[P2825][HEOI2016/TJOI2016]游戏

Algorithm	匈牙利算法 图论建模
Created	@Aug 10, 2020 7:07 PM
Difficulty	省选/NOI-
Related to 近期更新 (Property)	❤ [P2825][HEOI2016/TJOI2016]游戏
URL	https://www.luogu.com.cn/problem/P2825

画外音：在调试时我是这个表情：；AC了这题时我是这个表情：；看到这题的题解还没满时我是这个表情：

题目链接：

[HEOI2016/TJOI2016]游戏

在 2016 年，佳媛姐姐喜欢上了一款游戏，叫做泡泡堂。 简单的说，这个游戏就是在一张地图上放上若干个炸弹，看是否能炸到对手，或者躲开对手的炸弹。在玩游戏的过程中，小 H 想到了这样一个问题：当给定一张地图，在这张地图上最多能放上多少个炸弹能使得任意两个炸弹之间不会互相炸到。炸弹能炸到的范围是

https://www.luogu.com.cn/problem/P2825

「TJOI / HEOI2016」游戏 - 题目 - LibreOJ

在 2016 年，佳媛姐姐喜欢上了一款游戏，叫做泡泡堂。 简单的说，这个游戏就是在一张地图上放上若干个炸弹，看是否能炸到对手，或者躲开对手的炸弹。在玩游戏的过程中，小 H 想到了这样一个问题：当给定一张地图，在这张地图上最多能放上多少个炸弹能使得任意两个炸弹之间不会互相炸到。炸弹能炸到的范围是

https://loj.ac/problem/2057

题解：

在做这题前，你（不）需要先AC这题并且熟练掌握匈牙利算法求二分图的最大匹配：

[ZJOI2007]矩阵游戏

小 Q 是一个非常聪明的孩子，除了国际象棋，他还很喜欢玩一个电脑益智游戏——矩阵游戏。矩阵游戏在一个 $n \times n$ 黑白方阵进行（如同国际象棋一般，只是颜色是随意的）。每次可以对该矩阵进行两种操作： - 行交换操作：选择矩阵的任意两行，交换这两行（即交换对应格子的颜色）。 - 列交换操作：选择矩

https://www.luogu.com.cn/problem/P1129

首先观察一下题目，若是把硬石子全部当成软石子，就可以将题意转换成：“每摆放一个炸弹，它所在的行和列都不再能摆放其它的棋子，其中本身就有一些点是不可以摆放炸弹的。”是不是很简单？将行看成集合S，将列看成集合T，那么能够摆放的炸弹的个数就是S与T的最大匹配（两个集合内部不会相交，众所周知，行与行不相交，列与列也不相交），这是一个经典的二分图最大匹配应用题。

加了硬石子，又怎么做？

我们可以从二分图的本质考虑起。二分图重在同一集合内部元素不相交，不同集合元素有相连边。那么此时，如果一行内存在一个硬石子，这一行就会被分成不相交的两部分。如果把这两部分都看成一个独立的行，此时，行与行仍然不相交，列与行仍然会相交，满足二分图的性质，那么，此方案可行。

拿样例举个例子吧：

```
#***
*#**
**#*
xxx#
```

原来的矩阵长这样

```
#111
2#22
33#3
xxx#
```

如果将硬石子看成软石子，我们会这样给每行中可以放炸弹的节点编号

```
#111
2#33
44#5
xxx#
```

但是硬石子将每行划分成了多个单独的行，因此我们要将硬石子后的可放石子的节点所处的行加一。注意：这里的行已经不是原来意义上的行了，这里是被硬石子分割后抽象出来的单独的行

这样，我们就把行的编号处理完了，列的编号也同理。

直接上代码吧，这样方便理解，几个比较坑的点在注释里讲了：

```
//File: P2825.cpp
//Author: yanyanlongxia
//Date: 2020/8/10
//
#include <bits/stdc++.h>
using namespace std;
int n,m,ntot,mtot,tot,head[3000],nxt[3000],ver[3000],row[60][60],col[60][60],matc
h[3000],ans;
bool vis[3000];
void add(int x,int y)//链式前向星加边
{
    ver[++tot]=y;
    nxt[tot]=head[x];
    head[x]=tot;
}
bool find(int x)//匈牙利算法
{
    for(int i=head[x];i;i=nxt[i])
    {
        int y=ver[i];
        if(!vis[y])
        {
            vis[y]=true;
            if(!match[y] || find(match[y]))
            {
                match[y]=x;
                return true;
            }
        }
    }
    return false;
}
int main() {
    scanf("%d%d",&n,&m);
    char s[60][60];
    bool flag=1;//注意第一次需要单开一行
    pair<int,int>last;//与动态规划的思想相似，每个点所在的抽象出的行的编号都可以由上一个有编号的
点转移过来
    for(int i=1;i<=n;i++)
    {
        scanf("%s",(s[i]+1));
        for(int j=1;j<=m;j++)
```

```
        {
            if(s[i][j]=='#')//下一次另开一行（抽象意义上的行）
            {
                flag=1;
                continue;
            }
            if(flag==0)//不需要加1
            {
                if(s[i][j]!='x')
                {
                    row[i][j]=row[last.first][last.second];//row数组的意义为该点所在的
抽象的行的编号
                    last=make_pair(i,j);
                }
            } else
                if(s[i][j]!='x')
                {
                    row[i][j]=row[last.first][last.second]+1;
                    ntot++;
                    last=make_pair(i,j);
                    flag=0;//当前的加处理完毕
                }
        }
        flag=1;//换行一定要加一
    }
    flag=1;
    last=make_pair(0,0);//清零，赋初值
    for(int i=1;i<=m;i++)//列的处理，与行相似，但是i与j要取反
    {
        for(int j=1;j<=n;j++)
        {
            if(s[j][i]=='#')
            {
                flag=1;
                continue;
            }
            if(flag==0)
            {
                if(s[j][i]!='x')
                {
                    col[j][i]=col[last.second][last.first];//col数组与row数组的意义相
似，是列的编号
                    last=make_pair(i,j);
                }
            } else
                if(s[j][i]!='x')
                {
                    col[j][i]=col[last.second][last.first]+1;
                    mtot++;
                    last=make_pair(i,j);
                    flag=0;
                }
        }
        flag=1;
    }
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j) {
            if (s[i][j]=='*')
```

```
                add(row[i][j],col[i][j]);//加边
        }
    for (int i = 1; i <= ntot; ++i) {
        memset(vis,0,sizeof(vis));
        if(find(i))
            ans++;//统计最大匹配
    }
    printf("%d\n",ans);
    return 0;
}
```

另外，如果你的程序过了样例，但是却WA了，洛谷上还无法下载数据，你可以到LOJ里去下载数据，文章开头有链接。

下面直接附上本题测试数据（虽然和洛谷数据不一样，但也足够让你从WA变成AC了）：

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0cecc638-713c-4d50-9915-287949ef120e/2057.zip