Assignment #9: dfs, bfs, & dp

2024 fall, Complied by 吕金浩，物理学院

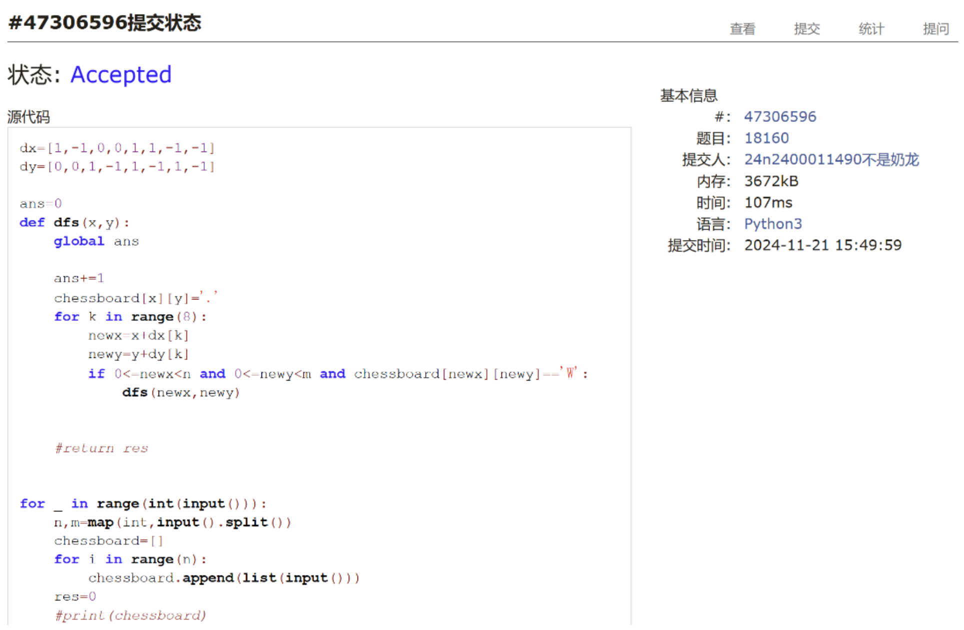
1. 题目

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)
```


#47306596提交状态

状态: Accepted

源代码

```
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)
```

基本信息

#:	47306596
题目:	18160
提交人:	24n2400011490不是奶龙
内存:	3672kB
时间:	107ms
语言:	Python3
提交时间:	2024-11-21 15:49:59

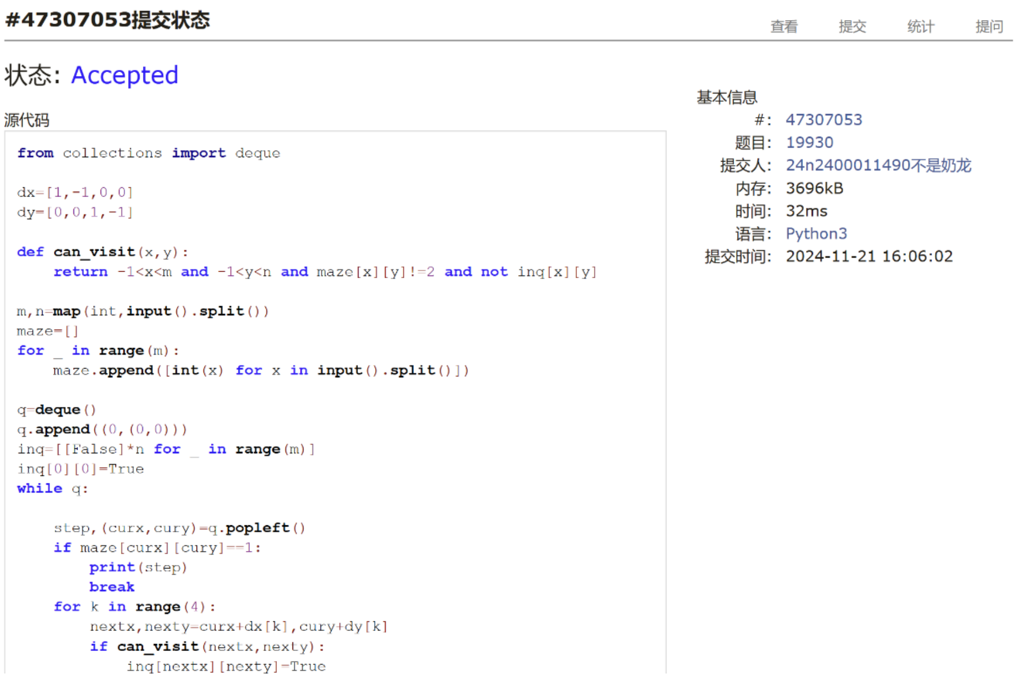
19930: 寻宝

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')
```



