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


Updated 2107 GMT+8 Nov 19, 2024
```

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

```
**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

### 18160：最大连通域面积

dfs similar,

http://cs101.openjudge.cn/practice/18160

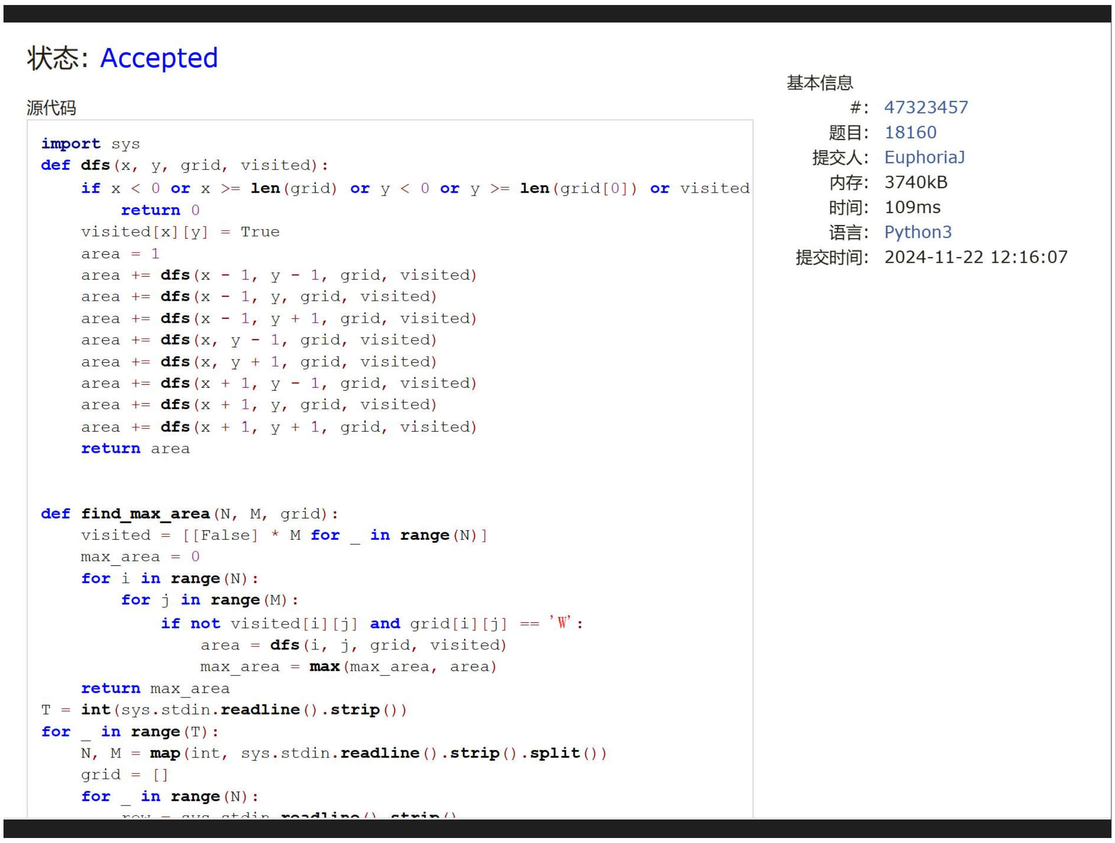
思路：先用 dfs 函数找出所有的连通区域（遍历周围的八个空间）（关键：第一个 area 是固定了坐标（x,y）然后对 x,y 周围的连通区域向八个方向进行探索，而
第二个area的含义是对于矩阵中的任意一个点来向外探索），用 dfs 找出连通区域中的最大值。(1h)

