy=[0,0,1,-1]

def can\_visit(x,y):

return -1<x<m and -1<y<n and maze[x][y]!=2 and not inq[x][y]

m,n=map(int,input().split())

maze=[]

for \_ in range(m):

maze.append([int(x) for x in input().split()])

q=deque()

q.append((0,(0,0)))

inq=[[False]\*n for \_ in range(m)]

inq[0][0]=True

while q:

step,(curx,cury)=q.popleft()

if maze[curx][cury]==1:

print(step)

break

for k in range(4):

nextx,nexty=curx+dx[k],cury+dy[k]

if can\_visit(nextx,nexty):

inq[nextx][nexty]=True

q.append((step+1,(nextx,nexty)))

else:

print('NO')



**04123: 马走日**

dfs, <http://cs101.openjudge.cn/practice/04123>

思路：

代码：

def can\_visit(x,y):

return -1<x<n and -1<y<m and not visited[x][y]

dx=[2,2,1,1,-1,-1,-2,-2]

dy=[1,-1,2,-2,2,-2,1,-1]

ans=0

def dfs(cnt,x,y):

global ans

if cnt==m\*n:

ans+=1

return

visited[x][y]=True

for k in range(8):

nx,ny=x+dx[k],y+dy[k]

if can\_visit(nx,ny):

dfs(cnt+1,nx,ny)

visited[x][y]=False

for \_ in range(int(input())):

n,m,a,b=map(int,input().split())

visited=[[False]\*m for \_ in range(n)]

ans=0

dfs(1,a,b)

print(ans)



**sy316: 矩阵最大权值路径**

dfs, <https://sunnywhy.com/sfbj/8/1/316>

思路：把所有路径存起来，然后排序（应该比较费时）

代码：

n,m=map(int,input().split())  
matrix=[]  
for \_ in range(n):  
 matrix.append([int(x) for x in input().split()])  
visited=[[False]\*m for \_ in range(n)]  
valid\_path=[]  
  
def can\_visit(x,y):  
 return 0<=x<n and 0<=y<m and not visited[x][y]  
dx=[1,-1,0,0]  
dy=[0,0,1,-1]  
path=[[(0,0)],matrix[0][0]]  
  
def dfs(x,y):  
 if x==n-1 and y==m-1:  
 valid\_path.append([path[0][:],path[1]])  
 return  
 visited[x][y]=True  
 for k in range(4):  
 nx=x+dx[k]  
 ny=y+dy[k]  
 if can\_visit(nx,ny):  
 #visited[nx][ny]=True  
 path[0].append((nx,ny))  
 path[1]+=matrix[nx][ny]  
 #print(path)  
 dfs(nx,ny)  
 path[0].pop()  
 path[1]-=matrix[nx][ny]  
 visited[x][y]=False  
dfs(0,0)  
  
#print(valid\_path)  
  
valid\_path.sort(reverse=True ,key=lambda x: x[-1])  
a=valid\_path[0][0][:]  
for x,y in a:  
 print(str(x+1)+' '+str(y+1))



**LeetCode62.不同路径**

dp, <https://leetcode.cn/problems/unique-paths/>

思路：有公式(m+n-2)!/(m-1)!(n-1)!，python支持大数字运算挺好

代码：

class Solution:

    def uniquePaths(self, m: int, n: int) -> int:

        def factorial(x):

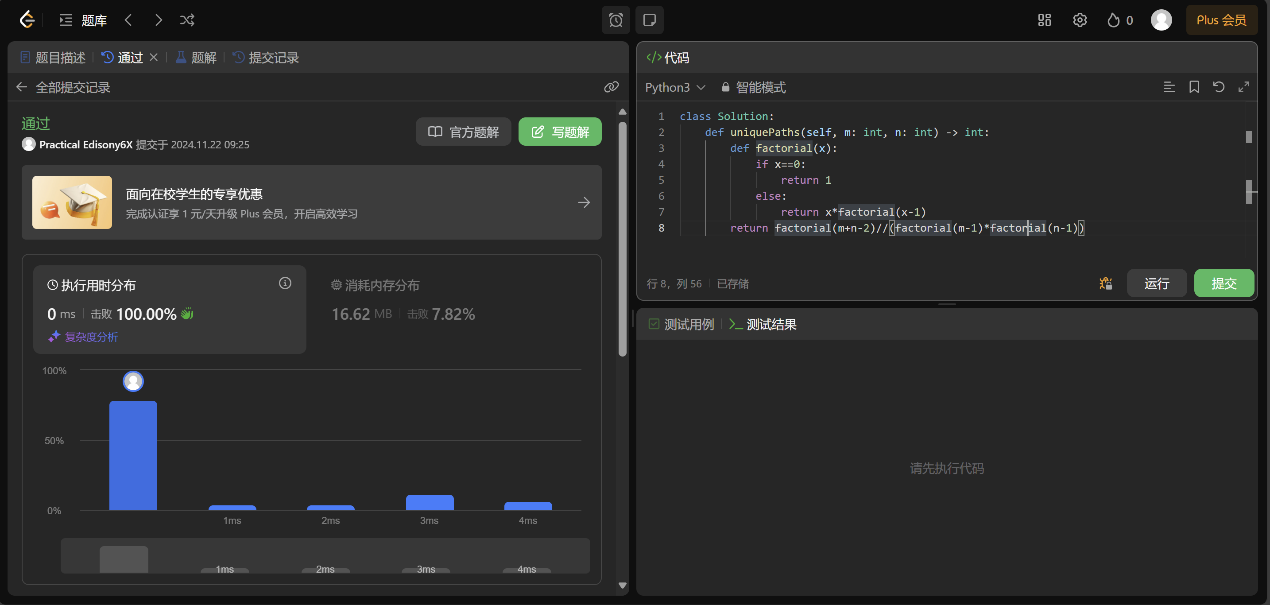
            if x==0:

                return 1

            else:

                return x\*factorial(x-1)

        return factorial(m+n-2)//(factorial(m-1)\*factorial(n-1))



**sy358: 受到祝福的平方**

dfs, dp, <https://sunnywhy.com/sfbj/8/3/539>

思路：递归，为了判断一个数是不是祝福数，如果存在一种二分分割，其前一半是正平方数，且后一半是祝福数，则它是祝福数。

代码：

from math import \*  
from functools import lru\_cache  
  
def square\_int(x):  
 return int(sqrt(x))\*\*2==x and x!=0  
  
@lru\_cache(maxsize=None)  
def if\_square(x):  
 #ans=square\_int(int(x))  
 if square\_int(int(x)):  
 return True  
  
 for i in range(1,len(x)):  
 a=x[:i]  
 b=x[i:]  
 #ans=ans or (square\_int(int(a)) and if\_square(b))  
 if square\_int(int(a)) and if\_square(b):  
 return True  
 return False  
print('Yes' if if\_square(input()) else 'No')



**2. 学习总结和收获**

刚开始跟着老师学bfs，虽然dfs和bfs题目做起来不能很快（平均一道大概写个二三十来分钟），但目前的dfs和bfs题目似乎都比较模板化，做起来比较顺利。每日选做也正在持续跟进。