状压前置技能：位运算

位运算

- >>,<<,&,|,^
- 位运算的优先级：
- 位反（~）> 算术 > 位左移、位右移 > 关系运算> 位与 > 位或 > 位异或 > 逻辑
- **可能你记不住这些优先级，所以使用位运算的时候需要加括号**
- 常用操作：
- 与操作，1不变，0变0
- 或操作，0不变，1变1
- 异或操作,0不变，1取反

常规操作

去掉最后一个零	>>1
在最后加一个零	<<1
在最后加一个1	(x<<1)+1
把最后一位变成1	x | 1
把最后一位变成0	x & ~1
最后一位取反	x ^ 1
把右数第K位变成1	x | (1<<(k-1))
把右数第k位变成0	x & ~(1<<(k-1))

常规操作

右数第K位取反	x ^ (1<<(k-1))
把最右面连续的1变成0（从最右开始）	x & (x+1)
把右起第一个0变成1	x | (x+1)
把最右面连续的0变成1（从最右开始）	x | (x-1)
取末尾K位	x &((1<<k)-1)
把末尾K位变成1	x | ((1<<k)-1)
末尾K位取反	x ^ ((1<<k)-1)
取右数第K位	(x>>(k-1)) & 1

其他操作

- 如何保证不能存在两个连续的1？（任意两个1不相邻）
- x&(x<<1)大于0或者x&(x>>1)大于0（不满足条件）

状态压缩动态规划

冯海荣

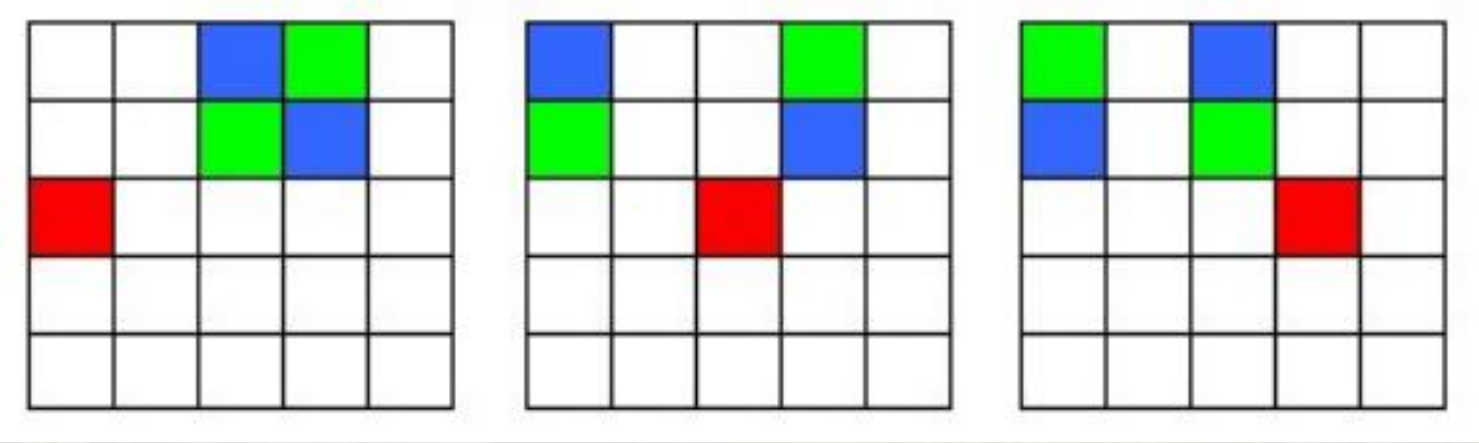
引例

- 在 n*n(n≤20)的方格棋盘上放置n 个车(可以攻击所在行、列)，求使它们不能互相攻击的方案总数。
- 题解：我们一行一行放置，则第一行有*n*种选择，第二行*n*-1，……，最后一行只有1种选择，根据乘法原理，答案就是*n!*