代码：

```
import sys
def dfs(x, y, grid, visited):
    if x < 0 or x >= len(grid) or y < 0 or y >= len(grid[0]) or visited[x][y] or grid[x][y] == '.':
        return 0
    visited[x][y] = True
    area = 1
    area += dfs(x - 1, y - 1, grid, visited)
    area += dfs(x - 1, y, grid, visited)
    area += dfs(x - 1, y + 1, grid, visited)
    area += dfs(x, y - 1, grid, visited)
    area += dfs(x, y + 1, grid, visited)
    area += dfs(x + 1, y - 1, grid, visited)
    area += dfs(x + 1, y, grid, visited)
    area += dfs(x + 1, y + 1, grid, visited)
    return area
```

```
def find_max_area(N, M, grid):
    visited = [[False] * M for _ in range(N)]
    max_area = 0
    for i in range(N):
        for j in range(M):
            if not visited[i][j] and grid[i][j] ==
'W':
                area = dfs(i, j, grid, visited)
                max_area = max(max_area, area)
    return max_area
T = int(sys.stdin.readline().strip())
for _ in range(T):
    N, M = map(int,
sys.stdin.readline().strip().split())
    grid = []
    for _ in range(N):
        row = sys.stdin.readline().strip()
        grid.append(list(row))
    print(find_max_area(N, M, grid))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>


状态: Accepted

源代码

```python
import sys
def dfs(x, y, grid, visited):
    if x < 0 or x >= len(grid) or y < 0 or y >= len(grid[0]) or visited
        return 0
    visited[x][y] = True
    area = 1
    area += dfs(x - 1, y - 1, grid, visited)
    area += dfs(x - 1, y, grid, visited)
    area += dfs(x - 1, y + 1, grid, visited)
    area += dfs(x, y - 1, grid, visited)
    area += dfs(x, y + 1, grid, visited)
    area += dfs(x + 1, y - 1, grid, visited)
    area += dfs(x + 1, y, grid, visited)
    area += dfs(x + 1, y + 1, grid, visited)
    return area


def find_max_area(N, M, grid):
    visited = [[False] * M for _ in range(N)]
    max_area = 0
    for i in range(N):
        for j in range(M):
            if not visited[i][j] and grid[i][j] == 'W':
                area = dfs(i, j, grid, visited)
                max_area = max(max_area, area)
    return max_area
T = int(sys.stdin.readline().strip())
for _ in range(T):
    N, M = map(int, sys.stdin.readline().strip().split())
    grid = []
    for _ in range(N):
        row = sys.stdin.readline().strip()
```

基本信息

#:	47323457
题目:	18160
提交人:	EuphoriaJ
内存:	3740kB
时间:	109ms
语言:	Python3
提交时间:	2024-11-22 12:16:07

### 19930: 寻宝

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

思路:这道 bfs 感觉跟 dfs 有相似也有不同,首先是 import deque, 然后照样是初始化变量（visited,real_list,start 的点, 找出目标点）, 然后就是 bfs 函数模板:

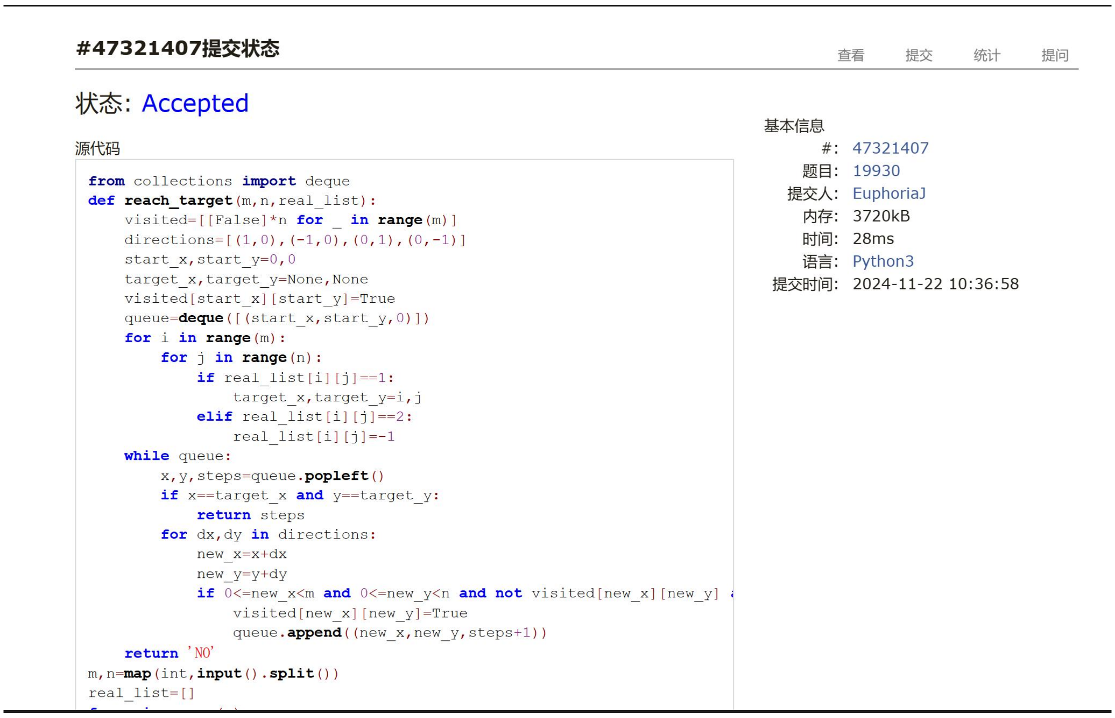
queue=deque(......),while queue,x,y,steps=queue.popleft(),然后再讨论终点情况与中间情况（中间情况与 dfs 有相似之处，也是要把那一点赋值为 True，但后面是 append...）

最后输出即可。（1h）

代码：

