



北京大学
PEKING UNIVERSITY

北京大学暑期课 《ACM/ICPC竞赛训练》

北京大学信息学院 郭炜

guo_wei@PKU.EDU.CN

<http://weibo.com/guoweiofpku>

课程网页: http://acm.pku.edu.cn/summerschool/pku_acm_train.htm



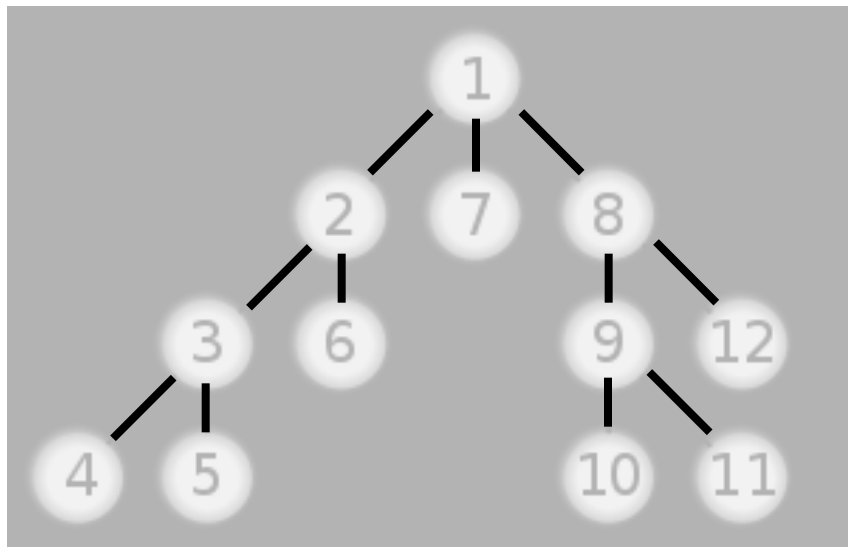
北京大学
PEKING UNIVERSITY

深度优先搜索

入门：城堡问题

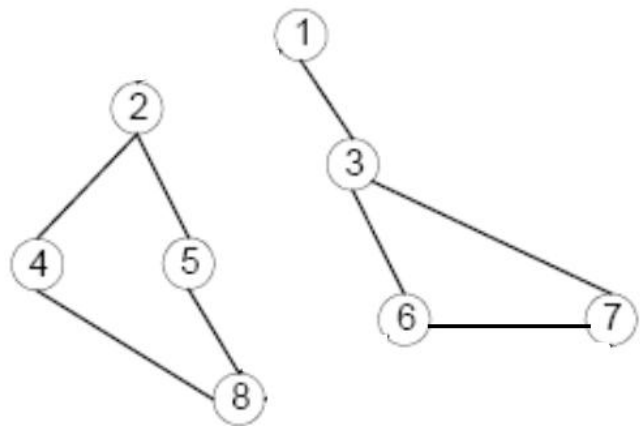
深度优先搜索 (Depth-First-Search)

- 将整个问题空间表示为一个图。
- 从图中某顶点 v 出发：
 - (1) 访问顶点 v ；
 - (2) 依次从 v 的未被访问的邻接点出发，对图进行[深度优先遍历](#)；直至图中和 v 有路径相通的顶点都被访问；
 - (3) 若此时图中尚有顶点未被访问，则从一个未被访问的顶点出发，重新进行[深度优先遍历](#)，直到图中所有顶点均被访问过为止。



将问题的各状态之间的转移关系描述
为一个图, 则深度优先搜索遍历整个图的
框架为:

```
Dfs(v) {  
    if( v 访问过)  
        return;  
    将v标记为访问过;  
    对和v相邻的每个点 u : dfs(u);  
}  
  
int main() {  
    while(在图中能找到未访问过的点 k)  
        Dfs(k);  
}
```

