信息学奥林匹克竞赛

Olympiad in Informatics

状态压缩动态规划

陈 真

太原市第五中学校

基本概念

对于有些问题，我们需要记录所有元素的状态才足以进行决策，并且每个元素的状态只有较小的K种，这时候我们可以用一个K进制下的数来表示整个元素集合，数的每一位对应集合中一个元素的状态，然后我们可以通过枚举或者搜索所有元素的状态来进行决策，这种DP 方式叫做状态压缩DP。

状压DP 和搜索都针对无法在多项式时间复杂度内求解的问题，而它的时间复杂度通常比搜索要优一些。

A.搜索/枚举状态类

B. 棋盘放子类

C. 棋盘铺牌类

D. 子集枚举类

E. 插头/轮廓线DP

棋盘放子类问题

【题目01】Corn Fields(来源： USACO06NOV)

【问题描述】给出一个$n \times m$的棋盘，一些格子不能放置棋子。求共有多少种放置棋子的方案，使得每一行每一列的任两个棋子间至少有一个空格。

$n \leq 12, m \leq 12$

棋盘放子类问题

【题目01】Corn Fields(来源： USACO06NOV)

使用搜索时间复杂度开销过大。考虑用一个二进制整数代表每一行的状态，用1 表示放了一个棋子，用0 表示没有放棋子，例如下图可以用100100 表示。

我们可以发现:(1) 每一行的状态是类似的(2) 每一行只受上一行影响。设当前为第k 行, 上一行状态为i, 当前行状态为j，我们可以得到一个转移方程：

$$f(0,0)=1,\ f(k,j)=\sum f(k-1,i)$$

复杂度为： $O(n \times 4m)$

限制1：上一行与当前行！(j & i)

限制2：不能放子的格子！(j & u)

限制3：当前行内棋子间！(j & (j << 1)

棋盘放子类问题

【题目01】Corn Fields(来源： USACO06NOV)

```
1   //限制2:不能放子的格子
2   for(int i=1;i<=n;i++){
3       for(int j=1;j<=m;j++){
4           scanf("%d",&x);
5           x=!x;
6           map[i]|=x<<(j-1);
7       }
8   }
9   //限制3:当前行内棋子间
10  for(int i=0;i<=(1<<m)-1;i++)
11      if(!(i&(i<<1)))
12          way[++w]=i;
```

棋盘放子类问题

【题目01】Corn Fields(来源： USACO06NOV)

```
14      f[0][0]=1;
15      for(int k=1;k<=n;k++)
16          for(int i=1;i<=w;i++)
17              for(int j=1;j<=w;j++)
18                  if(!(way[i]&map[k])&&!(way[i]&way[j]))
19                      f[k][way[i]]=(f[k][way[i]]+f[k-1][way[j]])%mod;
20      for(int i=1;i<=w;i++)
21          ans=(ans+f[n][way[i]])%mod ;
```

棋盘放子类问题

【题目02】 sgu little kings

【问题描述】在$N \times N$ 的棋盘里面放K 个国王，使他们互不攻击，共有多少种摆放方案。国王能攻击到它上、下、左、右，以及左上、左下、右上、右下八个方向上离它最近的一个格子，共8 个格子。$N \leqslant 9$