```
from collections import deque

def reach_target(m,n,real_list):

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

    directions=[(1,0),(-1,0),(0,1),(0,-1)]

    start_x,start_y=0,0

    target_x,target_y=None,None

    visited[start_x][start_y]=True

    queue=deque([(start_x,start_y,0)])

    for i in range(m):

        for j in range(n):

            if real_list[i][j]==1:

                target_x,target_y=i,j

            elif real_list[i][j]==2:

                real_list[i][j]=-1
    while queue:
        x,y,steps=queue.popleft()
        if x==target_x and y==target_y:
            return steps
        for dx,dy in directions:
            new_x=x+dx
            new_y=y+dy
            if 0<=new_x<m and 0<=new_y<n and not
visited[new_x][new_y] and real_list[new_x][new_y]!=-1:
                visited[new_x][new_y]=True
                queue.append((new_x,new_y,steps+1))
    return 'NO'
m,n=map(int,input().split())
real_list=[]
for r in range(m):
    real_list.append(list(map(int,input().split())))
print(reach_target(m,n,real_list))
```

```
代码运行截图 ==（至少包含有"Accepted"）==
```


#47321407提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from collections import deque
def reach_target(m,n,real_list):
    visited=[[False]*n for _ in range(m)]
    directions=[(1,0),(-1,0),(0,1),(0,-1)]
    start_x,start_y=0,0
    target_x,target_y=None,None
    visited[start_x][start_y]=True
    queue=deque([(start_x,start_y,0)])
    for i in range(m):
        for j in range(n):
            if real_list[i][j]==1:
                target_x,target_y=i,j
            elif real_list[i][j]==2:
                real_list[i][j]=-1
    while queue:
        x,y,steps=queue.popleft()
        if x==target_x and y==target_y:
            return steps
        for dx,dy in directions:
            new_x=x+dx
            new_y=y+dy
            if 0<=new_x<m and 0<=new_y<n and not visited[new_x][new_y]
                visited[new_x][new_y]=True
                queue.append((new_x,new_y,steps+1))
    return 'NO'
m,n=map(int,input().split())
real_list=[]
```

基本信息

#:	47321407
题目:	19930
提交人:	EuphoriaJ
内存:	3720kB
时间:	28ms
语言:	Python3
提交时间:	2024-11-22 10:36:58

