Assignment #9: dfs, bfs, & dp

Updated 2107 GMT+8 Nov 19, 2024

2024 fall, Complied by <mark>汤伟杰，信息管理系</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：

dfs 的作用是计数每一片的 cnt，在主程序部分每次调用函数，都要把返回的 cnt 与当前的最大值 a 取最大值，这样最后就得到了最大值。dfs 内部是对每个 W 点原地修改，就是 lake counting 的模型。

代码：

```
'''dfs
最大连通域面积 http://cs101.openjudge.cn/practice/18160
'''
#pylint:skip-file
def dfs(i,j):
    global cnt
    s[i][j]='.'
    cnt+=1
    dx=[-1,0,1,-1,1,-1,0,1]
    dy=[-1,-1,-1,0,0,1,1,1]
    for _ in range(8):
        x=i+dx[_]
        y=j+dy[_]
        if 0<=x<=n-1 and 0<=y<=m-1:
            if s[x][y]=='W':
                dfs(x,y)
    return cnt

for _ in range(int(input())):
    n,m=map(int,input().split())
    a=0
```

```
    s=[[i for i in input()] for _ in range(n)]

    for i in range(n):
        for j in range(m):
            if s[i][j]=='W':
                cnt=0
                ans=dfs(i,j)
                a=max(ans,a)
    print(a)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

状态: Accepted

源代码

```
'''dfs
最大连通域面积 http://cs101.openjudge.cn/practice/18160
'''
#pylint:skip-file
def dfs(i,j):
    global cnt
    s[i][j]='.'
    cnt+=1
    dx=[-1,0,1,-1,1,-1,0,1]
    dy=[-1,-1,-1,0,0,1,1,1]
    for _ in range(8):
        x=i+dx[_]
        y=j+dy[_]
        if 0<=x<=n-1 and 0<=y<=m-1:
            if s[x][y]=='W':
                dfs(x,y)
    return cnt


for _ in range(int(input())):
    n,m=map(int,input().split())
    a=0
    s=[[i for i in input()] for _ in range(n)]

    for i in range(n):
        for j in range(m):
            if s[i][j]=='W':
                cnt=0
                ans=dfs(i,j)
                a=max(ans,a)

    print(a)
```

19930: 寻宝

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

思路：

一开始用 dfs 没做出来，后来才发现这题标签是 bfs。bfs 函数的作用是依次遍历每一个点和它周围的四个点，一旦发现了 1，直接 return 步数，因为是 bfs，只要第一次找到就是最短路径。由于函数部分不是像 dfs 那样需要检测退出条件，所以在开头额外检测了初始位置是不是 1。即对于数据

```
1 1
1
```

如果少了开头的检测，会输出 NO。

代码：

```
'''bfs
寻宝 http://cs101.openjudge.cn/practice/19930
'''
from collections import deque
dx,dy=[0,-1,1,0],[-1,0,0,1]

def bfs(x,y):
    inq=set()
    inq.add((x,y))
    q=deque()
    q.append((0,x,y))
    #额外检测第一个位置是不是 1
    if maze[x][y]==1:
        return 0
    while q:
        front=q.popleft()
        for i in range(4):
            nx=front[1]+dx[i]
            ny=front[2]+dy[i]
            if maze[nx][ny]==0 and (nx,ny) not in inq:
                inq.add((nx,ny))
                q.append((front[0]+1,nx,ny))
            if maze[nx][ny]==1 and (nx,ny) not in inq:
                return front[0]+1
    return 'NO'

n,m=map(int,input().split())
maze=[[-1]*(m+2)]+[[-1]+list(map(int,input().split()))+[-1] for _ in
range(n)]+[[-1]*(m+2)]
result=bfs(1,1)
print(result)
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
'''bfs
寻宝 http://cs101.openjudge.cn/practice/19930
'''
from collections import deque
dx,dy=[0,-1,1,0],[-1,0,0,1]