#47307053提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
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
```

基本信息

#:	47307053
题目:	19930
提交人:	24n2400011490不是奶龙
内存:	3696kB
时间:	32ms
语言:	Python3
提交时间:	2024-11-21 16:06:02

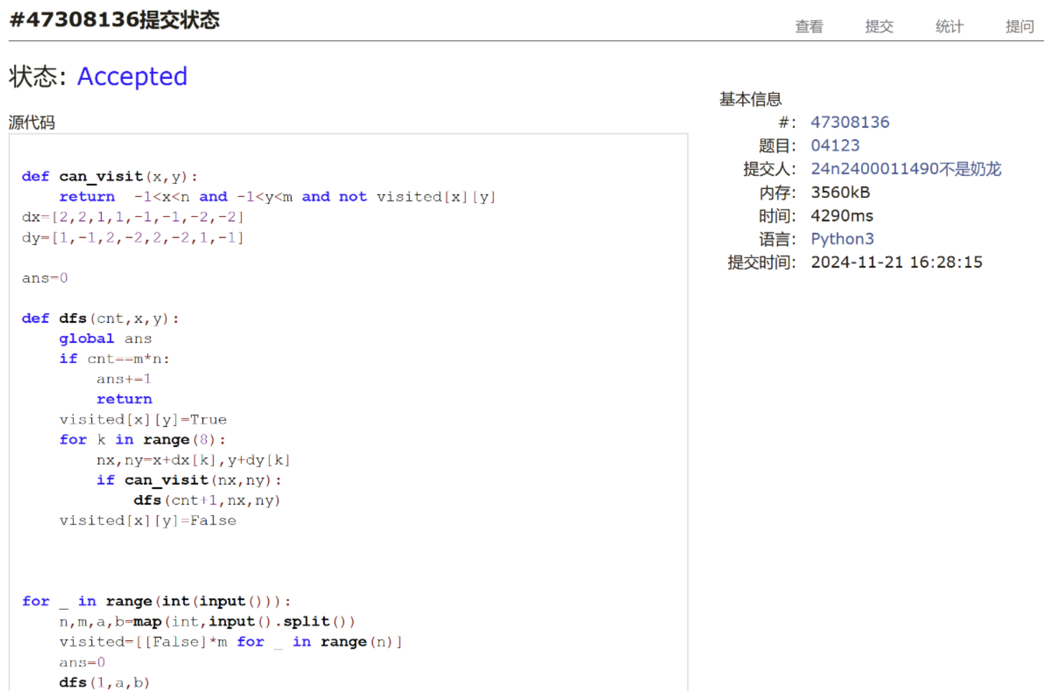
04123: 马走日

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)
```



#47308136提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
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)
```

基本信息

#:	47308136
题目:	04123
提交人:	24n2400011490不是奶龙
内存:	3560kB
时间:	4290ms
语言:	Python3
提交时间:	2024-11-21 16:28:15

sy316: 矩阵最大权值路径

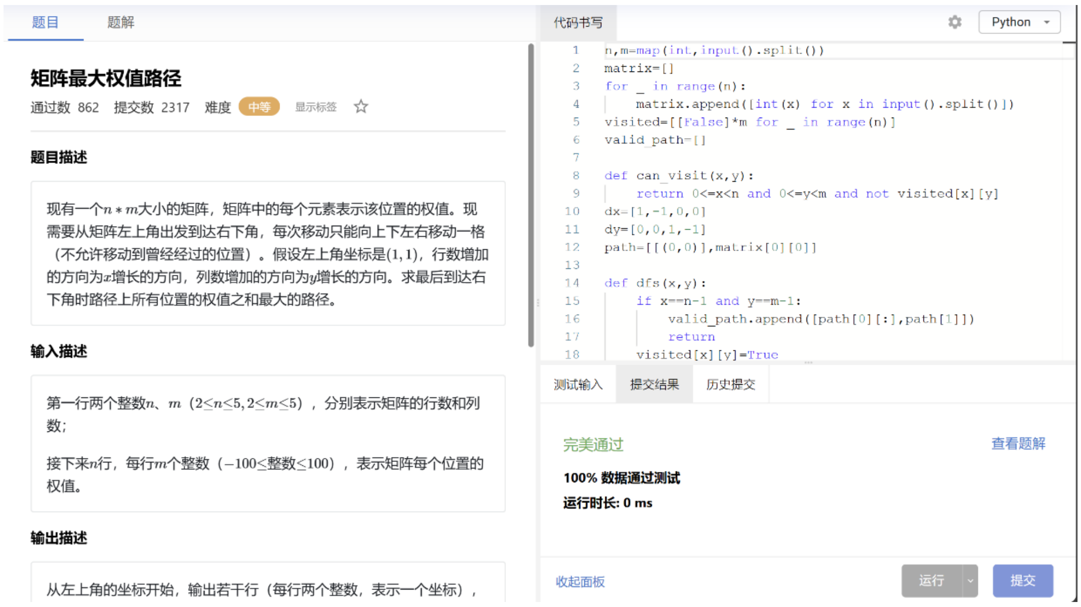
dfs, https://sunnywhy.com/sfbj/8/1/316

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))
```



