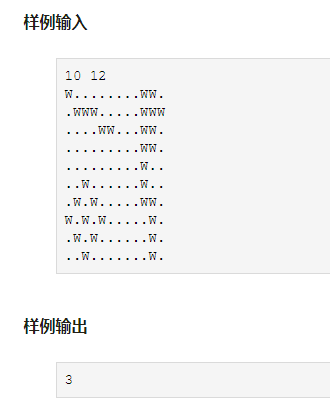
深度优先搜索：

这种算法就是建立在递归之上的，大体思路就是：找到最深处，返回，找第二种，再返回……直到找完，解决得了很多迷宫问题（不要用此算法尝试数据大得恶心的最优解题，否则会严重超时）

广度优先搜索：

顾名思义，从一个点找到他所有可以到的点，入队，即“一层一层”地找下一位置，将head移至下一个（弹出），再找head能去的所有点，入队，步数++……直到无处可找，当然，和深搜一样，要标记找过的点，且要判断此点是否出界或遇到墙，最后的节点的步数即是最短的路径了（对比深搜，优势就在很快（特别是最优解）），如果要输出路径的话，加一个father，最后递归输出就行了~

找有多少个水塘（"W"代表有水，"."代表干旱），共8个方向（哪八个不解释……）都算作一个水塘。



思路：

1. 深搜：查每个点，只要是“W”，就开始dfs，把所有和它相连的“W”变为“.”，sum++，最后输出sum。

#include<cstdio>

char s[102][102];

int n,m;

int f[8][2]={{1,0},{-1,0},{0,1},{0,-1},{1,1},{1,-1},{-1,1},{-1,-1}};//八个方向

int sum;

void w(int x,int y)//深搜

{

if(x<0||y<0||x==n||y==m||s[x][y]=='.')//边界条件

return;

s[x][y]='.';//标记

for(int i=0;i<8;i++)

w(x+f[i][0],y+f[i][1]);//查找下一个

}

int main()

{

scanf("%d%d",&n,&m);

for(int i=0;i<n;i++)

scanf("%s",s[i]);//读入

for(int i=0;i<n;i++)//这里要看你是从哪里开始的

for(int j=0;j<m;j++)//注意这里一定要从0开始，从一开始也行，参看BFS

if(s[i][j]=='W')//找到“W”

{

w(i,j);//其实它的作用是将“W”所在的池塘全部变为干旱地

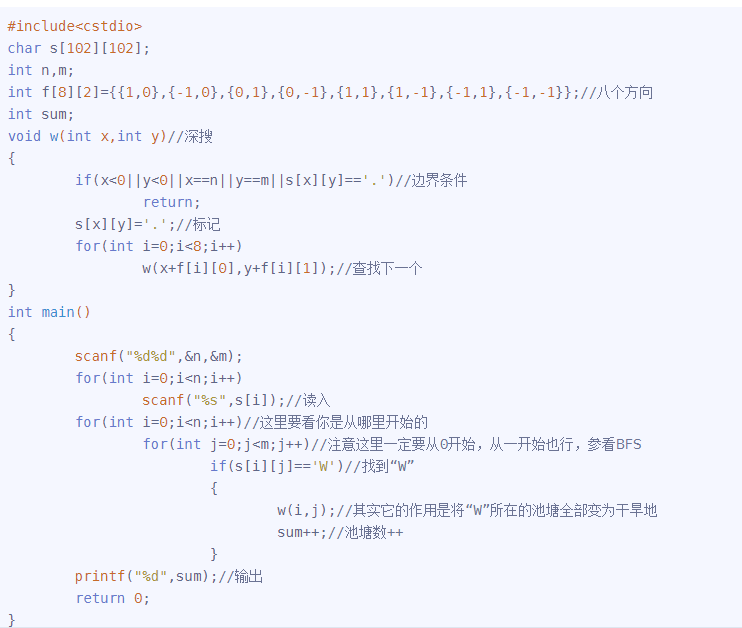
sum++;//池塘数++

}

printf("%d",sum);//输出

return 0;

}



2.广搜：同样查每个点，是要是“W”，就开始bfs，将每个和它有相连的入队，找下一个，每找一个便将它变为“.”，sum++，最后输出sum。

#include<cstdio>

#include<cstring>

struct queue

{

int x,y;//结构体模拟队列，存坐标

}q[10005];

int n,m,head=1,tail=1,sum;//定义头和尾，头事实上指的是尾的父亲，而尾指的是当前位置

int f[8][2]={{0,1},{0,-1},{1,0},{-1,0},{1,1},{-1,-1},{1,-1},{-1,1}};

char map[105][105];

void bfs(int x,int y)

{

q[tail].x=x; q[tail].y=y; tail++; map[x][y]='.';//初始化，将第一个坐标入队

while(head<tail)//头小于尾时说明队列不为空

{

for(int i=0;i<8;i++)

{

int \_x,\_y;

\_x=q[head].x+f[i][0];

\_y=q[head].y+f[i][1];//新坐标

if(\_x>n||\_x<1||\_y>m||\_y<1)//边界

continue;

if(map[\_x][\_y]=='W')

{

map[\_x][\_y]='.';//标记

q[tail].x=\_x;

q[tail].y=\_y;//入队

tail++;//尾移向下一个

}

}

head++;//注意头一定要++，因为这里和头有关系的已经枚举完了，该枚举下一个了

}

sum++;//统计chitangshu

memset(q,0,sizeof(q));//清零（应该可以不用）

tail=1; head=1;

}

int main()

{

scanf("%d%d",&n,&m);

for(int i=1;i<=n;i++)

scanf("%s",map[i]+1);//读入，“+1”指往后一位存，数组从1下标开始存

for(int i=1;i<=n;i++)

for(int j=1;j<=m;j++)//所以这里可以从1开始

if(map[i][j]=='W')

bfs(i,j);

printf("%d",sum);//输出

}