



深度优先搜索

从一个点开始，沿着某条路往下走，一直走到底（**能深则深**），如果走完后发现不能达到目标解，返回到上一个点（**不能深则退**），换条路，然后继续走到底，如此往复，直至所有可能的结果都被搜索完。

这种**能深则深，不能深则退**的方法，称之为**深度优先搜索**简称**DFS**。

深度优先搜索模板

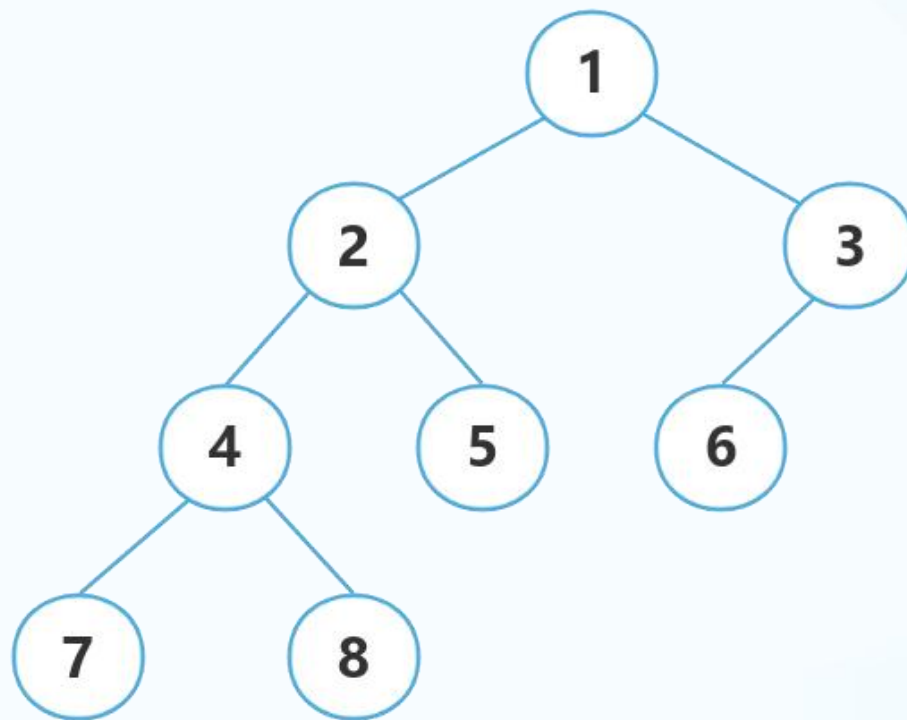
```
int vis[10][10]; // 标记数组
int dx[4]={1,0,-1,0}; // 下右上左
int dy[4]={0,1,0,-1}; // 下右上左
void dfs(int x,int y){
    if( ① 目标条件 ){
        return;
    }
    for(int i=0;i<4;i++){ ② 搜索周围四个方向
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<=n&&ny>0&&ny<=n&&vis[nx][ny]==0){ ③ 判断当前房间是否在
            迷宫内，同时房间未搜索过
            vis[nx][ny]=1; ④ 标记当前房间搜索过
            dfs(nx,ny);
        }
    }
}
```



回顾练习

根据深度优先搜索思想，下图按照**先左后右**的顺序从1号开始搜索，搜索的路径是 **D**
(搜索过的点不再统计)

- A. 1-2-3-4-5-6-7-8
- B. 1-2-4-7-3-6-5-8
- C. 1-2-4-7-5-8-3-6
- D. 1-2-4-7-8-5-3-6



连通块





码上得知-1-认识连通块

想一想

地图如下图所示，1表示障碍不能通行，0表示可通行；从 (1,1) 位置出发，走时只能是“上下左右”相邻的格子移动，并且不能走出地图；这样可以访问多少个格子？

访问的格子数量：

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	1	0	1
3	1	0	1	1	1	1
4	0	0	1	0	0	0
5	0	1	0	0	0	0
6	1	0	1	0	0	0

格子元素相同，任意2个格子都能直接或间接相互到达，格子数量尽可能的多，这种区域称之为**连通块**。

连通块

一个地图内，可能存在一个或多个连通块。
例如下图所示，1表示障碍物不能通行，0表示可通行，格子之间只能“上下左右”四个方向移动。

用0表示的连通块一共有4个，用1表示的连通块一共有1个，整张地图一共有5个连通块。

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	1	1	0	1
3	1	0	1	1	1	1
4	1	1	1	0	0	0
5	1	1	1	0	0	0
6	1	0	1	0	0	0