矩阵最大权值路径

通过数 862　提交数 2317　难度 中等　显示标签

题目描述

现有一个$n*m$大小的矩阵，矩阵中的每个元素表示该位置的权值。现需要从矩阵左上角出发到达右下角，每次移动只能向上下左右移动一格（不允许移动到曾经经过的位置）。假设左上角坐标是$(1,1)$，行数增加的方向为x增长的方向，列数增加的方向为y增长的方向。求最后到达右下角时路径上所有位置的权值之和最大的路径。

输入描述

第一行两个整数n、m（$2 \le n \le 5, 2 \le m \le 5$），分别表示矩阵的行数和列数;

接下来n行，每行m个整数（$-100 \le$整数≤ 100），表示矩阵每个位置的权值。

输出描述

从左上角的坐标开始，输出若干行（每行两个整数，表示一个坐标），

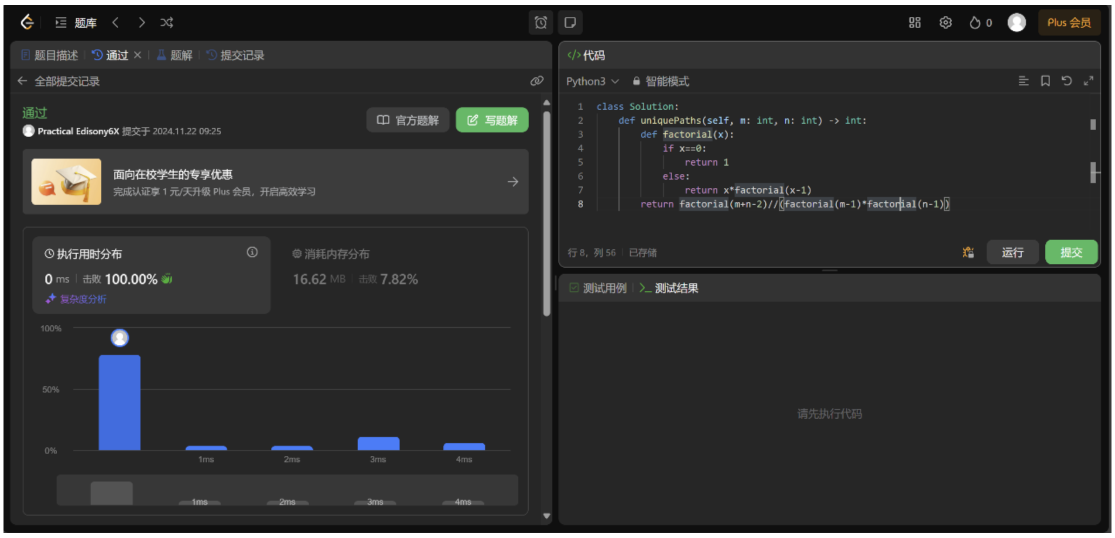
```
n,m=map(int,input().split())
matrix=[]
for _ in range(n):
    matrix.append([int(x) for x in input().split()])
visited=[[False]*m for _ in range(n)]
valid_path=[]

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
```