### 04123: 马走日

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

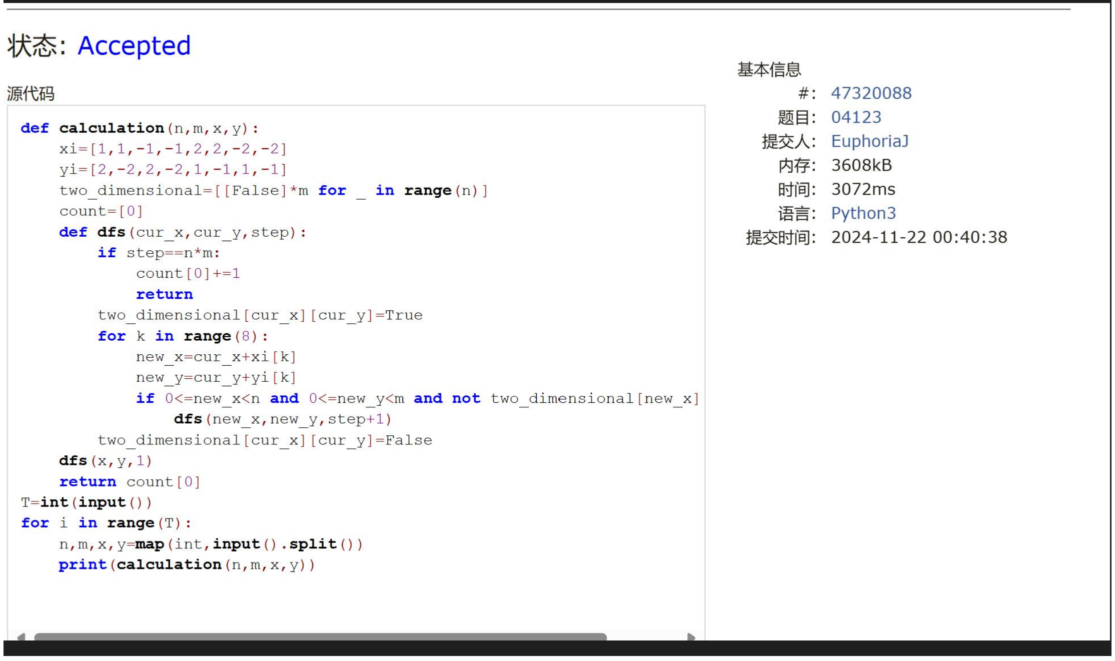
思路：dfs 问题，与下面一个题很类似，也是先初始化变量（方向矩阵，False 列表标记位置，走过步数），再套用 dfs 函数模版，(最终状态，中间状态（注意 if 条件句有三个条件，况且要记得回溯），最后给出 dfs 函数从（x,y,1）开始运行。（1.5h）

代码:

```
def calculation(n,m,x,y):
    xi=[1,1,-1,-1,2,2,-2,-2]
    yi=[2,-2,2,-2,1,-1,1,-1]
    two_dimensional=[[False]*m for _ in range(n)]
    count=[0]
    def dfs(cur_x,cur_y,step):
        if step==n*m:
            count[0]+=1
            return
        two_dimensional[cur_x][cur_y]=True
        for k in range(8):
            new_x=cur_x+xi[k]
            new_y=cur_y+yi[k]
            if 0<=new_x<n and 0<=new_y<m and not
two_dimensional[new_x][new_y]:
                dfs(new_x,new_y,step+1)
        two_dimensional[cur_x][cur_y]=False
    dfs(x,y,1)
    return count[0]
```

```
T=int(input())

for i in range(T):

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

    print(calculation(n,m,x,y))
```

```
代码运行截图 <mark>（至少包含有"Accepted"）</mark>
```