棋盘放子类问题

【题目02】 sgu little kings

上一道题的简单变式，多了攻击范围限制+ 棋子数目限制。

设当前为第k 行, 上一行状态为i, 当前行状态为j，前i 行要放u 个棋子，当前行状态中有*num[j] 个棋子，我们可以得到一个转移方程：*

$$f_{0,0,0} = 1, f_{k,u,j} = \sum f_{k-1,u-num[j],i}$$

复杂度为：*O(n × K × num^2) ≈ 6 × 10^6*

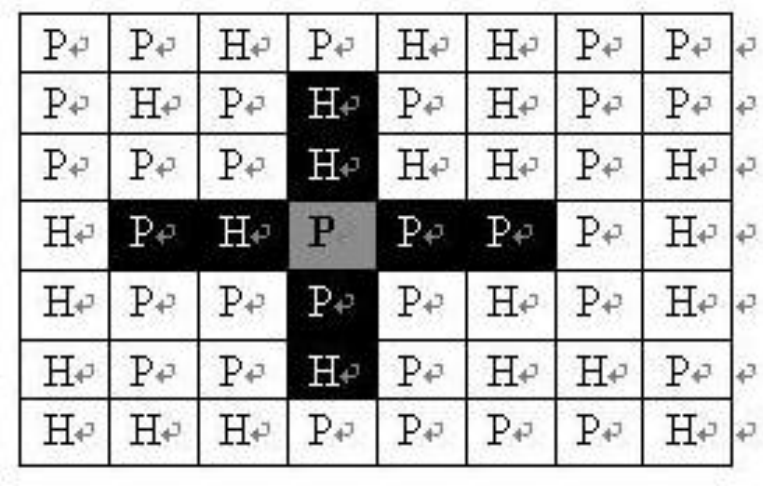
```
1  for(int j=0;j<n;j++)
2      if(i>>j&1) one++;
3  way[++tot]=i;
4  num[tot]=one;
```

限制：上一行与当前行*!(j&i)&&!(j&(i<<1))&&!(j&(i>>1))*

棋盘放子类问题

【题目03】炮兵阵地

【题目描述】司令部的将军们打算在N*M的网格地图上部署他们的炮兵部队。一个N*M的地图由N行M列组成，地图的每一格可能是山地（用"H" 表示），也可能是平原（用"P"表示），如下图。在每一格平原地形上最多可以布置一支炮兵部队（山地上不能够部署炮兵部队）；一支炮兵部队在地图上的攻击范围如图中黑色区域所示：

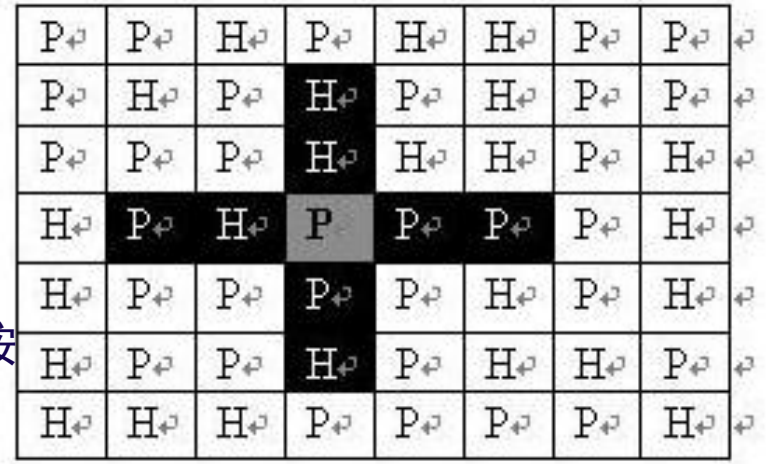


如果在地图中的灰色所标识的平原上部署一支炮兵部队，则图中的黑色的网格表示它能够攻击到的区域：沿横向左右各两格，沿纵向上下各两格。图上其它白色网格均攻击不到。从图上可见炮兵的攻击范围不受地形的影响。

现在，将军们规划如何部署炮兵部队，在防止误伤的前提下（保证任何两支炮兵部队之间不能互相攻击，即任何一支炮兵部队都不在其他支炮兵部队的攻击范围内），在整个地图区域内最多能够摆放多少我军的炮兵部队。