状压思路

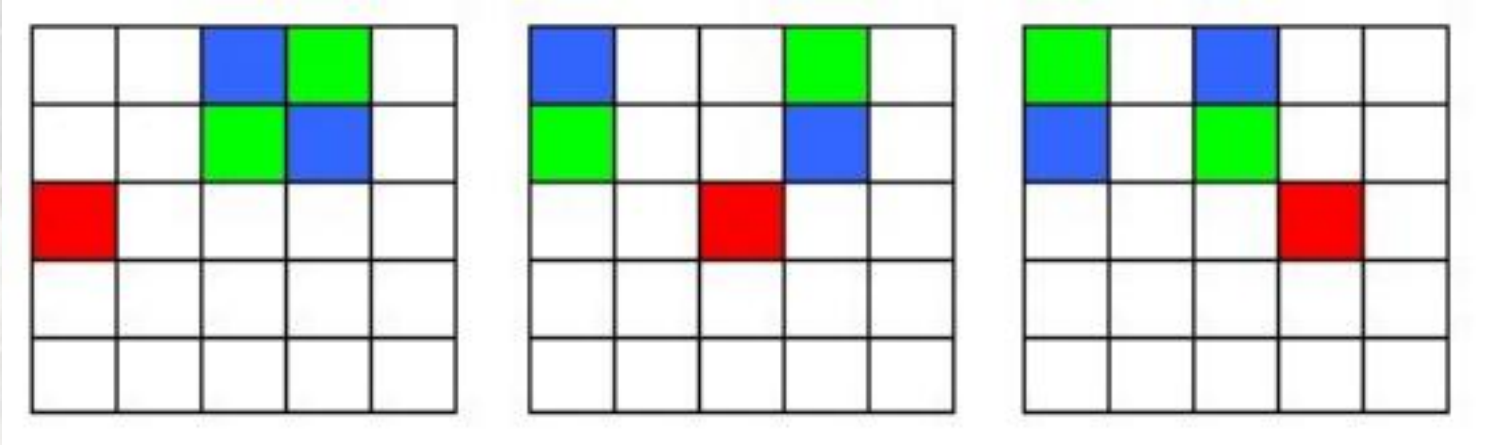
- 既然n<=20，可以状压，如何状压?
- 我们仍然一行一行放置。
- 取棋子的放置情况作为状态，某一列如果已经放置棋子则为1，否则为0。这样，一个状态就可以用一个最多20位的二进制数表示。
- 例如n=5,第1、3、4列已经放置，则这个状态可以表示为01101(**从右到左**)。设f[s] 为达到状态s的方案数，则可以尝试建立f的递推关系。

- 考虑n=5,s=01101
- 因为我们是一行一行放置的，所以当达到s时已经放到了第三行。又因为一行能且仅能放置一个车，所以我们知道状态s一定来自：
  - 前两行在第3、4列放置了棋子(不考虑顺序，下同)，第三行在第1列放置
  - 前两行在第1、4列放置了棋子，第三行在第3列放置
  - 前两行在第1、3列放置了棋子，第三行在第4列放置



- 这三种情况互不相交，且只可能有这三种情况，根据加法原理，f_s应该等于这三种情况的和。写成递推式就是：

$$f_{01101} = f_{01100} + f_{01001} + f_{00101}$$



状态压缩——引例解法

- 根据上面的讨论思路推广之，得到引例的解决办法：

$$f_0 = 1$$

$$f_s = \sum f_{s-2^i}$$

其中s的右起第i+1位为1

(其实就是在枚举s的二进制表示中的1,挨个去掉1,因为当前状态由前一个状态推过来的，010110这个状态说明放置了三个车，但是具体第三次在第三行放的是哪一个1不知道，哪个都有可能，所以**010110的情况=000110+010100+010010**的和，所以就是挨个去掉1)