状态: Accepted

源代码

```
def calculation(n,m,x,y):
    xi=[1,1,-1,-1,2,2,-2,-2]
    yi=[2,-2,2,-2,1,-1,1,-1]
    two_dimensional=[[False]*m for _ in range(n)]
    count=[0]
    def dfs(cur_x,cur_y,step):
        if step==n*m:
            count[0]+=1
            return
        two_dimensional[cur_x][cur_y]=True
        for k in range(8):
            new_x=cur_x+xi[k]
            new_y=cur_y+yi[k]
            if 0<=new_x<n and 0<=new_y<m and not two_dimensional[new_x]
                dfs(new_x,new_y,step+1)
        two_dimensional[cur_x][cur_y]=False
    dfs(x,y,1)
    return count[0]
T=int(input())
for i in range(T):
    n,m,x,y=map(int,input().split())
    print(calculation(n,m,x,y))
```

基本信息

#:	47320088
题目:	04123
提交人:	EuphoriaJ
内存:	3608kB
时间:	3072ms
语言:	Python3
提交时间:	2024-11-22 00:40:38

```
### sy316: 矩阵最大权值路径
```

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

思路：经典的 dfs 问题，先初始化变量（directions,opt_path,temp_path)，然后套用 dfs 函数模板（最终结果，中间的过程并回溯），注意最后要将 x,y 加上 1

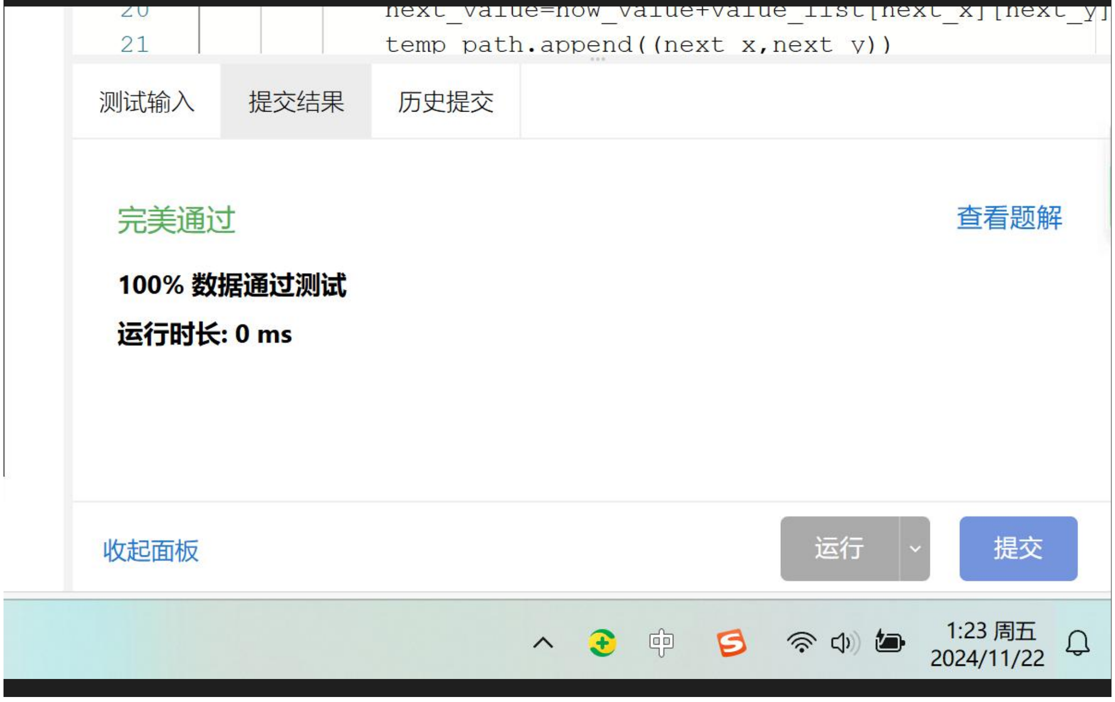
很巧妙的地方：1.用 global 函数来全局化变量；2.用 max_value 设为负无穷来不断更新最大的 value 值（30min）

