程序员的算法趣题之Q03：翻牌

这里有100张写着数字1-100的牌，并按顺序排列着。最开始所有牌都是背面朝上放置。某人从第2张牌开始，隔一张牌翻牌。然后第2,4,6，…，100张牌就会变成正面朝上。

接下来，另一个人从第3张牌开始，隔2张牌翻牌（原本背面朝上的，翻转成正面朝上；原本正面朝上的，翻转成背面朝上）。再接下来，又有一个人从第4张牌开始，隔3张牌翻牌。

像这样，从第n张牌开始，每隔n-1张牌翻牌，直到没有可翻动的牌为止。

**那么问题来了……**

求当所有牌不再变动时，所有背面朝上的牌的数字。

提示：因为只是单纯从左往右的处理，所以请用简单的方法实现。

==================分割线===================

思路

只要根据问题描述，按顺序对牌进行翻转处理就可以了。用数组保存牌的状态，如果牌正面朝上，则设置值为true,反之为false。这样一来，我们就可以简单地模拟翻转操作了。

上述过程可以简化。从左到右按顺序处理也就意味着“已经翻转过的部分不再翻转”。

Python代码如下：

*# 对于每一张牌，将其未来将会发生的翻转操作一次性完成***for** i **in** range(1, 101):  
 flag = **False** *# 寻找这张牌的所有约数，即每一次翻转的情形* **for** j **in** range(1, i+1):  
 **if** i % j == 0:  
 flag = **not** flag *# 相当于一次翻牌* **if** flag:  
 print(i)

执行这个代码后就可以正确输出答案“1、4、9、16、25、36、49、64、81、100”。从答案可以看到，结果都是“平方数”。

如果翻牌操作进行了奇数次，则最后是正面朝上；如果进行了偶数次，则最后是背面朝上。也就是说，这个问题等价于“寻找被翻转次数为偶数的牌”。而翻牌操作的时机则是“翻牌间隔数字是这个数的约数时”，因此也就相当于寻找拥有偶数个“1以外的约数”的数字。

举个栗子，12的约数时“1、2、3、4、6、12”这6个，也就是偶数个，把约数从小到大排列，并将两端的数按顺序相乘就可以得到原数。也就是1×12,2×6,3×4。

不过16的约数是“1,2,4,8,16”这5个，也就是奇数个。我们把约数从小到大排列，并将两端的数按顺序相乘后，会剩下正中间的数字4。而4乘以自身就可以得到原数（4×4=16）。

也就是说，只有当牌面数字是平方数的时候约数才是奇数个，也就是除1以外的约数是偶数个。了解到这个规律后，即便不编程，也能知道答案。在日常工作中，动手编程之前最好也像这样好好想一想。

参考《程序员的算法趣题》