棋盘放子类问题

【题目03】炮兵阵地



【输入格式】

第一行包含两个由空格分割开的正整数，分别表示N和M；

接下来的N行，每一行含有连续的M个字符('P'或者'H')，中间没有空格。按顺序表示地图中每一行的数据。N <= 100；M <= 10。

【输出格式】

仅一行，包含一个整数K，表示最多能摆放的炮兵部队的数量。

【样例输入】

```
5 4
PHPP
PPHH
PPPP
PHPP
PHHP
```

【样例输出】

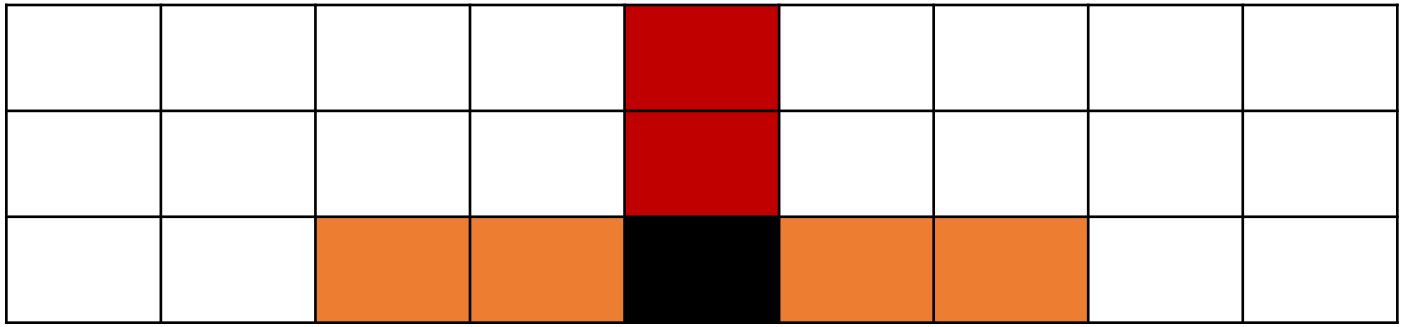
```
6
```

棋盘放子类问题

【题目03】炮兵阵地

思考1：同一行的冲突与否？ 预处理时判定

思考2：上一行与上两行的冲突与否？



思考3：当前行与下两行的冲突与否？

棋盘放子类问题

【题目03】炮兵阵地

第一道题的简单变式，多了攻击范围限制+ 求最多放置

设当前为第k 行, 上上行状态为u, 上一行状态为j，当前行状态为i, 当前行状态中有num[i] 个棋子，我们可以得到一个转移方程：

$$f_{0,0,0} = 0, f_{k,i,j} = \max(f_{k-1,j,u} + num[i])$$

复杂度为：`O(n * num^3)`

限制：当前行之间`!(i&(i<<1))&&!(i&(i<<2))`

棋盘放子类问题

【题目04】[SCOI2005]互不侵犯

【问题描述】在N×N的棋盘里面放K个国王，使他们互不攻击，共有多少种摆放方案。国王能攻击到它上下左右，以及左上左下右上右下八个方向上附近的各一个格子，共8个格子。

注：数据有加强（2018/4/25）

【输入格式】只有一行，包含两个数N，K （ 1 <=N <=9, 0 <= K <= N * N）

【输出格式】所得的方案数

【输入样例】

3 2

【输出样例】16

棋盘放子类问题

【题目04】[SCOI2005]互不侵犯

【步骤1】：统计每一行放置的国王的数量

统计 十进制的二进制表示下有多少个1.

【步骤2】：攻击上下左右等八个方向的国王

当前行的合法性： x &（x<<1）==0 x &(x>>1)==0

两行间的合法性： x & y x&（y<<1） x & （y>>1）

第x行与第y行

【步骤3】：dp方程的设定？？？？

棋盘放子类问题

【题目05】[USACO06NOV]Corn Fields G

【问题描述】农场主John新买了一块长方形的新牧场，这块牧场被划分成M行N列(1 ≤ M ≤ 12; 1 ≤ N ≤ 12)，每一格都是一块正方形的土地。John打算在牧场上的某几格里种上美味的草，供他的奶牛们享用。
遗憾的是，有些土地相当贫瘠，不能用来种草。并且，奶牛们喜欢独占一块草地的感觉，于是John不会选择两块相邻的土地，也就是说，没有哪两块草地有公共边。
John想知道，如果不考虑草地的总块数，那么，一共有多少种种植方案可供他选择？（当然，把新牧场完全荒废也是一种方案）