完美通过

100% 数据通过测试

运行时长: 0 ms

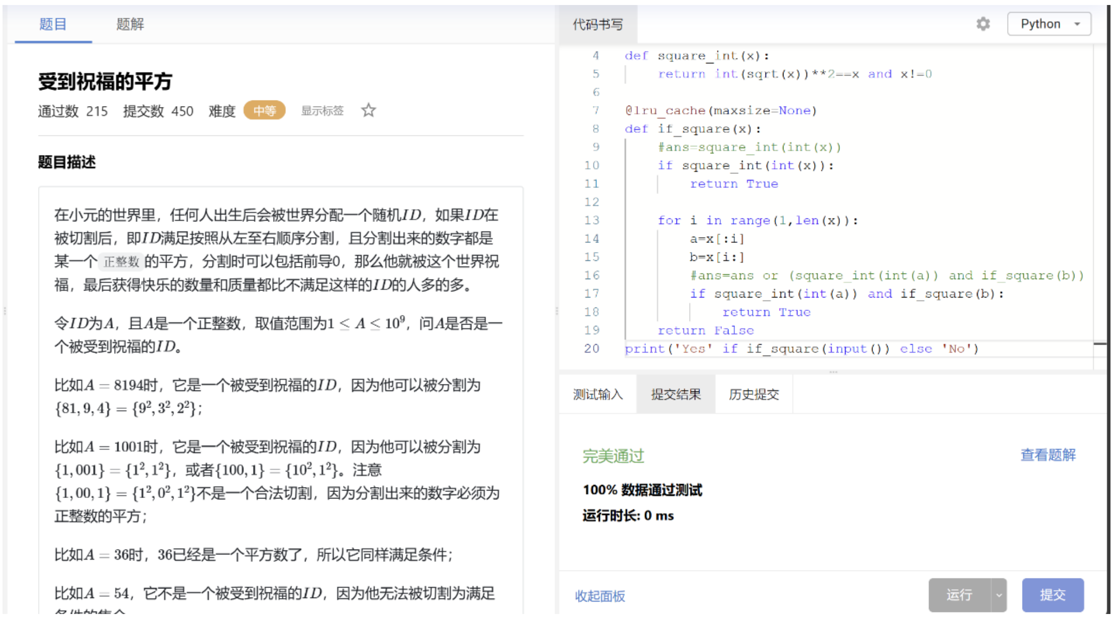
LeetCode62.不同路径
dp, https://leetcode.cn/problems/unique-paths/
思路：有公式(m+n-2)!/(m-1)!(n-1)!，python 支持大数字运算挺好
代码：
```
class Solution:
    def uniquePaths(self, m: int, n: int) -> int:
        def factorial(x):
            if x==0:
                return 1
            else:
                return x*factorial(x-1)
        return factorial(m+n-2)//(factorial(m-1)*factorial(n-1))
```



通过
Practical Edisony6X 提交于 2024.11.22 09:25
执行用时分布
0 ms | 击败 100.00%
消耗内存分布
16.62 MB | 击败 7.82%

```
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))
```
行 8，列 56 | 已存储

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')
```



受到祝福的平方

通过数 215　提交数 450　难度 中等　显示标签

题目描述

在小元的世界里，任何人出生后会被世界分配一个随机ID，如果ID在被切割后，即ID满足按照从左至右顺序分割，且分割出来的数字都是某一个 正整数 的平方，分割时可以包括前导0，那么他就被这个世界祝福，最后获得快乐的数量和质量都比不满足这样的ID的人多的多。

令ID为A，且A是一个正整数，取值范围为$1 \leq A \leq 10^9$，问A是否是一个被受到祝福的ID。

比如$A = 8194$时，它是一个被受到祝福的ID，因为他可以被分割为$\{81, 9, 4\} = \{9^2, 3^2, 2^2\}$；

比如$A = 1001$时，它是一个被受到祝福的ID，因为他可以被分割为$\{1, 001\} = \{1^2, 1^2\}$，或者$\{100, 1\} = \{10^2, 1^2\}$。注意$\{1, 00, 1\} = \{1^2, 0^2, 1^2\}$不是一个合法切割，因为分割出来的数字必须为正整数的平方；

比如$A = 36$时，36已经是一个平方数了，所以它同样满足条件；

比如$A = 54$，它不是一个被受到祝福的ID，因为他无法被切割为满足

```
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')
```

完美通过　　查看题解

100% 数据通过测试

运行时长: 0 ms

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。