单独的一个0，也是连通块

练一练1

下列地图中数字0表示可以通行，数字1表示障碍不能通行，只能是相邻四个格子移动，迷宫中用0表示的有 4 个连通块，最大的连通块有 11 个格子。

0	0	1	1	0	0	0
1	0	0	1	0	1	0
1	0	1	1	1	1	0
0	0	1	0	0	0	1
0	1	0	0	0	0	1
0	0	1	0	0	1	0
1	1	1	0	0	1	1



码上得知-2-代码求连通块

岛屿

【问题描述】

一张4*4的地图，0表示岛屿，1表示海水。现在从(1,1)岛屿出发，只能是“上下左右”往相邻的岛屿移动，能到达多少个岛屿？

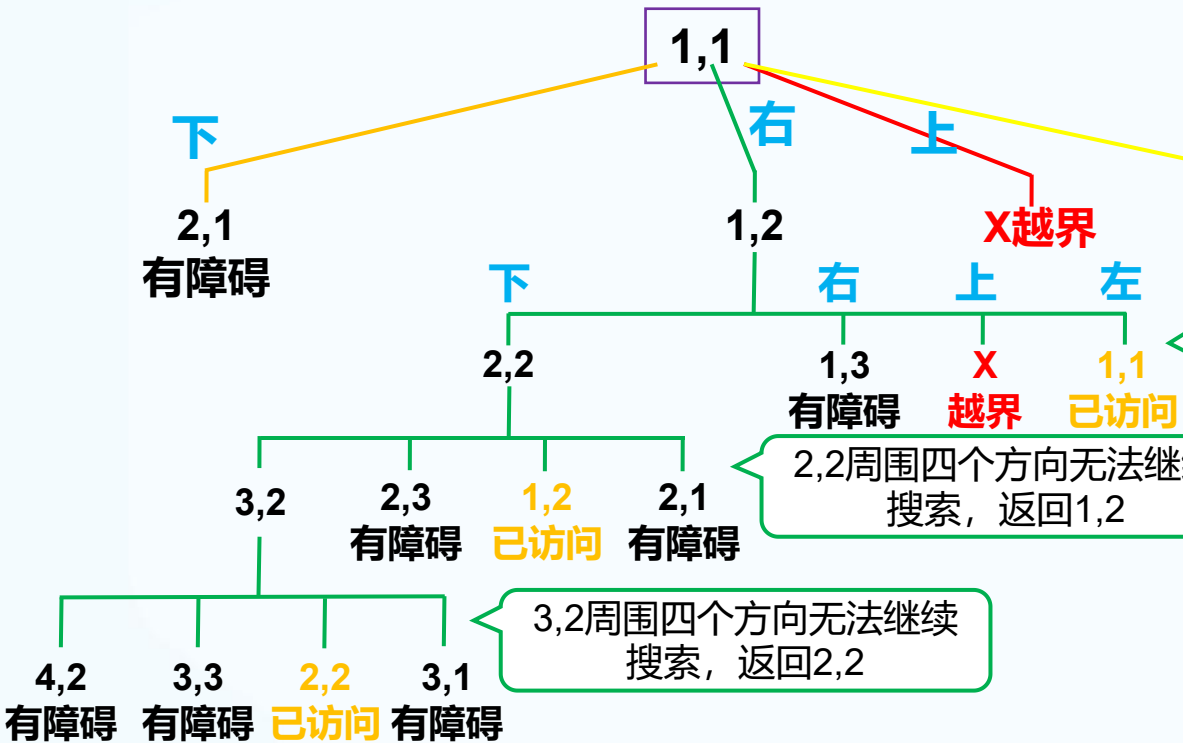
此题求绿色连通块中格子的数量

	1	2	3	4
1	0	0	1	0
2	1	0	1	1
3	1	0	1	1
4	1	1	1	0

从(1,1)点开始往周围四个方向进行搜索，尝试把所有能访问的格子全部访问一遍，这种解题思路符合深度优先搜索。

搜索过程

规定一个搜索顺序：下右上左



能访问的格子，
标记已访问

1,2周围四个方向搜索
结束，返回1,1

2,2周围四个方向无法继续
搜索，返回1,2

3,2周围四个方向无法继续
搜索，返回2,2

	1	2	3	4
1	√	√	1	0
2	1	√	1	1
3	1	√	1	1
4	1	1	1	0

深搜标记过的格子就是所求的岛屿数量
(也就是一个连通块)

