// 状压 牛可方的方案数

#define rep(i,a,b) for(ll i = a; i<=b ;++i)

#define per(i,a,b) for(ll i = a; i>=b ;--i)

#define maxn 30

/\*

分析：假如知道第 i-1 行的所有的可以放的情况.对于第 i 行的可以放的一种情况

只要判断它和 i - 1 行的所有情况的能不能满足题目的所有牛不相邻

一种放法是最多由12个 0 或者 1 组成的，用二进制的一个数来表示一种放法。

定义状态dp【i】【j】，第 i 行状态为 j 的时候放牛的种数

j 的话我们转化成二进制，从低位到高位依次 1 表示放牛0表示没有放牛，

就可以表示一行所有的情况

那么转移方程 dp【i】【j】=sum（dp【i-1】【k】）

\*/

//x & （x<<1）来判断一个数相邻两位是不是同时为1

//x & y 的布尔值来判断相同为是不是同时为1。

/\*每一行总共2^N种状态，其中删去有相邻的状态，删除不符合给定条件的状态。

保留下最终的所有状态。然后开始dp。

从上往下，每一行之依赖于上一行。

所以从上一行转移过去，把状态数加起来就行了。\*/

const int N = 13;

const int M = 1<<N;

const int mod = 100000000;

int st[M],mp[M]; //分别存每一行的状态和给出地的状态

int dp[N][M]; //表示在第i行状态为j时候可以放牛的种数

bool judge2(int i, int x)//不能放的地方放了

{

//cout << i << x << mp[i] << st[x] << ' ' << (mp[i]&st[x])<< endl;

/\*

10 00 0

11 01 0

12 02 0

13 04 0

14 05 0

20 50 0

21 51 1

22 52 0

23 54 4

24 55 5

\*/

return mp[i]&st[x];

}

int main()

{

int n, m, x;

while( cin >> n >> m )

{

memset(st,0,sizeof(st));

memset(mp,0,sizeof(mp));

memset(dp,0,sizeof(dp));

rep(i, 1, n)

{

rep(j, 1, m)

{

cin >> x;

if( x == 0)

mp[i] += (1<<(j-1));

}

}

//cout << (1<<(j-1)) << mp[i]<<endl;

/\*

2 3

1 1 1 mp[1] = 0

1.0 1 0

1.1 2 0

1.2 4 0

0 1 0 mp[2] = 5

2.0 1 1 //001不可放入

2.1 2 1 //001

2.2 4 5 //101把二进制压缩成一个十进制数字

\*/

int k = 0;

rep(i, 0, (1<<m)-1 )//选出所有相邻不等的状态

if( !( i & i<<1 ) ) //i&(i<<1) 为假说明不存在相邻的情况

st[k++] = i;

//http://blog.csdn.net/became\_a\_wolf/article/details/50592475

//st[0] = 0 [1] = 1 [2] = 2 [3] = 4 [4] = 5 可以放的

rep(i, 0, k-1)

if( !judge2(1, i))

dp[1][i] = 1;

rep(i, 2, n)

{

rep(j, 0, k-1)

{

if(judge2(i,j))//判断第i行 假如按状态j放牛的话行不行。

continue;

rep(f, 0, k-1)

{

if( judge2(i-1, f))//http:blog.csdn.net/became\_a\_wolf/article/details/50592475

continue;//剪枝 判断上一行与其状态是否满足

// 枚举找出 与i-1行的图的 实际情况 匹配的情况

if( !(st[j]&st[f] ) )

//i 行 存在的情况 与 i-1行必须完全不匹配

dp[i][j] +=dp[i-1][f];

}

}

}

int ans = 0;

rep(i, 0, k-1)

{

ans += dp[n][i];

ans %= mod;

}

cout << ans << endl;

}

}