状态压缩——引例解法

```
int f[ 1 << 20 ];
scanf("%d", &n);

f[0] = 1;
for(i=1;i<(1<<n);++i)
//这个循环的作用是不断寻找二进制中的1
    for(t =i;t>0;t-=(t&-t))//t&-t的作用是找到末尾最后一个1
        f[i]+=f[i&~(t&-t)];
printf("%d\n",f[(1<<n)-1]);
```


状态压缩——引例解法

- 反思这个算法，其正确性毋庸置疑(可以和n!对比验证)
- 但是算法的时间复杂度为$O(n2^n)$，空间复杂度$O(2^n)$，是个指数级的算法
- 当n比较小的时候，我们可以状压

状态压缩——例一

- 在n*n(n≤20)的方格棋盘上放置n个车，某些格子不能放，求使它们不能互相攻击的方案总数。
- 30s思考时间

状态压缩——例一

- 引例的算法是在枚举当前行(即s中1的个数，设为r)的放置位置(即枚举每个1)
- 而对于例1，第r行可能存在无法放置的格子，怎么解决?

枚举1的时候判断一下！

状态压缩——例一

- 事实上，我们并不需要对引例的算法进行太大的改变，只要在枚举s中的1的时候判断一下是否是不允许放置的格子即可
- 然而对于n=20，$O(n2^n)$的复杂度已经不允许我们再进行多余的判断。所以实现这个算法时应该应用一些技巧。
- 我们用a_r表示第r行不允许放置的情况，如果第r行某一位不允许放置，**则a_r此位为0，否则为1**，这可以在读入数据阶段完成

状态压缩——例一

- 然后对于需要处理的状态s，用ts=s&a_r来代替s进行枚举，即不枚举s中的1转而枚举ts中的1。
- 因为ts保证了不允许放置的位为0，这样就可以不用其它的判断来实现算法
- 代码中只增加了计算a数组和r的部分，而时间复杂度没有变化。

状态压缩——例二

- 有一个n*m的棋盘(n、m≤80,n*m≤80)要在棋盘上放p(p≤20)个棋子，使得任意两个棋子不相邻。求合法的方案总数

状态压缩——例二

- 观察题目给出的规模，n、m≤80，这个规模要想用SC是困难的，2^{80}无论在时间还是空间上都无法承受
- 然而我们还看到**n*m≤80**
- 稍微思考我们可以发现：9*9=81>80，即如果n,m都大于等于9，将不再满足n*m≤80这一条件。所以，我们有n或m小于等于8，而2^8是可以承受的！

状态压缩——例二

- 我们假设m≤n，n是行数m是列数，则每行的状态可以用m位的二进制数表示
- 但是本题和例1又有不同：例1每行每列都只能放置一个棋子，而本题却只限制每行每列的棋子不相邻
- 上例中枚举当前行的放置方案的做法依然可行。我们用数组s保存一行中所有的num个放置方案，则s数组可以在预处理过程中求出，同时用c_i保存第i个状态中1的个数以避免重复计算

状态压缩——例二

- 本题状态的维数需要增加，原因在于并不是每一行只放一个棋子，也不是每一行都要求有棋子，原先的表示方法已经无法完整表达一个状态。
- 我们用$f_{i,j,k}$表示第i行的状态为s_j且前i行总共放置k个棋子的方案数。沿用枚举当前行方案的做法，只要当前行的放置方案和上一行的不冲突。就是要两行的状态s_1和s_2中没有同为1的位即可，亦即$\mathbf{s_1 \& s_2 = 0}$

状态压缩——例二

- 然而，虽然我们枚举了第i行的放置方案，但却不知道其上一行(第i-1行)的方案
- 为了解决这个问题，我们不得不**连第i-1行的状态一起枚举**，则可以写出递推式：

$$f_{0,1,0} = 1$$

$$f_{i,j,k} = \sum f_{i-1,p,k-c_j} \quad (\text{要求 } s_j \text{ 与 } s_p \text{ 不冲突})$$