深度优先搜索

定义dfs()函数，实现深度优先搜索。

```
int dx[4]={1,0,-1,0}; //下右上左
int dy[4]={0,1,0,-1};
```

当前行号 当前列号

```
void dfs(int x,int y){ //从当前点(x,y)开始深度优先搜索
    for(int i=0;i<4;i++){ //遍历四个方向
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<5&&ny>0&&ny<5){
            //地图内
        }
    }
}
```

(nx,ny)新坐标，需要确保新的坐标在地图范围内，并确保没有障碍物

	1	2	3	4
1	0	0	1	0
2	1	0	1	1
3	1	0	1	1
4	1	1	1	0

) {

深度优先搜索

```
int mp[5][5]; //地图
int dx[4]={1,0,-1,0}; //下右上左
int dy[4]={0,1,0,-1};
int vis[5][5]; //标记数组
int ans; //统计已搜索格子数量

void dfs(int x,int y){ //从当前点(x,y)开始深度优先搜索
    for(int i=0;i<4;i++){ //遍历四个方向
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<5&&ny>0&&ny<5&&mp[nx][ny]==0&&vis[nx][ny]==0){
            vis[nx][ny]=1; //标记已搜索
            ans++; //每标记一个格子,就统计一次
            dfs(nx,ny); //从(nx,ny)继续搜索
        }
    }
}
```

全局变量
创建二维数组vis标记是否搜索, 0表示未搜索, 1表示已搜索

	1	2	3	4
1	0	0	1	0
2	1	0	1	1
3	1	0	1	1
4	1	1	1	0

mp[][]

可以通行

未搜索过

地图内



主函数调用dfs()函数

主函数调用dfs()函数，实现从(1,1)点开始搜索。

```
int main(){
    for(int i=1;i<=4;i++){
        for(int j=1;j<=4;j++){
            cin>>mp[i][j];//读入地图数据
        }
    }
    vis[1][1]=1;//标记(1,1)格子已搜索
    ans=1;//记录搜索过的格子数量
    dfs(1,1);//从(1,1)开始向周围搜索
    cout<<ans<<endl;
    return 0;
}
```

	1	2	3	4
1	√	0	1	0
2	1	0	1	1
3	1	0	1	1
4	1	1	1	0

mp[]

写一写

```
#include<bits/stdc++.h>
using namespace std;
int mp[5][5]; //地图
int dx[4]={1,0,-1,0}; //下右上左
int dy[4]={0,1,0,-1};
int vis[5][5]; //标记数组
int ans; //统计已搜索格子数量
void dfs(int x,int y){ //从当前点(x,y)开始深度优先搜索
    for(int i=0;i<4;i++){ //遍历四个方向
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<5&&ny>0&&ny<5&&mp[nx][ny]==0
            &&vis[nx][ny]==0){
            vis[nx][ny]=1; //标记已搜索
            ans++;
            dfs(nx,ny); //从(nx,ny)继续搜索
        }
    }
}
```

```
int main(){
    for(int i=1;i<=4;i++){
        for(int j=1;j<=4;j++){
            cin>>mp[i][j];
        }
    }
    vis[1][1]=1;
    ans=1;
    dfs(1,1);
    cout<<ans<<endl;
    return 0;
}
```

连通块搜索模版

```
int mp[N][M]; // 地图
int vis[N][M]; // 标记数组
int dx[4] = {1, 0, -1, 0}; // 下右上左
int dy[4] = {0, 1, 0, -1};
int ans; // 统计能到达的格子
void dfs(int x, int y){
    for(int i=0; i<4; i++){ ① 搜索周围四个方向
        int nx = x + dx[i]; // N*M的地图内
        int ny = y + dy[i]; // 可以通行
        if(nx > 0 && nx <= n && ny > 0 && ny <= m && mp[nx][ny] == 0 && vis[nx][ny] == 0){ // 未搜索过
            ② 判断当前格子是否在迷宫内, 同时满足格子可以通行, 且格子未搜索过
            vis[nx][ny] = 1; // ③ 标记当前格子搜索过, 并统计能访问格子的数量
            ans++;
            dfs(nx, ny);
        }
    }
}
```