代码：

```
n,m=map(int,input().split())
value_list=[list(map(int,input().split())) for _ in range(n)]
visit_list=[[False]*m for i in range(n)]
directions=[(1,0),(-1,0),(0,1),(0,-1)]
max_value=float('-inf')
opt_path=[]
temp_path=[(0,0)]
def dfs(x,y,now_value):
    global max_value,opt_path
    if x==n-1 and y==m-1:
        if now_value>max_value:
```

```python
            max_value=now_value
            opt_path=temp_path[:]
        return
    visit_list[x][y]=True
    for dx,dy in directions:
        next_x=x+dx
        next_y=y+dy
        if 0<=next_x<n and 0<=next_y<m and not visit_list[next_x][next_y]:
            next_value=now_value+value_list[next_x][next_y]
            temp_path.append((next_x,next_y))
            dfs(next_x,next_y,next_value)
            temp_path.pop()
    visit_list[x][y]=False
dfs(0,0,value_list[0][0])
for x,y in opt_path:
    x+=1
    y+=1
    print(x,y)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



```
next_value=now_value+value_list[next_x][next_y]
temp_path.append((next_x,next_y))
```

完美通过

100% 数据通过测试

运行时长: 0 ms

### LeetCode62.不同路径

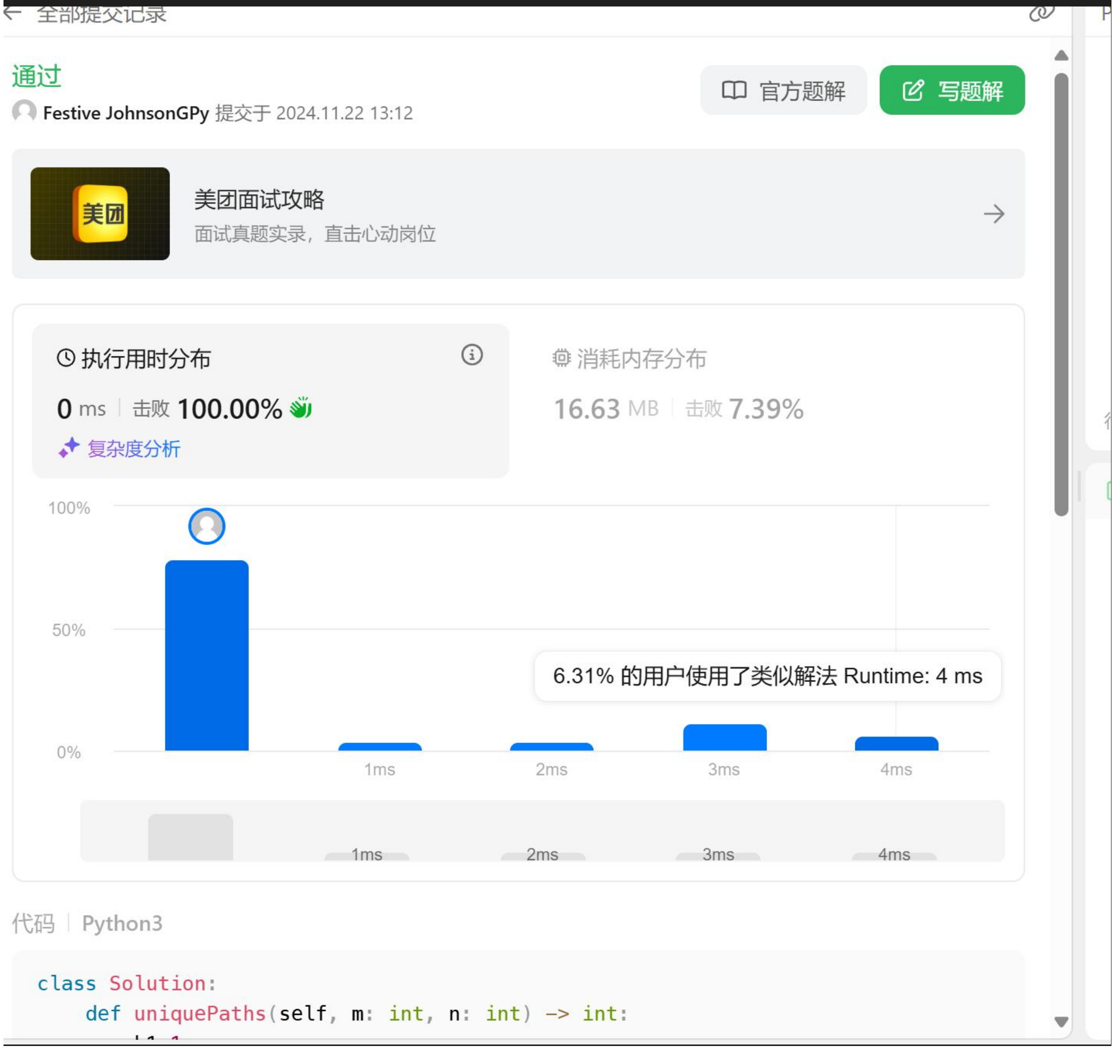
dp, https://leetcode.cn/problems/unique-paths/

思路：从起点到终点一定有（m+n-2）步，其中横着走一共（m-1）步，即从(m+n-2)个数中取出(m-1)个的组合数表示即可（8min）

代码:

```
class Solution:

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

        k1=1

        p1=1

        k2=1

        p2=1

        k3=1

        p3=1

        while 1<=k1<=(m+n-2):

            p1*=k1

            k1+=1

        while 1<=k2<=(m-1):

            p2*=k2

            k2+=1

        while 1<=k3<=(n-1):

            p3*=k3

            k3+=1

        return(int(p1/(p2*p3)))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



← 全部提交记录

通过

Festive JohnsonGPy 提交于 2024.11.22 13:12

官方题解　写题解

美团面试攻略
面试真题实录，直击心动岗位

执行用时分布
0 ms | 击败 100.00%
复杂度分析

消耗内存分布
16.63 MB | 击败 7.39%

6.31% 的用户使用了类似解法 Runtime: 4 ms

代码 | Python3

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

### sy358：受到祝福的平方

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

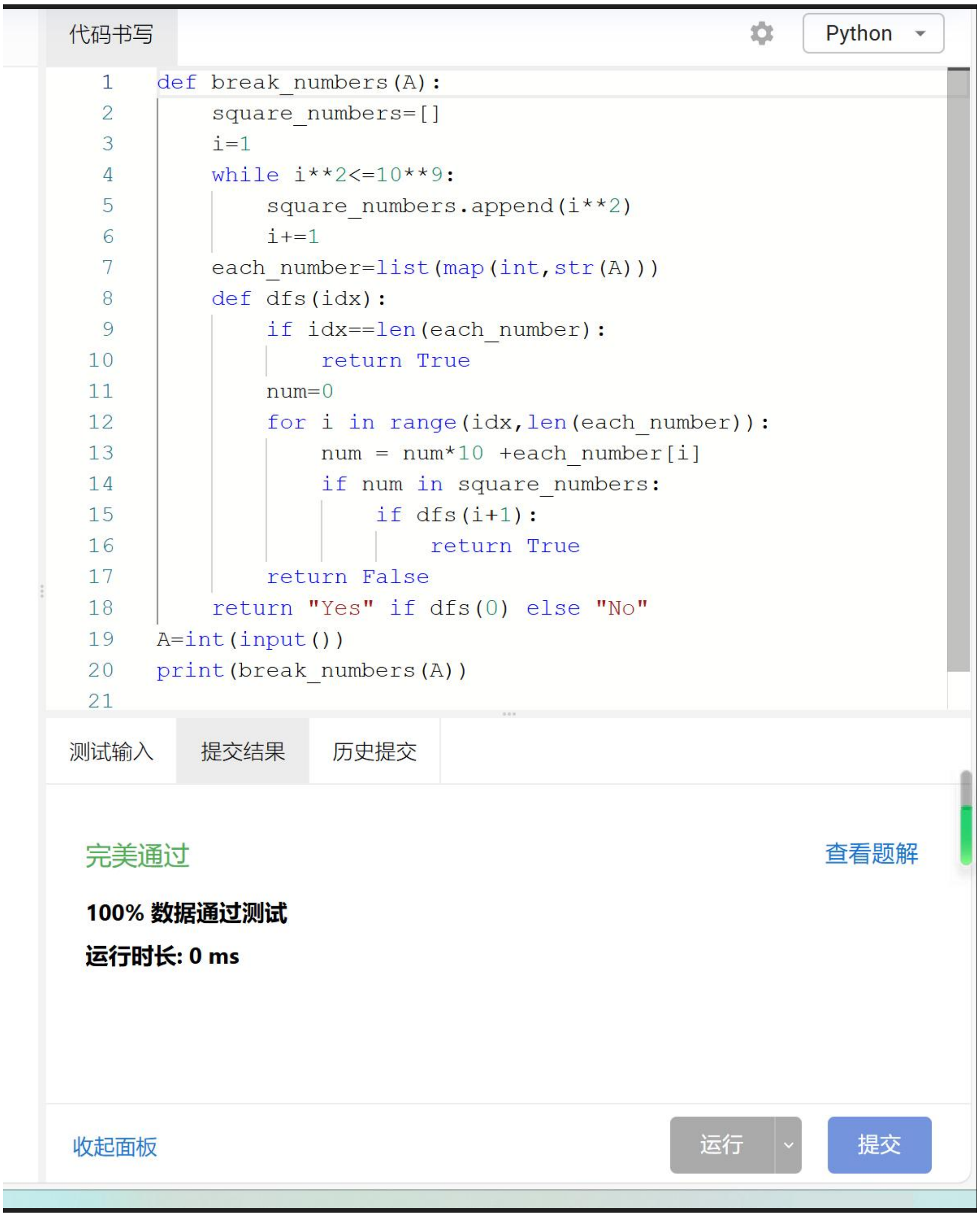
思路：也是一道典型的 dfs 问题，先初始化变量，然后再把所有平方数生成列表，再套用 dfs 模版（最终状态，即 idx=len(each_number)），中间状态（巧妙之处就在于它很好表示了拆分后