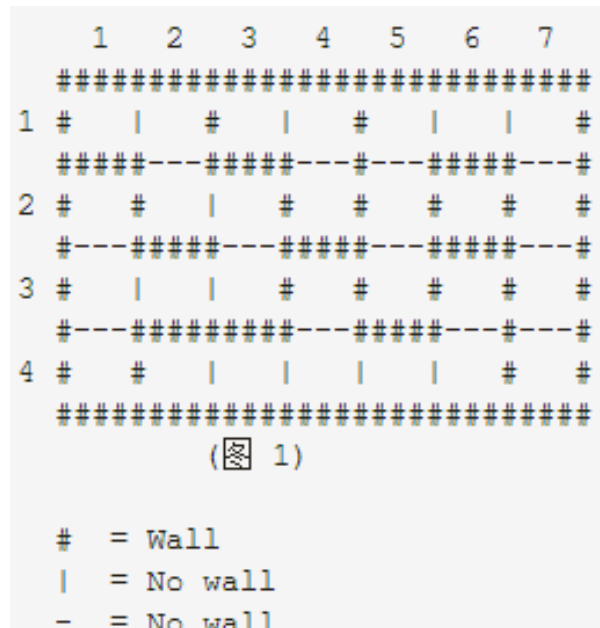


2-4-8-5

1-3-6-7

例题：百练2815 城堡问题

- 右图是一个城堡的地形图。请你编写一个程序，计算城堡一共有多少房间，最大的房间有多大。城堡被分割成 $m \times n$ ($m \leq 50$, $n \leq 50$) 个方块，每个方块可以有0~4面墙。



输入输出

- 输入

- 程序从标准输入设备读入数据。
- 第一行是两个整数，分别是南北向、东西向的方块数。
- 在接下来的输入行里，每个方块用一个数字($0 \leq p \leq 50$)描述。用一个数字表示方块周围的墙，1表示西墙，2表示北墙，4表示东墙，8表示南墙。**每个方块用代表其周围墙的数字之和表示。**城堡的内墙被计算两次，方块(1,1)的南墙同时也是方块(2,1)的北墙。
- 输入的数据保证城堡至少有两个房间。