def bfs(x,y):
    inq=set()
    inq.add((x,y))
    q=deque()
    q.append((0,x,y))
    if maze[x][y]==1:
        return 0
    while q:
        front=q.popleft()
        for i in range(4):
            nx=front[1]+dx[i]
            ny=front[2]+dy[i]
            if maze[nx][ny]==0 and (nx,ny) not in inq:
                inq.add((nx,ny))
                q.append((front[0]+1,nx,ny))
            if maze[nx][ny]==1 and (nx,ny) not in inq:
                return front[0]+1
    return 'NO'

n,m=map(int,input().split())
maze=[[-1]*(m+2)]+[[-1]+list(map(int,input().split()))+[-1] for _ in ra
result=bfs(1,1)
print(result)
```

基本信息

#:	47325030
题目:	19930
提交人:	24n2400016635
内存:	3728kB
时间:	31ms
语言:	Python3
提交时间:	2024-11-22 14:56:

04123: 马走日

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

思路：

dfs 的推出条件很像全排列，如果遍历完成，就是马走了 n*m 步，此时就要退出，所以设定一个记录步数的参数 k 作为检测退出条件的数字，一旦要退出了，就将路径数 num+=1，所以这个 dfs 函数的作用是对 num 计数。

有一个坑是如果把 num 初始化在函数之前而没有放在主程序里，如果有多个测试数据 num 会累计，导致错误。正确的应该放在主函数里，每一组测试数据都要更新一次变成 0。

代码：

```
'''dfs
马走日 http://cs101.openjudge.cn/practice/04123
遍历所有点，n*m 个，每走一步加 1，看看 k 能不能到 n*m
'''
#pylint:skip-file
dx=[-1,1,-2,2,-2,2,-1,1]
dy=[-2,-2,-1,-1,1,1,2,2]
def dfs(x,y,k):
    global num
    if k==total:
```

```
        num+=1
        return

    s[x][y]=1
    for _ in range(8):
        nx=x+dx[_]
        ny=y+dy[_]
        if 0<=nx<n and 0<=ny<m:
            if s[nx][ny]==0:
                dfs(nx,ny,k+1)
                s[nx][ny]=0

for _ in range(int(input())):
    num=0
    n,m,x,y=map(int,input().split())
    total=n*m
    s=[[0]*m for _ in range(n)]
    dfs(x,y,1)
    print(num)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

状态: Accepted

源代码

```
'''dfs
马走日 http://cs101.openjudge.cn/practice/04123
遍历所有点，n*m个，每走一步加1，看看k能不能到n*m
'''
#pylint:skip-file
dx=[-1,1,-2,2,-2,2,-1,1]
dy=[-2,-2,-1,-1,1,1,2,2]
def dfs(x,y,k):
    global num
    if k==total:
        num+=1
        return

    s[x][y]=1
    for _ in range(8):
        nx=x+dx[_]
        ny=y+dy[_]
        if 0<=nx<n and 0<=ny<m:
            if s[nx][ny]==0:
                dfs(nx,ny,k+1)
                s[nx][ny]=0


for _ in range(int(input())):
    num=0
    n,m,x,y=map(int,input().split())
    total=n*m
    s=[[0]*m for _ in range(n)]
    dfs(x,y,1)
    print(num)
```

基本信息

#:	47324123
题目:	04123
提交人:	24n2400016635
内存:	3572kB
时间:	3435ms
语言:	Python3
提交时间:	2024-11-22 13:47:20

sy316: 矩阵最大权值路径

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

思路：

dfs 函数的作用是更新 max_path，为了保持记录无重复且防止少记录，我选择在函数内部进行 now_path.append((x,y))，并且在函数结尾处回溯 now_path.pop()，因此主程序传入的参数中 now_v=0,now_path=[]。其余部分就与矩阵最大权值差不多了。此外还有一个坑是 max_path 是 now_path 的复制列表，要用 max_path=now_path[:]。