再将数字给组合起来），再写出 dfs 从哪里开始（0）。（1h）

代码：

```
def break_numbers(A):
    square_numbers=[]
    i=1
    while i**2<=10**9:
        square_numbers.append(i**2)
        i+=1
    each_number=list(map(int,str(A)))
    def dfs(idx):
        if idx==len(each_number):
            return True
        num=0
        for i in range(idx,len(each_number)):
            num = num*10 +each_number[i]

            if num in square_numbers:

                if dfs(i+1):

                    return True

        return False

    return "Yes" if dfs(0) else "No"

A=int(input())

print(break_numbers(A))
```

```
代码运行截图 <mark>（至少包含有"Accepted"）</mark>
```



```python
def break_numbers(A):
    square_numbers=[]
    i=1
    while i**2<=10**9:
        square_numbers.append(i**2)
        i+=1
    each_number=list(map(int,str(A)))
    def dfs(idx):
        if idx==len(each_number):
            return True
        num=0
        for i in range(idx,len(each_number)):
            num = num*10 +each_number[i]
            if num in square_numbers:
                if dfs(i+1):
                    return True
        return False
    return "Yes" if dfs(0) else "No"
A=int(input())
print(break_numbers(A))
```

完美通过

100% 数据通过测试

运行时长: 0 ms

## 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>

感觉学到了很多很多东西，对 bfs 和 dfs 熟悉了很多，还学会了 sys 和 try except，但作业实在太难了 qwq，只能独立写出一个题，剩下的倒是用 AI 很快能看懂，但感觉
离自己写出来还差了很多（感觉自己写就会漏掉很多东西，比如初始化变量根本不完全，def 的变量老是搞不清楚），找这种感觉期末考试是完犊紫啊....（每日选做倒是补上很多
毕竟一周没看数学....）据说有些班的月考题能很快 AC4-5 个，而我们班的题明显比他们难，这种放在期末考试中是否对我们不会太好（尤其是对苯人这种比较菜的），到时候只 AC
两三个岂不是完蛋了......