连通块应用





红与黑

【问题描述】

有一间长方形的房子，地上铺了**红色**、**黑色**两种颜色的正方形瓷砖。你站在其中一块黑色的瓷砖上，只能向四周相邻的黑色瓷砖移动。请写一个程序，计算你总共能够到达多少块黑色的瓷砖。

【输入格式】

第一行是两个整数 n 和 m ，表示房间是 n 行 m 列大小($1 < n, m < 20$)。在接下来的 n 行中，每行包括 m 个字符。每个字符表示一块瓷砖的颜色，规则如下：

- 1) **'.'**：黑色的瓷砖；
- 2) **'#'**：红色的瓷砖；
- 3) **'@'**：黑色的瓷砖，并且你站在这块瓷砖上。该字符在数据中出现一次。

【输出格式】

一行，显示你从初始位置出发能到达的瓷砖数(**注意**：记数时包括初始位置的瓷砖)。

【输入样例】

```
5 6
....#.
....#
.....
#@...#
.#..#
```

【输出样例】

21

问题分析

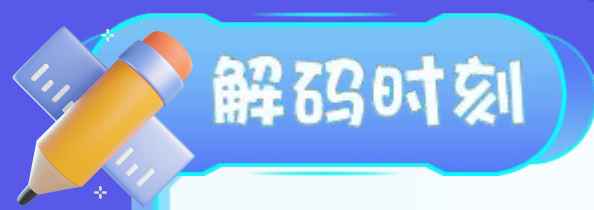
n 行 m 列的地板一共有三种字符:

- 1) '.' : 黑色的瓷砖, 相当于0;
- 2) '#' : 红色的瓷砖, 相当于1;
- 3) '@' : 黑色的瓷砖, 表示开始的位置 (该字符在数据中出现一次) 相当于0。

记数时包括初始位置的瓷砖

	1	2	3	4	5	6
1	.	.	.	.	#	.
2	.	.	.	.	.	#
3	.	.	.	.	.	.
4	#	@	.	.	.	#
5	.	#	.	.	#	.

从 '@' 开始向四周相邻的格子搜索, 统计可以到达的格子数量, 深度优先搜索可以解决。



解题步骤

1. 创建需要的数组及变量。

```
char mp[20][20]; // 地图  
int vis[20][20]; // 标记数组  
int n, m; // n行m列
```

2. 主函数内读入数据

```
int sx, sy; // (sx, sy) @ 的位置  
cin >> n >> m;  
for (int i = 1; i <= n; i++) {  
    for (int j = 1; j <= m; j++) {  
        cin >> mp[i][j]; // 读入地图数据  
        if (mp[i][j] == '@') {  
            sx = i;  
            sy = j;  
        }  
    }  
}
```

解题步骤

3. 创建dfs()函数，实现深度优先搜索。

```
int dx[4]={1,0,-1,0}; // 下右上左
```

连通块模板

```
int dy[4]={0,1,0,-1}; // 下右上左
```

```
int ans; // 统计能到达的格子
```

```
void dfs(int x,int y){
```

```
    for(int i=0;i<4;i++){ ①
```

搜索周围四个方向

```
        int nx=x+dx[i];
```

N*M的地图内

```
        int ny=y+dy[i];
```

可以通行

```
        if((nx>0&&nx<=n&&ny>0&&ny<=m&& mp[nx][ny]!='#' && vis[nx][ny]==0)){ ②
```

```
            vis[nx][ny]=1;
```

```
            ans++;
```

```
            dfs(nx,ny);
```

③

标记当前格子已搜索，并统计能到达的格子

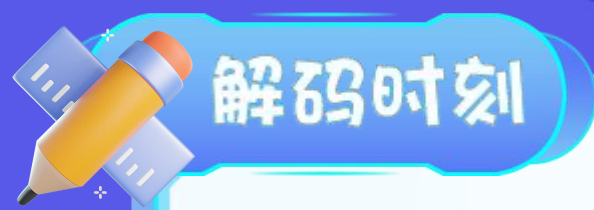
```
        }
```

```
    }
```

```
}
```

	1	2	3	4	5	6
1	.	.	.	.	#	.
2	.	.	.	.	.	#
3	.	.	.	.	.	.
4	#	@	.	.	.	#
5	.	#	.	.	#	.