代码：

```
'''dfs
矩阵最大权值路径 中等 https://sunnywhy.com/sfbj/8/1/316
'''
max_v=-float('inf')
max_path=[]
dx,dy=[0,-1,1,0],[-1,0,0,1]
def dfs(x,y,now_v,now_path):
    global max_v,max_path
    if x==n and y==m:
        now_v+=maze[n][m]
        now_path.append((n,m))
        if now_v>=max_v:
            max_v=now_v
            max_path=now_path[:]
        now_path.pop()
        return

    now_path.append((x,y))
    for i in range(4):
        nx,ny=x+dx[i],y+dy[i]
        if maze[nx][ny]!=-200:
            curr=maze[x][y]
            maze[x][y]=-200
            dfs(nx,ny,now_v+curr,now_path)
            maze[x][y]=curr
    now_path.pop()

n,m=map(int,input().split())
maze=[[-200]*(m+2)]+[[-200]+list(map(int,input().split()))+[-200] for _ in
range(n)]+[[-200]*(m+2)]
dfs(1,1,0,[])
for i in max_path:
    print(*i)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

LeetCode62.不同路径

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

思路：

二维 dp，每一个位置表示到达该位置的路径数，因此 dp[-1] [-1]就是最后的答案。其中 dp 的第一行和第一列都是 1，其余位置的递推是 dp[i] [j]=dp[i-1] [j]+dp[i] [j-1]，就是其左边和上边两个位置的路径数之和（因为只有这两个位置能到达 dp[i] [j]）。

代码：

```
'''dp
不同路径 https://leetcode.cn/problems/unique-paths/
'''
class Solution:
    def uniquePaths(self, m: int, n: int) -> int:
        dp=[[0 for _ in range(n)] for _ in range(m)]
        for i in range(n):
            dp[0][i]=1
        for i in range(m):
            dp[i][0]=1
        for i in range(1,m):
            for j in range(1,n):
                dp[i][j]=dp[i-1][j]+dp[i][j-1]
        return dp[-1][-1]
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

sy358: 受到祝福的平方

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

思路：

dfs 函数的作用是修改全局变量 can_split，思路是遍历每一个 index 和以这个 index 为开头，对应的所有子链，一旦发现这个子链是平方数，就对子链的下一个位置的数字进行递归；否则就说明以这个 index 开头找不到符合的子链，此时有隐式的自动返回上一层的 dfs（因为 dfs 函数内部如果没进入下一层 dfs，就相当于无变化，返回了上一层 dfs）。因此对所有分割方式进行了遍历，答案是正确的。

代码：

```
'''dfs
受到祝福的平方 中等 https://sunnywhy.com/sfbj/8/3/539
'''

can_split=False
def dfs(n,i):
    global can_split
    if can_split:
        return
    for j in range(i,n):
        if is_square(int(s[i:j+1])) and int(s[i:j+1])!=0:
            if j==n-1:
```

```
                can_split=True
                return
            dfs(n,j+1)

def is_square(x):
    if x**0.5==int(x**0.5):
        return True
    return False

s=input()
n=len(s)
dfs(n,0)
print('Yes' if can_split else 'No')
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

代码书写 Python

```python
can_split=False
def dfs(n,i):
    global can_split
    if can_split:
        return
    for j in range(i,n):
        if is_square(int(s[i:j+1])) and int(s[i:j+
            if j==n-1:
                can_split=True
                return
            dfs(n,j+1)

def is_square(x):
    if x**0.5==int(x**0.5):
        return True
    return False

s=input()
n=len(s)
```

测试输入 提交结果 历史提交

完美通过 查看题解

100% 数据通过测试

运行时长: 0 ms

收起面板 运行 提交

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>

dfs 对我的启蒙题目是全排列和 lake counting，在全排列学到了在函数开头设置退出条件，在 lake counting 学到了设置 dx 和 dy 来进行递归并原地修改，同时在回溯时修改为原始状态。

感觉像矩阵最大权值路径和最大连通区域面积的题目，要额外设置一些参数，这些参数在 dfs 函数中修改的逻辑和初始值的设定都需要具体问题具体分析，我做题的时候总是会出错，但是要保持冷静，查看一下参数到底应该放在哪里修改，才能既不冗余，又是必要的。

bfs 的题目还在进一步做题，积累手感！