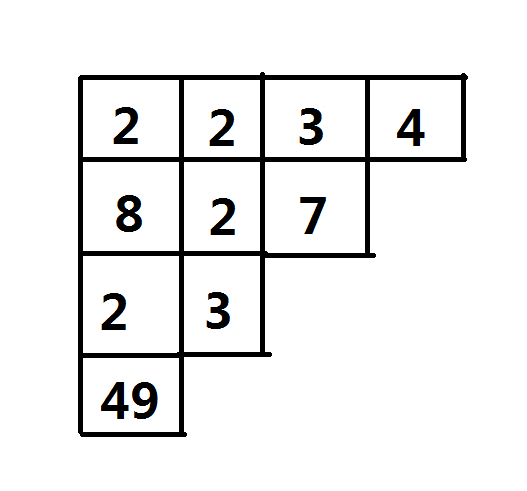
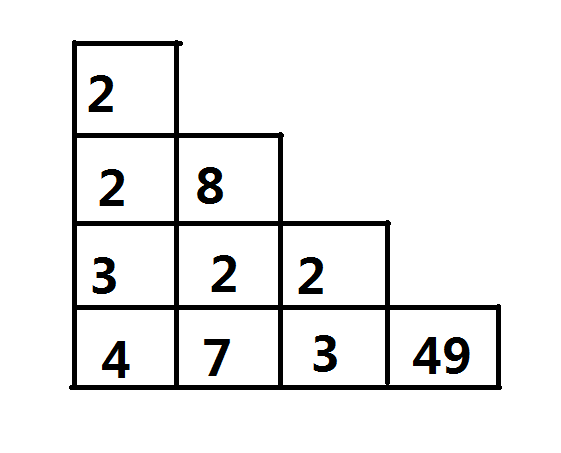
**打砖块**

对于样例数据，我们可看成



如果旋转一下：



我们能够发现，取8的前提是取上一行的2以及去其前面的2

取第三行最右边的2的前提是取上一行所有以及最上面一行，和他前面的所有元素。

以此类推，一个规律就显然出现了：

如果要将第i行第j个敲掉，那么第i行的前j-1个都需要敲掉，同时第i-1行也要敲掉前j-1个。

我们不妨设f[i][j][k]表示第i行取前j个并且一共取到了k个点。

F[i][j][k]=max{f[i-1][v][k-j]}+sum[i,j](v>=j-1,sum[i,j]表示第i行的前j项和。

其中有一点需要注意，因为枚举的j可能为零，这里需要判断一下。

**单词缩写**

先暴力去掉所有的单词，只对有效单词进行计算。

用F[i][j][k]表示当前做到缩写的第i位，全称中第j个单词的第k位，有多少种解释方法。

因为每个全称中的单词必须有一个被选中为缩写中的一个字符，所以转移只需要枚举缩写的第i+1位出现在第j个单词的第p（p>=k）位，F[i+1][j][p]=F[i+1][j][p]+F[i][j][k].

或者出现在第j+1个单词的第L位，F[i+1][j+1][L]=F[i+1][j+1][k]+F[i][j][k]。

注意：两种转移的第p位和第L位都必须是缩写的第i+1位字母的小写形式。最后答案只需要枚举缩写最后一个字符出现在全称中最后一个单词的第x位累加起来。

**二进制**

DP。设L为A(2)，B(2)，C(2)中长度的最大值，nA为A(2)中1的个数，nB为B(2)中1的个数，nC为C(2)中1的个数，然后用dp解决。F[L][i][j][k][p]表示长度为L，X中已有i个1，Y中已有j个1，Z中已有k个1，前一位进位情况为p的状态下Z的最小值。转移枚举X,Y,Z这三个数当前位分别是0还是1，设分别为u,v,w，那么在这位上就需要满足(u+v+p)mod 2= w，那么F[L][i][j][k][p]+w\*2l就可以转移给F[L+1][i+u][j+v][k+w][(u+v+p-w)/2]

最后的答案就为F[L][nA][nB][nC][0]。

**塔**

背包DP变形，对于第i个木块，我们可以不用，也可以用在较高的塔上，也可以用在较低的塔上。

用F[i][j]表示已经处理完前i个木块，当前两座塔的高度差为j的时候，较高的那座塔的最高高度能为多少，然后分三种情况转移即可：

1. 不用幕木块i+1，f[i][j]->f[i+1][j];
2. 将木块放到较高的塔上，f[i][j]+length[i+1]->f[i+1][j+length[i+1];
3. 将木块放到较低的塔上，f[i][j]+max(0,length[i+1]-j)->f[i+1][abs(j-length[i+1])。

时间效率：

O(N\*sum(length[i])；

空间效率：

O(N\*sum(length[i])；

可以使用滚动数组，将效率改为：O(sum(length[i])；