【输入格式】第一行：两个整数M和N，用空格隔开。
第2到第M+1行：每行包含N个用空格隔开的整数，描述了每块土地的状态。第i+1行描述了第i行的土地，所有整数均为0或1，是1的话，表示这块土地足够肥沃，0则表示这块土地不适合种草。

【输出格式】一个整数，即牧场分配总方案数除以100,000,000的余数。

【输入样例】【输出样例】9

```
2 3
1 1 1
0 1 0
```

棋盘放子类问题

【题目05】[USACO06NOV]Corn Fields G

步骤1： 二进制的地图每行如何表示为十进制 。。。

步骤2: 判断某一行是否为合法状态

```
g[i]=(!(i&(i<<1)) && (!(i&i>>1)))
```

步骤3: 枚举不同的状态　检测状态的合法性

步骤4: dp的表示含义？？？

棋盘放子类问题

模型特点：每行状态类似（可预处理）、棋子有攻击范围、可能存在不可放置的格子

目标类型：方案数、特定棋子时方案数（多一维）、最多棋子数

转移方程：由前一（多）行的状态整体向当前行转移

限　　制：二进制下左右移+ 按位与解决（寻找冲突）

时　　间：预处理后运用公式计算

空　　间：由于状态转移通常只涉及几行，可以使用滚动数组优化

棋盘铺牌类问题

【题目06】 Mondriaan’s Dream

【问题描述】给出*n×m 的棋盘，用1×2 的长方形骨牌不重叠地覆盖这个棋盘，求覆盖满的方案数。*

$n \le 11, m \le 11$

棋盘铺牌类问题

【题目06】 Mondriaan’s Dream

显然，n,m 都是奇数时无解

转移方式类似, 设当前为第k 行, 上一行状态为j，当前行状态为i，把前k-1 行填满，我们可以得到一个转移方程：

$$f_{0,2^m-1} = 1, f_{k,i} = \sum f_{k-1,j}$$

复杂度为：*O(n × 2^m) – O(n × 3^m)*

限　　制：牌之间不叠放

目标状态：铺满

棋盘铺牌类问题

【题目07】 Mondriaan’s Dream

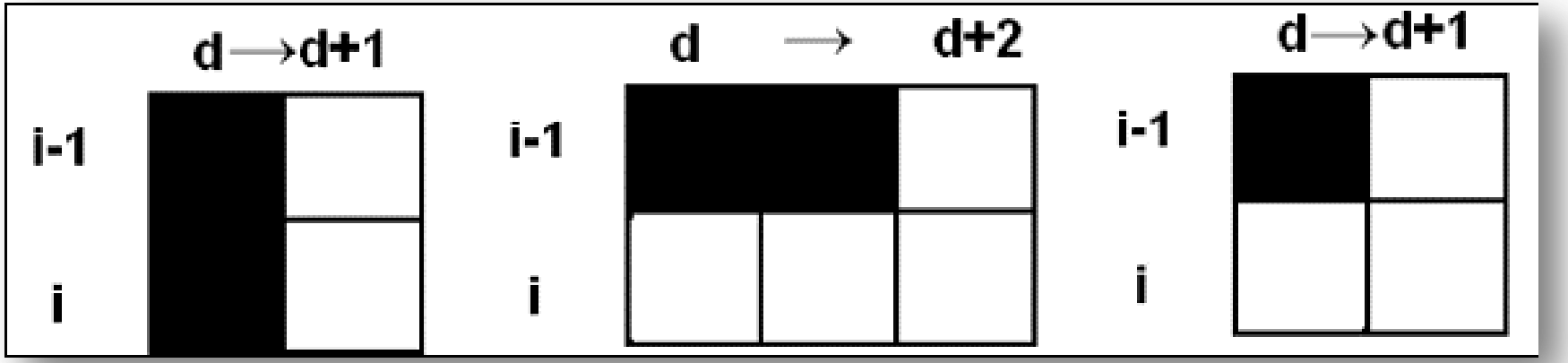
用1 表示横着放和竖着放的第二个，0 表示竖着放的第一个和不放

对于当前行的状态，它是由前一行的状态转化过来的，对于该行某个位置：

如果前一行该位置为0, 那么该位置可以竖放即0-> 1

如果前一行连续两个位置为1, 那么这两个连续位置可以横放即11-> 11

如果前一行该位置为1, 那么该位置可以留空即1-> 0



棋盘铺牌类问题

【题目07】 Mondriaan’s Dream

转　移

```
3  void dfs ( int l,int las,int noww) {
4      if(l>m) return;
5      if (l==m) {