未搜索过



解题步骤

4. 主函数调用dfs()函数。

```
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>mp[i][j];//读入地图数据
            if(mp[i][j]=='@'){
                sx=i;
                sy=j;
            }
        }
    }
    vis[sx][sy]=1;//@位置标记已访问
    ans=1;//包括初始位置的瓷砖
    dfs(sx,sy);
    cout<<ans<<endl;
    return 0;
}
```

完整代码

```
#include<bits/stdc++.h>
using namespace std;
char mp[20][20]; //地图
int vis[20][20]; //标记数组
int n,m,ans; //n行m列 ans统计可以到达的黑色瓷砖数量
int sx,sy; // (sx,sy)@的位置
int dx[4]={1,0,-1,0},dy[4]={0,1,0,-1}; //下右上左
void dfs(int x,int y){ //从当前点(x,y)开始深度优先搜索
    for(int i=0;i<4;i++){
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<=n&&ny>0&&ny<=m&&mp[nx][ny]!='#'
            &&vis[nx][ny]==0){
            vis[nx][ny]=1; //标记已搜索
            ans++; //统计能到达的黑色瓷砖
            dfs(nx,ny);
        }
    }
}
```

```
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>mp[i][j];
            if(mp[i][j]=='@'){
                sx=i;
                sy=j;
            }
        }
    }
    vis[sx][sy]=1;
    ans=1; //包括初始位置的瓷砖
    dfs(sx,sy);
    cout<<ans<<endl;
    return 0;
}
```



红与黑题目链接

OJ网站 6822 红与黑

<https://oj.aicoders.cn/problem/6822>



疾病预防系统

【问题描述】

某国新研发了一套疾病预防系统，传染病爆发时，监测系统会把下面反应上来的情况在系统中形成一个矩形阵列，由数字 **0** 和 **1** 组成，数字1代表**染病居民**，现在求出感染区域的个数。一个感染区域为染病居民**上下左右**为1的连通区域。

【输入格式】

第一行为两整数 n, m ，表示矩阵的大小为 $n \times m$ ($0 < m, n \leq 100$)。

从第二行开始是一个 $n \times m$ 的矩阵。

【输出格式】

只有一行为矩阵中的感染区域的个数。

【输入样例】

```
5 6
0 1 0 1 0 1
0 1 0 1 1 1
0 1 1 0 0 1
1 0 0 0 0 0
1 0 0 1 0 0
```

【输出样例】

```
4
```

问题分析

数字1代表染病居民，现在求出感染区域的个数。感染区域为染病居民**上下左右**为1的连通区域。此题求1表示的连通块数量。

	1	2	3	4	5	6
1	0	1	0	1	0	1
2	0	1	0	1	1	1
3	0	1	1	0	0	1
4	1	0	0	0	0	0
5	1	0	0	1	0	0

问题分析

1. (1,1) 位置开始遍历矩阵（从左到右，从上到下）寻找数字1。
2. 搜索当前数字1所在的连通块，标记这个连通块所有格子为已搜索。
3. 标记为已搜索的格子不再搜索。

1表示的连通块数量：

2

数量+1

搜索这个连通块，
标记格子已搜索

标记的格子不再搜索

搜索这个连通块，
标记格子已搜索

继续做相同操作，
直到遍历完矩阵

	1	2	3	4	5	6
1	0 → √	→ 0 → √	→ 0 → √	→ 0 → √		
2	→ 0	→ √	→ 0	→ √	→ √	→ √
3	→ 0	→ √	→ √	→ 0	→ 0	→ √
4	√	0	0	0	0	0
5	√	0	0	√	0	0

解题步骤

1. 创建需要的数组及变量。

```
int mp[101][101]; //地图
int vis[101][101]; //标记数组
int n,m; //n行m列
```

2. 主函数内读入数据

```
cin>>n>>m; //n行m列
for(int i=1;i<=n;i++) {
    for(int j=1;j<=m;j++) {
        cin>>mp[i][j];
    }
}
```