其中s_1=0，即在当前行不放置棋子；
j和p是需要枚举的两个状态编号。

状态压缩——例二

- 本题至此基本解决。当然，实现上仍有少许优化空间，例如第i行只和第i-1行有关，可以用滚动数组节省空间。
- 这个算法时间复杂度$O(n*p*num^2)$，空间复杂度(滚动数组)$O(p*num)$
- 运用简单的组合数学知识可以求出本题中的num=144，因而对于本题的数据规模可以很快出解

P2622 关灯问题II

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

对于20%数据，输出无解可以得分。

对于20%数据，n<=5

对于20%数据，m<=20

上面的数据点可能会重叠。

对于100%数据 n<=10,m<=100

输入输出样例

输入样例#1： 复制

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

输出样例#1： 复制

```
2
```

算法分析

- 灯的个数很小，好像可以状压，1表示灯是开的，0表示灯是关着的
- F[11111111]开始状态， F[0000000]表示目标状态
- 一次操作，就是改变一次状态
- 最少的步数改到00000000
- 广搜即可

P1879 [USACO06NOV]玉米田Corn Fields

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#1： 复制

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

输出样例#1： 复制

```
9
```


题意简述

- 1表示土地肥沃，0表示土地贫瘠
- 两片相邻的土地不能同时种草
- 提问：一共有多少种植方案可以选择?
- 首先，土地的状态可以使用一个二进制表示
- 状态如何设计?
- f[i][j]表示前i行状态为j的时候的方案数
- f[i][j]=(f[i][j]+f[i-1][k])（j&k==0）
- 最后答案是前n行各种状态的和

代码技巧

- 由于两片相邻的土地不能同时种草，所以我们可以将状态进行预处理，存到数组中，节省动规枚举的时间
- 草地的状态需要处理，但是在处理的过程中，需要注意：二进制的方式是从右到左，而输入的方式是从左到右
- 肥沃的地方为1，贫瘠的地方为0，种草的地方为1，不种草的地方为0，对于一个状态j而言，如何才能合法?
- J&a[i]==j，表示这种种植方法合法，因为是1的地方都能种植

P1896 [SCOI2005]互不侵犯

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）

输入输出格式

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

输入输出样例

输入样例#1： 复制

```
3 2
```

输出样例#1： 复制

```
16
```


算法分析

- 观察数据范围 n<=9
- 本题中有数量限制，放置p个国王
- 国王的攻击方式为相邻的八个方向
- 预处理状态i，两个1不能相邻
- 预处理状态i，不同的i，能放的国王不一样（里面的1不同）

算法分析

- 如何设计状态?
- F[i][j]能不能表示了?
- 不能，还有个数，应该增加个数
- f[i][j][x]表示前i行，状态为j的时候，放x个国王的方案数量
- f[i][j][x]+=f[i-1][k][x-cnt[j]];(j和k不能相互攻击)
- 最终答案是f[n][i][kk]，所有状态相加即可

P2704 [NOI2001]炮兵阵地

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：

P	P	H	P	H	H	P	P
P	H	P	H	P	H	P	P
P	P	P	H	H	H	P	H
H	P	H	P	P	P	P	H
H	P	P	P	P	H	P	H
H	P	P	H	P	H	H	P
H	H	H	P	P	P	P	H

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。

P2704 [NOI2001]炮兵阵地

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。

输入输出样例

输入样例#1： 复制

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

输出样例#1： 复制

```
6
```


算法分析

- 字母P可以放炮，字母H不可以放炮
- 炮的攻击范围为上下两个格
- 最多能摆放多少个炮?
- 表示原图状态，字母P的位置为1，字母H的地方为0
- 状态预处理：相邻三个1或者相邻两个1的不能同时使用
- f[i][j]表示前i行，状态为j时最多能摆放多少个炮
- 不仅和i-1行相关，同时和i-2行相关
- f[i][j][k]=max(f[i][j][k],f[(i-1)][k][t]+cnt[j])
- (j和k，j和t，k和t不能相互攻击)
- 最后答案是任意一行任意状态的最大值

