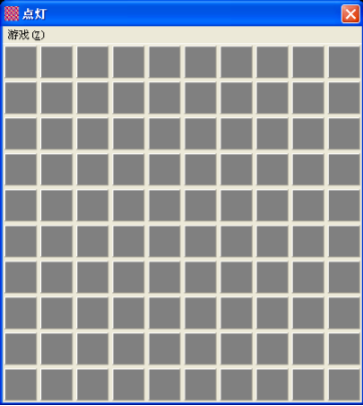
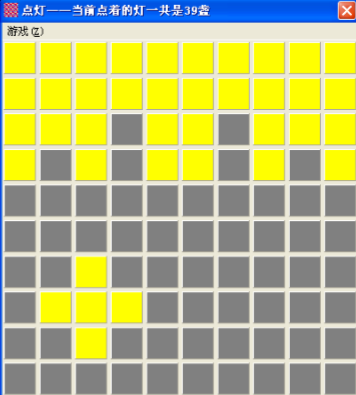
Problem 1084: 地板游戏

Time limit: 1 Seconds Memory limit: 32768K

问题描述：

地板游戏又称点灯游戏。游戏规则：如下图所示，由 n×m 盏灯组成的矩阵中，灯亮则显示为黄色， 否则为灰色。点击其中任意一盏灯，该盏灯及其上下左右的灯如果原来是亮着的，则熄灭；如果原来是熄 灭的，则点亮。游戏目的：要将所有的灯都点亮。初始时，所有灯都是熄灭的。 给定 n 和 m 的值，计算至少需要点击多少盏灯，才能使得所有灯都点亮。

输入描述：

输入文件中包含多个测试数据。每个测试数据占一行，为两个正整数 n 和 m，2≤n,m≤10。输入文件 最后一行为 0 0，代表输入结束。

输出描述：

对输入文件中的每个测试数据，首先输出一个正整数，表示需要点击的最小次数。

样例输入：

4 4

4 5

0 0

样例输出：

4

10

题目分析：

1. 我们要通过枚举每种解法并记录步数然后得出最优解的步数
2. 如何枚举每种解法：
   1. 从表面来看，灯阵由2^(m\*n)种状态，但实际上很多种状态都是由第一行的初始状态决定，因为一旦第一行的状态确定，第二行如何点击就确定了，一次类推，我们只需要检查最后一行的状态是否全部熄灭就可以了。所以我们只需要去枚举第一行的所有状态即可，由2^m种状态。
3. 如何枚举第一行的状态：
   1. 每盏灯的状态无非两种：开 | 关 ，所以我们可以用1和0分别代表开与关的状态，进行枚举。所以第一行的状态转换为计算机语言即为多个0、1相间的串，所以我们可以用二进制枚举的方法，来枚举第一行的所有状态。
   2. 什么是二进制枚举，如何使用：
      1. 每个数字在计算机中都以二进制的方式保存，每个数字都有唯一对应的二进制数码，所以我们可以通过枚举十进制的数字，譬如1~9，然后运用位运算取出每一个1的位置，其余位置默认为零，如此我们就得到一种状态。
      2. 如何取出每一个1的位置，我们借助位运算表达式if(z&(1<<y))，这个式子所代表的含义是：在第Z中情况下，从右往左数的第y个数字是否为零，通过m次(灯阵的列数)循环，我们就能够知道十进制数字z的二进制数的1的位置，从而知道这个状态中哪些灯应该是亮着的。
4. 如何计算步数i与最优解步数i\_least：
   1. 设置一个全局变量i ，每次按下灯都自增1，然后在每种状态最后在进行i与i\_least的比较。
5. 如何表示灯阵，设置一个m\*n的布尔数组，bool lights[m][n]
6. 按下之后如何反转：反转=1-初始 ，但是因为每次反转都涉及到上下左右的反转，所以可能会导致数组的越界，比如在[4][0]上面的左边反转的话就是1-[4][1-0]，我们知道，数组中是不存在-1的下标的，所以越界了，尽管可以通过几个if判断来解决，但最简单的方法还是将原始的数组扩大一圈，都从1开始计数，这样不仅解决了数组越界问题，还使整个代码更加好读了。只是有一点要注意，就是在二进制枚举的位运算表达式要稍作修改，变为 if(z&(1<<(y-1)))，因为y是从1开始计数的。