	1	2	3	4	5	6
1	0	1	0	1	0	1
2	0	1	0	1	1	1
3	0	1	1	0	0	1
4	1	0	0	0	0	0
5	1	0	0	1	0	0

解题步骤

3. 创建dfs()函数，实现深度优先搜索。

```
int dx[4]={1,0,-1,0}; // 下右上左
int dy[4]={0,1,0,-1}; // 下右上左
```

连通块模板

```
void dfs(int x,int y){
```

```
    for(int i=0;i<4;i++){ ① 搜索四个方向
```

```
        int nx=x+dx[i];
```

n*m的地图内

```
        int ny=y+dy[i];
```

可以通行

```
        if((nx>0&&nx<=n&&ny>0&&ny<=m&& mp[nx][ny]==1 && vis[nx][ny]==0)){ ②
```

```
            vis[nx][ny]=1;
```

③ 标记当前格子搜索过，可以通行

```
            dfs(nx,ny);
```

```
        }
```

```
    }
```

```
}
```

	1	2	3	4	5	6
1	0	1	0	1	0	1
2	0	1	0	1	1	1
		1	1	0	0	1
4	1	0	0	0	0	0
5	1	0	0	1	0	0

未搜索过

解题步骤

4. 遍历地图，如果是数字1且没有标记已搜索，搜索这个连通块。

```
int cnt=0; //统计有多少区域
for(int i=1; i<=n; i++){
    for(int j=1; j<=m; j++){
        if(mp[i][j]==1 && vis[i][j]==0){
            vis[i][j]=1; //标记已访问
            dfs(i,j);
            cnt++;
        }
    }
}
cout<<cnt;
```

遍历完所有格子，输出1表示的连通块数量

	1	2	3	4	5	6
1	0	→√	0	√	0	√
2	0	√	0	√	√	√
3	0	√	√	0	0	√
4	√	0	0	0	0	0
5	√	0	0	√	0	0

完整代码

```
#include <bits/stdc++.h>
using namespace std;
int mp[101][101]; //地图
int vis[101][101]; //标记数组
int n, m, cnt; //n行m列cnt统计有多少个区域
int dx[4]={1,0,-1,0}, dy[4]={0,1,0,-1}; //下右上左
void dfs(int x, int y) { //从当前点(x,y)开始深搜
    for(int i=0; i<4; i++) {
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<=n&&ny>0&&ny<=m&&mp[nx][ny]==1
            &&vis[nx][ny]==0){
            vis[nx][ny]=1; //标记已访问
            dfs(nx, ny);
        }
    }
}
```

```
int main() {
    cin>>n>>m; //n行m列
    for(int i=1; i<=n; i++) {
        for(int j=1; j<=m; j++) {
            cin>>mp[i][j];
        }
    }
    for(int i=1; i<=n; i++) {
        for(int j=1; j<=m; j++) {
            if(mp[i][j]==1&&vis[i][j]==0){
                vis[i][j]=1; //标记已访问
                dfs(i, j);
                cnt++;
            }
        }
    }
    cout<<cnt;
    return 0;
}
```



疾病预防系统题目链接

OJ网站 6823 疾病预防系统

<https://oj.aicoders.cn/problem/6823>

多少块水洼

【问题描述】

有一块 $N \times M$ 的土地，雨后积起了水，有水标记为 'W'，干燥为 '.'。八连通的积水被认为是连接在一起的。请求出院子里共有多少水洼？

【输入格式】

第一行为 N, M ($1 \leq N, M \leq 100$)。

从第二行开始是一个 $n \times m$ 的矩阵。

【输出格式】

一行，共有的水洼数。

【输入样例】

```
7 9
W.....W
.WWWW...W
...WW..W
.....W
.....W
..W....W
.W.W...W
```

【输出样例】

```
3
```

问题分析

每块积水可以与周围八个方向的积水相连。
回顾一下八个方向的存储。
假设当前点($x=3, y=3$)