P2915 [USACO08NOV]奶牛混合起来Mixed Up Cows

约翰家有N头奶牛，第i头奶牛的编号是Si，每头奶牛的编号都是唯一的。这些奶牛最近 在闹脾气，为表达不满的情绪，她们在挤奶的时候一定要排成混乱的队伍。在一只混乱的队 伍中，相邻奶牛的编号之差均超过K。比如当K = 1时，1, 3, 5, 2, 6, 4就是一支混乱的队伍， 而1, 3, 6, 5, 2, 4不是，因为6和5只差1。请数一数，有多少种队形是混乱的呢？

输入格式：

* Line 1: Two space-separated integers: N and K

* Lines 2..N+1: Line i+1 contains a single integer that is the serial number of cow i: S_i

输出格式：

* Line 1: A single integer that is the number of ways that N cows can be 'Mixed Up'. The answer is guaranteed to fit in a 64 bit integer.

N<=16

0=<号码的值<=25000

1=<k<=34000

输入输出样例

输入样例#1： 复制

```
4 1
3
4
2
1
```

输出样例#1： 复制

```
2
```

算法分析

- N<=16,貌似可以状压
- I表示所有奶牛的排位情况
- 存在的奶牛为1，不存在的奶牛为0
- 考虑最后进来的奶牛是谁？对于一个状态，01101100,最后进来的奶牛可以是1中的任意一个
- f[i][j]表示状态为i时，最后进来的奶牛为j的方案
- f[i][j]= f[i][j]+f[i^(1<<(n-j))][k] (abs(a[j]-a[k])>kk)
- 最后答案是f[11111][1---n]加起来

P3092 [USACO13NOV]没有找零No Change

约翰到商场购物，他的钱包里有K(1 <= K <= 16)个硬币，面值的范围是1..100,000,000。

约翰想按顺序买 N个物品(1 <= N <= 100,000)，第i个物品需要花费c(i)块钱，(1 <= c(i) <= 10,000)。

在依次进行的购买N个物品的过程中，约翰可以随时停下来付款，每次付款只用一个硬币，支付购买的内容是从上一次支付后开始到现在的这些所有物品（前提是该硬币足以支付这些物品的费用）。不幸的是，商场的收银机坏了，如果约翰支付的硬币面值大于所需的费用，他不会得到任何找零。

请计算出在购买完N个物品后，约翰最多剩下多少钱。如果无法完成购买，输出-1

输入格式：

* Line 1: Two integers, K and N.

* Lines 2..1+K: Each line contains the amount of money of one of FJ's coins.

* Lines 2+K..1+N+K: These N lines contain the costs of FJ's intended purchases.

输出格式：

* Line 1: The maximum amount of money FJ can end up with, or -1 if FJ cannot complete all of his purchases.

P3092 [USACO13NOV]没有找零No Change

输入样例#1： 复制

```
3 6
12
15
10
6
3
3
2
3
7
```

输出样例#1： 复制

```
12
```


算法分析

- 虽然N很大，但是硬币数很少，<=16
- 好像可以状压
- 状态i中，1表示用掉，0表示没有用掉
- dp[i]表示当前硬币选择状态为i时，能买到的最大商品数
- 枚举i中的1，如何枚举呢?
- 预处理b数组，b中预存2的x次方
- If（i&b[j]）说明第j位是1
- F[i]=max(f[i],f[i^b[j]]+j)这一位能买到的最远位置
- 如果到达n的话，就退出

优化

- 每一个硬币最多能买到的位置是多少?
- 可以一个一个枚举
- N很大，枚举会超时
- 预处理前缀和，二分即可

练习题

- POJ1691 Painting A Board
- POJ2836 Rectangular Covering
- POJ2285 The Floor Bricks