- 输出

- 城堡的房间数、城堡中最大房间所包括的方块数。
- 结果显示在标准输出设备上。

- 样例输入

4

7

11 6 11 6 3 10 6

7 9 6 13 5 15 5

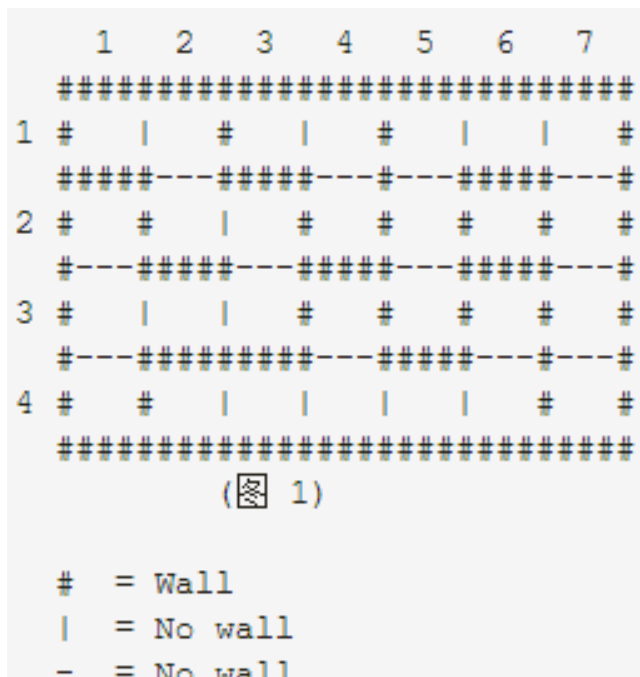
1 10 12 7 13 7 5

13 11 10 8 10 12 13

- 样例输出

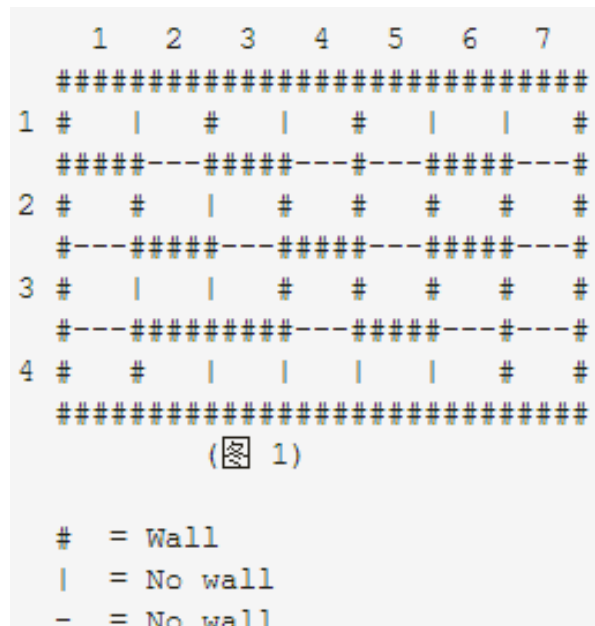
5

9



解题思路

- 对每一个房间，深度优先搜索，从而给这个房间能够到达的所有位置染色。最后统计一共用了几种颜色，以及每种颜色的数量。
 - 比如
- | | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 3 | 3 |
| 1 | 1 | 1 | 2 | 3 | 4 | 3 |
| 1 | 1 | 1 | 5 | 3 | 5 | 3 |
| 1 | 5 | 5 | 5 | 5 | 5 | 3 |
- 从而一共有5个房间，最大的房间（1）占据9个格子




```
#include <iostream>
#include <stack>
#include <cstring>
using namespace std;

int R,C;  //行列数
int rooms[60][60];
int color[60][60]; //房间是否染色过的标记
int maxRoomArea = 0, roomNum = 0;
int roomArea;
void Dfs(int i,int k) {
    if( color[i][k] )
        return;
    ++ roomArea;
    color [i][k] = roomNum;
    if( (rooms[i][k] & 1) == 0 ) Dfs(i,k-1); //向西走
    if( (rooms[i][k] & 2) == 0 ) Dfs(i-1,k); //向北
    if( (rooms[i][k] & 4) == 0 ) Dfs(i,k+1); //向东
    if( (rooms[i][k] & 8) == 0 ) Dfs(i+1,k); //向南
}
```

```

int main()  {
    cin >> R >> C;
    for( int i = 1;i <= R;++i)
        for ( int k = 1;k <= C; ++k)
            cin >> rooms[i][k];
    memset(color,0,sizeof(color));
    for( int i = 1;i <= R; ++i)
        for( int k = 1; k <= C; ++ k) {
            if( !color[i][k] ) {
                ++ roomNum ;    roomArea = 0;
                Dfs(i,k);
                maxRoomArea =
                    max(roomArea,maxRoomArea) ;
            }
        }
    cout << roomNum << endl;
    cout << maxRoomArea << endl;
}

```

```

void Dfs(int r,int c) { //不用递归，用栈解决，程序其他部分不变
    struct Room { int r,c; Room(int rr,int cc):r(rr),c(cc) { } };
    stack<Room> stk;
    stk.push(Room(r,c));
    while ( !stk.empty() ) {
        Room rm = stk.top();
        int i = rm.r; int k = rm.c;
        if( color[i][k]) stk.pop();
        else {
            ++ roomArea;
            color [i][k] = roomNum;
            if((rooms[i][k]&1)==0) stk.push(Room(i,k-1)); //西
            if((rooms[i][k]&2)==0) stk.push(Room(i-1,k)); //北
            if((rooms[i][k]&4) == 0) stk.push(Room(i,k+1)); //东
            if((rooms[i][k]&8) == 0) stk.push(Room(i+1,k)); //南
        }
    }
}

```