	1	2	3	4	5
1					
2		⑥	⑦	⑧	
3		⑤	x,y	①	
4		④	③	②	
5					

- ①右 $x+0, y+1$
- ②右下 $x+1, y+1$
- ③下 $x+1, y+0$
- ④左下 $x+1, y+(-1)$
- ⑤左 $x+0, y+(-1)$
- ⑥左上 $x+(-1), y+(-1)$
- ⑦上 $x+(-1), y+0$
- ⑧右上 $x+(-1), y+1$

```
int dx[8]={0,1,1,1,0,-1,-1,-1};
int dy[8]={1,1,0,-1,-1,-1,0,1};
```

dx存储x的加数
dy存储y的加数

问题分析

$n*m$ 的土地有两种字符，有积水记为 'W'，干燥为 '.'。
此题求八个方向W表示的连通块数量。

	1	2	3	4	5	6	7	8	9
1	W	.	.	.	.	.	.	.	W
2	.	W	W	W	.	.	.	.	W
3	.	.	.	.	W	W	.	.	W
4	.	.	.	.	.	.	.	.	W
5	.	.	.	.	.	.	.	.	W
6	.	.	W	.	.	.	.	.	W
7	.	W	.	W	.	.	.	.	W

解题步骤

1. 创建需要的数组及变量。

```
char mp[101][101]; //地图
int vis[101][101]; //标记数组
int n,m; //n行m列
```

2. 主函数内读入数据

```
cin>>n>>m; //n行m列
for(int i=1;i<=n;i++) {
    for(int j=1;j<=m;j++) {
        cin>>mp[i][j];
    }
}
```

	1	2	3	4	5	6	7	8	9
1	W	.	.	.	.	.	.	.	W
2	.	W	W	W	.	.	.	.	W
3	.	.	.	.	W	W	.	.	W
4	.	.	.	.	.	.	.	.	W
5	.	.	.	.	.	.	.	.	W
6	.	.	W	.	.	.	.	.	W
7	.	W	.	W	.	.	.	.	W

解题步骤

3. 创建dfs()函数，实现深度优先搜索。

```
int dx[8]={0,1,1,1,0,-1,-1,-1};
int dy[8]={1,1,0,-1,-1,-1,0,1};
```

通块模板

```
void dfs(int x,int y){
```

```
    for(int i=0;i<8;i++){①
```

搜索八个方向

```
        int nx=x+dx[i];
```

n*m的地图内

```
        int ny=y+dy[i];
```

可以通行

未搜索过

```
        if(nx>0&&nx<=n&&ny>0&&ny<=m&&mp[nx][ny]!='W'&&vis[nx][ny]==0){②
```

```
            vis[nx][ny]=1;
```

③

标记当前格子搜索过

可以通行

```
            dfs(nx,ny);
```

```
        }
```

```
    }
```

```
}
```

解题步骤

4. 遍历地图，如果是'W'且没有标记哦。

```
int cnt=0; //统计有多少区域
for(int i=1;i<=n;i++){
    for(int j=1;j<=m;j++){
        if(mp[i][j]=='W' && vis[i][j]==0){
            vis[i][j]=1; //标记已访问
            dfs(i,j);
            cnt++;
        }
    }
}
cout<<cnt;
```

搜索周围八个方向相连的格子，标记格子已搜索

遍历完所有格子，输出1表示的连通块数量

	1	2	3	4	5	6	7	8	9
1	√	.	.	.	.	.	.	.	√
2	.	√	√	√	.	.	.	.	√
3	.	.	.	.	√	√	.	.	√
4	.	.	.	.	.	.	.	.	√
5	.	.	.	.	.	.	.	.	√
6	.	.	√	.	.	.	.	.	√
7	.	√	.	√	.	.	.	.	√

完整代码

```
#include <bits/stdc++.h>
using namespace std;
char mp[101][101]; //地图
int vis[101][101]; //标记数组
int n, m, cnt; //n行m列cnt统计有多少个水洼
int dx[8] = {0, 1, 1, 1, 0, -1, -1, -1};
int dy[8] = {1, 1, 0, -1, -1, -1, 0, 1};
void dfs(int x, int y) { //从当前点(x,y)开始深度优先搜索
    for(int i = 0; i < 8; i++) {
        int nx = x + dx[i];
        int ny = y + dy[i];
        if(nx > 0 && nx <= n && ny > 0 && ny <= m &&
            mp[nx][ny] == 'W' && vis[nx][ny] == 0) {
            vis[nx][ny] = 1; //标记已访问
            dfs(nx, ny);
        }
    }
}
```

```
int main() {
    cin >> n >> m; //n行m列
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++) {
            cin >> mp[i][j];
        }
    }
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++) {
            if(mp[i][j] == 'W'
                && vis[i][j] == 0) {
                vis[i][j] = 1; //标记已访问
                dfs(i, j);
                cnt++;
            }
        }
    }
    cout << cnt;
    return 0;
}
```