6          f[i][noww]+=f[i.1][las];
7          return;
8      }
9      dfs(l+1,las<<1,noww<<1|1) ; //0 1
10     dfs(l+2,las<<2|3,noww<<2|3) ; // 11 11
11     dfs(l +1,las<<1|1,noww<<1) ; //1 0
12 }
13 f[0][(1<<m).1]=1;
14 for(i=1;i<=n;i++)
15     dfs(0,0,0) ;
16 printf( "%lld\n", f[n][(1<<m).1]);
```

棋盘铺牌类问题

【题目07】 Mondriaan’s Dream

记忆化搜索

不依赖外部变量，答案直接通过转移记录在数组中，无后效性

和递推同样为 DP 的表现形式

优点：相对递推不需要考虑转移顺序，思考难度较低，判断条件较少，相对搜索减 少了需要重复计算的内容

缺点：耗时较长（不一定

棋盘铺牌类问题

【题目08】 Hardwood Floor

【问题描述】给出 n×m 的棋盘，用 1×r 的长方形骨牌和 L 形的（2×2 的去掉一个角）骨牌不重叠地 覆盖这个棋盘，求覆盖满的方案数。 n ≤ 10，m ≤ 1

棋盘铺牌类问题

【题目08】 Hardwood Floor

```
void dfs(int l,int las,int noww,int b1,int b2){
    if(l==m){if(!b1&&!b2)f[i][noww]+=f[i-1][las];return;}
    if(!b1&&!b2){
        dfs(l+1,las<<1,noww<<1|1,0,0); // 10 10
        dfs(l+1,las<<1,noww<<1|1,1,0); // 11 10
        dfs(l+1,las<<1,noww<<1|1,0,1); // 10 11
    }
    if(!b1)
        dfs(l+1,las<<1,noww<<1|b2,1,1); // 11 01
    // 没有11 00
    if(!b2){
        dfs(l+1,las<<1|(1-b1 ),noww<<1|1,1,1); // 01 11
        dfs(l+1,las<<1|(1-b1 ),noww<<1|1,0,1); // 00 11
    }
    dfs(l+1,las<<1|(1-b1),noww<<1|b2,0,0); // 00 00
}
```

棋盘铺牌类问题

【题目09】棋盘覆盖

【问题描述】给出 n×m 的棋盘，用 1×r 的长方形骨牌不重叠地覆盖这个棋盘，求覆盖满的方案数。 n ≤ 9, m ≤ 9, r ∈ [2, 5], r 为给定数。

