dfs2.0

题目描述

> 在峰会期间，武装部队得处于高度戒备。警察将监视每一条大街，军队将保卫建筑物，领空将布满了F-2003飞机。此外，巡洋船只和舰队将被派去保护海岸线。不幸的是因为种种原因，国防海军部仅有很少的几位军官能指挥大型海战。因此，他们考虑培养一些新的海军指挥官，他们选择了“海战”游戏来帮助学习。
>
> 在这个著名的游戏中，在一个方形的盘上放置了固定数量和形状的船只，每只船却不能碰到其它的船。在这个题中，我们仅考虑船是方形的，所有的船只都是由图形组成的方形。编写程序求出该棋盘上放置的船只的总数。
>
> 输入格式
>
> 输入文件头一行由用空格隔开的两个整数R和C组成，1<=R,C<=1000，这两个数分别表示游戏棋盘的行数和列数。接下来的R行每行包含C个字符，每个字符可以为“#”，也可为“.”，“#”表示船只的一部分，“.”表示水。
>
> 输出格式
>
> 为每一个段落输出一行解。如果船的位置放得正确（即棋盘上只存在相互之间不能接触的方形，如果两个“#”号上下相邻或左右相邻却分属两艘不同的船只，则称这两艘船相互接触了）。就输出一段话“There are S ships.”,S表示船只的数量。否则输出“Bad placement.”。
>
> 输入输出样例
>
> ```
> 输入
> 6 8
> .....#.#
> ##.....#
> ##.....#
> .......#
> #......#
> #..#...#
>
> 输出
> There are 5 ships.
> ```

思路

dfs实质是暴力枚举

关注有什么值在动态变化，剪枝很重要，以及是否需要回溯

小tip 方向数组

这道题的思路是先判断是不是矩形，L型舍弃（一个九宫格里有三个#即为L），在用dfs去掉相连的所有#

这道题不需要回溯，但有的题如需要回溯

talk is cheap,show me the code

```
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
int ans=0;
int dx[4]={0,1,-1,0};//方向数组
int dy[4]={1,0,0,-1};//方向数组
int a,b;
char mapp[1005][1005];
void dfs(int x,int y){
    mapp[x][y]='.';
    for(int i=0;i<4;i++){//使用方向数组
        int xx=x+dx[i];
        int yy=y+dy[i];
        if(xx<=a&&yy<=b&&mapp[xx][yy]=='#'&&xx>=0&&yy>=0)//判断边界
        {
            dfs(xx,yy);
        }
    }
}

int main(){
    int temp=0;
    cin>>a>>b;
    for(int i=0;i<a;i++){
        for(int j=0;j<b;j++){
            cin>>mapp[i][j];
        }
    }
    for(int i=0;i<a-1;i++){
        for(int j=0;j<b-1;j++){//判断L型
            temp=0;
            if(mapp[i][j]=='#') temp++;
            if(mapp[i+1][j]=='#') temp++;
            if(mapp[i][j+1]=='#') temp++;
            if(mapp[i+1][j+1]=='#') temp++;
            if(temp==3) {
```

```
                cout<<"Bad placement.";
                return 0;
            }
        }
    }
    for(int i=0;i<a;i++){
        for(int j=0;j<b;j++){
            if(mapp[i][j]=='#') {
                dfs(i,j); //去掉#
                ans++;
            }
        }
    }
    cout<<"There are "<<ans<<" ships.";
    return 0;
}
```