代码：

// cqjtu\_OJ\_1084.cpp 地板游戏

#include<iostream>

#include<math.h>

using namespace std;

bool lights[20][20]; // 因为每次测试数据的数组大小不同，所以无法精确给定义，但是总的而言是小于10的，所以直接定义一个10\*10的数组

int i,least\_i; // i表示当前方法的步数，least\_i表示最优解的步数

int m,n; // 测试数据输入的行数和列数

void press(int x,int y)

{

lights[x][y] = 1 - lights[x][y]; //本身状态求反

lights[x-1][y] = 1 - lights[x-1][y]; //4个相邻位置状态也求反 // 将上边的数求反

lights[x][y-1] = 1 - lights[x][y-1]; // 将左边的数求反

lights[x+1][y] = 1 - lights[x+1][y]; // 将下边的数求反

lights[x][y+1] = 1 - lights[x][y+1]; // 将右边的数求反

i++;

}

int main()

{

int x,y,z;

while (cin>>n>>m)

{

if(m==0&&n==0) break;

least\_i=9999;

for(z=0;z<pow(2,m);z++) //我又犯傻了，根式计算要有math.h才行

{

i=0;

for(x=1;x<=n;x++)

{for(y=1;y<=m;y++)

lights[x][y]=1;}

for(y=1;y<=m;y++)

if(z&(1<<(y-1)))

press(1,y);

for(x=2;x<=n;x++)

for(y=1;y<=m;y++)

if(lights[x-1][y]==1) // 上一行的灯亮（等于1），则要熄掉它

press(x,y); //每次按下灯步数都要自增一，用于得出最优解

for(y=1;y<=m;y++)

if(lights[n][y]==1) break; //判断最后一行灯是否全灭

if(y>=m+1) //if后面要检查有没有不小心带上分号！！！

if(i<least\_i) least\_i=i;

}

cout<<least\_i<<endl;

}

}

教材的归纳：

1. 从题目归纳的结论：
   1. 因为按下两次按钮会使结果抵消，因此，每个按钮最多按一次
   2. 各个按钮按下的顺序对最终结果无影响
   3. 对第1行中每盏点亮的灯，按下第2行对应的按钮，就可以熄灭第一行的灯。如此重复，可以熄灭第1~4行全部的灯。同样，按下第1~5列的按钮，可以熄灭前5列的灯。
2. 实现技巧：
   1. 将数组长度增大一层，可以默认周围一圈也有按钮且都没有被按下，这样可以方便处理边界问题。
   2. 枚举第1行按钮的状态时，可以将其当成二进制数处理，使用二进制数分解获得不同的状态。例如，整数5的二进制数表示是000101，那么从右向左看可以认为第1，3盏灯处于打开状态，其他灯处于关闭状态。可以知道每个整数的二进制分解是唯一的，这样就可以枚举每一种状态，只需枚举000000 ~111111即可，也就是0 ~ 2^6-1。(此题跟教材数据上有出入，但思路相同)

做这个题的心路历程：

刚开始的时候完全不知道怎么枚举，尽管之前做过相似的题，但是这题要我求出最优解，让我有点懵，因为之前那个题是证明存在即可，后来发现只需要更改循环条件就可以了，后来做出来了，感觉还可以，但是提交的时候报错，用了好多方法去检验错误，最终发现是因为press函数在有些数据的情况下会越界，所以就将数组增大了一层，同时更改循环条件，但还是错误，而且错误数据更多了，但是数组越界的问题也没有出现了，后来发现是因为随着数组的扩大，尽管循环条件要变，但是二进制枚举的数字不能发生更改，这是计算上的问题，所以又接着修改了一下二进制枚举的位运算语句，最终终于得到了正确的加过。

在做的过程中还遇到了一些喷饭的小错误，以为iostream中有求次方的运算符^，实际中这个东西需要调用math.h头文件中pow()函数。还有一个就是不小心在if的判断条件后面不小心加上了分号，结果导致判断失效，查了好久才发现这个错误。

这题对于我来说是真的难，但是在做出来了之后还是有成就感，也感觉得到自己的一个自信和能力都提升了一丢丢，加油吧。