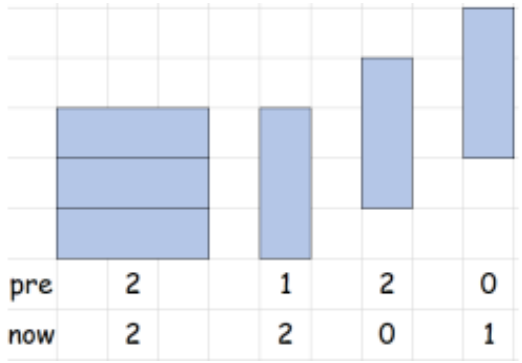
棋盘铺牌类问题

【题目09】棋盘覆盖

保存多行状态情况较多，空间开销较大，且不具有普适性 考虑用两个 r 进制整数代表每 r 行的状态，设当前为第 k 行，我们保证第 k-r+1 行 填满了，我们可以得到 DFS 转移：

```
2       int s1=las*r,s2=noww*r;
3       for(int i=0;i<=r-1;i++)
4           dfs(h,l+1,s1+(i+1)%r,s2+i);
5       dfs(h,l+r,las-pre[r]+pre[r]-1,noww-pre[r]+pre[r]-1);
```

r=3 时一种可能的几何含义



棋盘铺牌类问题

模型特点：每行状态类似、棋子有不可重叠覆盖范围、可能存在不可放置的格子

目标类型：方案数 **转移方程：**由前一（多）行的状态整体向当前行转移

限　　制：由于情况比放子类问题要多，常使用记忆化搜索直接构造合法情况

时　　间：估算/大数据测量

空　　间：由于状态转移通常只涉及几行，有时可以使用滚动数组优化

搜索/枚举状态类问题

【题目06】种花小游戏

【问题描述】植物大战僵尸这款游戏中，还有个特别有意思的赚钱方式——种花（能长金币的花）。玩家可以购买一只蜗牛来帮助捡金币，蜗牛总是喜欢以最少的行程来捡走所有的金币。 现在告诉你场上 n 个金币所在位置的坐标，以及蜗牛所在位置，请你求出蜗牛捡走所有 金币的最小行程。

n ⩽ 16

搜索/枚举状态类问题

【题目06】种花小游戏

【问题描述】一道体现记忆化搜索特点的题

普通搜索 O(n!) 开销太大，所以改写为记忆化搜索

这里我选择了从末状态往回搜，其实从起点开始也可以，关键在于不起眼的判断

```
1  double dfs(int x,int noww){
2      if(f[x][noww]<1e15 ) return f[x][noww];
3      int nxt=noww-(1<<(x-1));
4      for(int i=1;i<=n;i++)
5          if(nxt&(1<<(i-1)))
6              f[x][noww]=min(f[x][noww],dfs(i,nxt)+dist[x][i]);
7      return f[x][noww];
8  }
```

搜索/枚举状态类问题

【题目07】拓扑排序方案数

【问题描述】面馆特派员还有 N 件事情没有做完，做这些事情还有 M 个限制条件，每个限制条件均为：事件 A 必须在事件 B 之前完成。现在 SF 面馆特派员想知道，他做完这 N 件事一 共有多少种方法。 n ≤ 17, m ≤ 25

搜索/枚举状态类问题

【题目07】拓扑排序方案数

一个事件只有其前置事件全部被做完时才能被做，利用这个进行转移 设 s 为当前已做完事件集合，i 为待做事件，pre[i] 为 i 的前置事件集合，有转移方程：

$$f[0]=1, f[s|(1\ll(i-1)]=\sum_{s\&pre[i]=pre[i]\ \wedge\ s(1\ll(i-1))=0} f[s]$$

时间复杂度 O(n × 2^n)