多少块水洼题目链接

OJ网站 1791 多少块水洼

<https://oj.aicoders.cn/problem/1791>



围成面积

【问题描述】

编程计算由数字1围成的下列图形的面积。面积计算方法是统计数字1所围成的**闭合曲线**中0的数目。**注意：**数据确保最外圈不为1。

【输入格式】

输入一个包含 0 和 1 数字的 10*10 矩阵。

【输出格式】

输出数字1围住的面积（格子之间只能上下左右四个方向相连）。

【输入样例】

```
0000000000
0000111000
0000100100
0000010010
0010001010
0101011010
0100111110
0010000100
0001111100
0000000000
```

【输出样例】

13



问题分析

1. 从最外圈任意一个格子搜索当前连通块，搜索过程标记格子已访问。
2. 遍历矩阵，统计没有被标记且为数字0的格子。

例如:从(1,1)开始搜索标记当前连通块。

[illegible]



解题步骤

1. 创建需要的数组及变量。

```
int mp[11][11]; //地图
int vis[11][11]; //标记数组
```

2. 主函数内读入数据

```
for(int i=1;i<=10;i++) {
    for(int j=1;j<=10;j++) {
        cin>>mp[i][j];
    }
}
```

[illegible]



解题步骤

3. 创建dfs()函数，实现深度优先搜索。

```
int dx[4]={1,0,-1,0}; //下右上  
int dy[4]={0,1,0,-1}; //下右上左
```

连通块模板

此题不需要统计单个连通块的格子数量

```
void dfs(int x,int y){
```

```
    for(int i=0;i<4;i++){ ①
```

搜索四个方向

```
        int nx=x+dx[i];
```

10*10的地图内

```
        int ny=y+dy[i];
```

可以通行

未搜索过

```
        if(nx>0&&nx<=10&&ny>0&&ny<=10&&mp[nx][ny]==0&&vis[nx][ny]==0){ ②
```

```
            vis[nx][ny]=1;
```

③

标记当前格子搜索过

```
            dfs(nx,ny);
```

```
        }
```

```
    }
```

```
}
```



解题步骤

4. 统计数字1所围成的闭合曲线中0的数目。

```
int main(){
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>mp[i][j];
        }
    }
    dfs(1,1); //例如: 从(1,1)开始搜索
    int cnt=0; // 统计0的数目
    for(int i=1;i<=10;i++){
        for(int j=1;j<=10;j++){
            if(mp[i][j]==0&&vis[i][j]==0){
                cnt++;
            }
        }
    }
    cout<<cnt;
    return 0;
}
```

[illegible]



完整代码

```
#include <bits/stdc++.h>
using namespace std;
int mp[11][11]; //地图
int vis[11][11]; //标记数组
int dx[4] = {1,0,-1,0}; //下右上左
int dy[4] = {0,1,0,-1}; //下右上左
void dfs(int x,int y) {
    for(int i=0;i<4;i++) {
        int nx=x+dx[i];
        int ny=y+dy[i];
        if(nx>0&&nx<=10&&ny>0&&ny<=10&&mp[nx][ny]==0
            &&vis[nx][ny]==0){
            vis[nx][ny]=1;
            dfs(nx, ny);
        }
    }
}
```

```
int main() {
    for(int i=1;i<=10;i++){
        for(int j=1;j<=10;j++){
            cin>>mp[i][j];
        }
    }
    dfs(1,1);
    int cnt=0; // 统计0的数目
    for(int i=1;i<=10;i++){
        for(int j=1;j<=10;j++){
            if(mp[i][j]==0
                &&vis[i][j]==0){
                cnt++;
            }
        }
    }
    cout<<cnt;
    return 0;
}
```



拓展

数据最外圈可以是1的情况。

[illegible]



围成面积题目链接

OJ网站 1353 围成面积

<https://oj.aicoders.cn/problem/1353>