状态压缩动态规划（简称状压dp）是另一类非常典型的动态规划，通常使用在NP问题的小规模求解中，虽然是指数级别的复杂度，但速度比搜索快，其思想非常值得借鉴。

为了更好的理解状压dp，首先介绍位运算相关的知识。

1.’&’符号，x&y，会将两个十进制数在二进制下进行与运算，然后返回其十进制下的值。例如3(11)&2(10)=2(10)。

2.’|’符号，x|y，会将两个十进制数在二进制下进行或运算，然后返回其十进制下的值。例如3(11)|2(10)=3(11)。

3.’^’符号，x^y，会将两个十进制数在二进制下进行异或运算，然后返回其十进制下的值。例如3(11)^2(10)=1(01)。

4.’<<’符号，左移操作，x<<2，将x在二进制下的每一位向左移动两位，最右边用0填充，x<<2相当于让x乘以4。相应的，’>>’是右移操作，x>>1相当于给x/2，去掉x二进制下的最有一位。

这四种运算在状压dp中有着广泛的应用，常见的应用如下：

1.判断一个数字x二进制下第i位是不是等于1。

方法：if ( ( ( 1 << ( i - 1 ) ) & x ) > 0)

将1左移i-1位，相当于制造了一个只有第i位上是1，其他位上都是0的二进制数。然后与x做与运算，如果结果>0，说明x第i位上是1，反之则是0。

2.将一个数字x二进制下第i位更改成1。

方法：x = x | ( 1<<(i-1) )

证明方法与1类似，此处不再重复证明。

3.把一个数字二进制下最靠右的第一个1去掉。

方法：x=x&(x-1)

感兴趣的读者可以自行证明。

位运算在状压dp中用途十分广泛，请看下面的例题。

【例1】有一个N\*M(N<=5,M<=1000)的棋盘，现在有1\*2及2\*1的小木块无数个，要盖满整个棋盘，有多少种方式？答案只需要mod1,000,000,007即可。

例如：对于一个2\*2的棋盘，有两种方法，一种是使用2个1\*2的，一种是使用2个2\*1的。

【[**算法**](http://lib.csdn.net/base/datastructure)分析】

在这道题目中,N和M的范围本应该是一样的，但实际上，N和M的范围却差别甚远，对于这种题目，首先应该想到的就是，正确算法与这两个范围有关！N的范围特别小，因此可以考虑使用状态压缩动态规划的思想，请看下面的图：

假设第一列已经填满，则第二列的摆设方式，只与第一列对第二列的影响有关。同理，第三列的摆设方式也只与第二列对它的影响有关。那么，使用一个长度为N的二进制数state来表示这个影响，例如：4(00100)就表示了图上第二列的状态。

因此，本题的状态可以这样表示：

dp[i][state]表示该填充第i列，第i-1列对它的影响是state的时候的方法数。i<=M,0<=state<2N

对于每一列，情况数也有很多，但由于N很小，所以可以采取搜索的办法去处理。对于每一列，搜索所有可能的放木块的情况，并记录它对下一列的影响，之后更新状态。状态转移方程如下：

dp[i][state]=∑dp[i-1][pre]每一个pre可以通过填放成为state

对于每一列的深度优先搜索，写法如下：

1. //第i列，枚举到了第j行，当前状态是state，对下一列的影响是nex
2. **void** dfs(**int** i,**int** j,**int** state,**int** nex)
3. {
4. **if** (j==N)
5. {
6. dp[i+1][nex]+=dp[i][state];
7. dp[i+1][nex]%=mod;
8. **return**;
9. }
10. //如果这个位置已经被上一列所占用,直接跳过
11. **if** (((1<<j)&state)>0)
12. dfs(i,j+1,state,nex);
13. //如果这个位置是空的，尝试放一个1\*2的
14. **if** (((1<<j)&state)==0)
15. dfs(i,j+1,state,nex|(1<<j));
16. //如果这个位置以及下一个位置都是空的，尝试放一个2\*1的
17. **if** (j+1<N && ((1<<j)&state)==0 && ((1<<(j+1))&state)==0)
18. dfs(i,j+2,state,nex);
19. **return**;
20. }
21. **for** (**int** i=1;i<=M;i++)
22. {
23. **for** (**int** j=0;j<(1<<N);j++)
24. **if** (dp[i][j])
25. {
26. dfs(i,0,j,0);
27. }
28. }

最终，答案就是dp[M+1][0]