搜索/枚举状态类问题

【题目08】愤怒的小鸟

搜索/枚举状态类问题

【题目08】愤怒的小鸟

【问题描述】Kiana 最近沉迷于一款神奇的游戏无法自拔。 简单来说，这款游戏是在一个平面上进行的。有一架弹弓位于（0，0）处，每次 Kiana 可 以用它向第一象限发射一只小鸟，小鸟们的飞行轨迹均为形如 y = ax2 + bx 的曲线，其 中 a，b 是 Kiana 指定的参数，且必须满足 a ＜ 0。当小鸟落回地面（即 x 轴）时，它就 会瞬间消失。 在游戏的某个关卡里，平面的第一象限中有 n 只猪，如果某只小鸟的飞行轨迹经过了猪 所在坐标，那么该猪就会被消灭掉。 现在 Kiana 想知道，至少需要发射多少只小鸟才能消灭所有的猪。 n ≤ 18

搜索/枚举状态类问题

【题目09】[APIO2007] 动物园

【问题描述】圆形动物园坐落于太平洋的一个小岛上，包含一大圈围栏，每个围栏里有一种动物。 有一群小朋友来动物园参观，有的动物有一些小朋友喜欢，有的动物有一些小朋友害怕。 你可以选择将一些动物从围栏中移走以使得小朋友不会害怕。但你不能移走所有的动物， 否则小朋友们就没有动物可看了。每个小朋友站在大围栏圈的外面，可以看到连续的 5 个围栏。你得到了所有小朋友喜欢和害怕的动物信息。当下面两处情况之一发生时，小 朋友就会高兴：至少有一个他害怕的动物被移走，至少有一个他喜欢的动物没被移走 现在请求出至多能让多少个小朋友高兴。（具体信息查看原题面）

搜索/枚举状态类问题

【题目04】关灯问题II（洛谷 P2622）

【题目描述】现有n盏灯，以及m个按钮。每个按钮可以同时控制这n盏灯——按下了第i个按钮，对于所有的灯都有一个效果。按下i按钮对于第j盏灯，是下面3中效果之一：如果a[i][j]为1，那么当这盏灯开了的时候，把它关上，否则不管；如果为-1的话，如果这盏灯是关的，那么把它打开，否则也不管；如果是0，无论这灯是否开，都不管。
现在这些灯都是开的，给出所有开关对所有灯的控制效果，求问最少要按几下按钮才能全部关掉。

【输入格式】前两行两个数，n m
接下来m行，每行n个数,a[i][j]表示第i个开关对第j个灯的效果。

【输出格式】一个整数，表示最少按按钮次数。如果没有任何办法使其全部关闭，输出-1

【输入样例】【输出样例】　　【数据范围】对于100%数据 n<=10, m<=100

```
3                2
2
1 0 1
-1 1 0
```

搜索/枚举状态类问题

【题目04】关灯问题II（洛谷 P2622）

步骤1：读取矩阵，a[i][j]

步骤2：目标是将所有的灯关闭（初始时是全部开着的）

11111……　　→　00000……

步骤3：表达三种的不同的操作？怎么表示

步骤4：n盏灯的不同状态进行处理。。。。

子集枚举类问题

【题目09】Hackers’ Crackdown

【问题描述】你需要将 n 个集合分成尽量多组，使得每一组里面所有集合的并集等于全集 1 – n，输 出最多组数。 n ⩽ 16

枚举集合 s 的所有子集

```
for(int S1=S;S1!=0;S1=(S1-1)&S)
```

子集枚举类问题

【题目03】吃奶酪（洛谷 P1433）

【题目描述】房间里放着 n 块奶酪。一只小老鼠要把它们都吃掉，问至少要跑多少距离？老鼠一开始在 (0,0)点处。

【输入格式】

第一行一个正整数 n。

接下来每行 2 个实数，表示第i块奶酪的坐标。

两点之间的距离公式为

【输出格式】一个数，表示要跑的最少距离，保留 2位小数。

【输入样例】 **【输出样例】7.41**

```
4
1 1
1 -1
-1 1
-1 -1
```

子集枚举类问题

【题目03】吃奶酪（洛谷 P1433）

步骤1：可预处理任意两点之间的距离　d[i][j]

步骤2：不停的将现有的点加入到集合中

dp[i][j] 表示的含义