# 置换，置换的运算 - 孟起 - 博客园

置换的概念还是比较好理解的，《组合数学》里面有讲。对于置换的幂运算大家可以参考一下潘震皓的那篇《置换群快速幂运算研究与探讨》，写的很好。

**结论一**：一个长度为l的循环T，l是k的倍数，则T^k是k个循环的乘积，每个循环分别是循环T中下标i mod k=0,1,2…的元素按顺序的连接。

**结论二**：一个长度为l的循环T，gcd(l,k)=1，则T^k是一个循环，与循环T不一定相同。

**结论三**：一个长度为l的循环T，T^k是m=gcd(l,k)个循环的乘积，每个循环分别是循环T中下标i mod gcd(l,k)=0,1,2…的元素的连接。

如果长度与指数不互质，单个循环就没有办法来开方。不过，我们可以选择相应m个长度相同的循环交错合并来完成开方的过程。可在这种情况下，如果找不到m个长度相同的循环，那就一定不能开方。其中：m是gcd(l,k)的倍数

*简单题：（应该理解概念就可以了）

# pku3270 Cow Sorting

题目描述：

给你一个数字序列（每个数字唯一），每次你可以交换任意两个数字，代价为这两个数字的和，问最少用多少代价能把这个序列按升序排列好。

题目的具体做法是参考刘汝佳的《算法艺术与信息学奥赛》大概思路是：以后再用别种方法解，

1.找出初始状态和目标状态。明显，目标状态就是排序后的状态。

2.画出置换群，在里面找循环。例如，数字是8 4 5 3 2 7

明显，　　目标状态是2 3 4 5 7 8，能写为两个循环：(8 2 7)(4 3 5)。

3.观察其中一个循环，明显地，要使交换代价最小，应该用循环里面最小的数字2，去与另外的两个数字，7与8交换。这样交换的代价是：

sum - min + (len - 1) * min

化简后为：

sum + (len - 2) * min

其中，sum为这个循环所有数字的和，len为长度，min为这个环里面最小的数字。

4.考虑到另外一种情况，我们可以从别的循环里面调一个数字，进入这个循环之中，使交换代价更小。例如初始状态：1 8 9 7 6

可分解为两个循环：(1)(8 6 9 7)，明显，第二个循环为(8 6 9 7)，最小的数字为6。我们可以抽调整个数列最小的数字1进入这个循环。使第二个循环变为：(8 1 9 7)。让这个1完成任务后，再和6交换，让6重新回到循环之后。这样做的代价明显是：

sum + min + (len + 1) * smallest

其中，sum为这个循环所有数字的和，len为长度，min为这个环里面最小的数字，smallest是整个数列最小的数字。

5.因此，对一个循环的排序，其代价是sum - min + (len - 1) * min和sum + min + (len + 1) * smallest之中小的那个数字。但这里两个公式还不知道怎么推出来的。

6.我们在计算循环的时候，不需要记录这个循环的所有元素，只需要记录这个循环的最小的数及其和。

7.在储存数目的时候，我们可以使用一个hash结构，将元素及其位置对应起来，以达到知道元素，可以快速反查元素位置的目的。这样就不必要一个个去搜索。

代码

# pku1026 Cipher //先找出所有置换循环，然后对于每一位来计算k%循环长度后对应于哪个位置,O(n)复杂度。注意读写方面的东西。

代码

*置换幂运算：

# pku1721 CARDS //详细见05集训队论文《置换群快速幂运算研究与探讨》。

很显然，这题的一副扑克牌就是一个置换，而每一次洗牌就是这个置换的平方运算。由于牌的数量是奇数，并且一开始是一个大循环，所以做平方运算时候不会分裂。所以，在任意时间，牌的顺序所表示的置换一定是一个大循环。

那么根据文章开头提到的定理：设T^k=e，（T为一循环，e为单位置换），那么k的最小正整数解为T的长度。

可以知道，这个循环的n次方是单位循环，换句话说，如果k mod n=1，那么这个循环的k次方，就是它本身。我们知道，每一次洗牌是一次简单的平方运算，洗x次就是原循环的2x次方。

因为n是奇数，2x mod n=1一定有一个，假设这个解是a；那也就是说，一幅牌，洗a次，就会回到原来的顺序。使用最终顺序不停地洗，直到回到原始顺序，求出循环节长度a以后，再单纯地向前模拟a-s次，就可以得到原始顺序了。

上面的算法是出题方给出的标准算法。显然，时间复杂度为O(n2+logs)。

换一个方向：给定了结果和s以后，可以简单地将这个目标置换用3.1节的方法开方s次得到结果。时间复杂度为O(n*s)。

或者可以更简单地，算出2s，将目标置换直接开2s次方。这里有一个技巧，因为在开方时只需要在循环中前进2s次，所以我们只关心(2s) mod n，也就免去了大数字的运算。所以，计算2s需要O(logs)，而开方需要O(n)。整个时间复杂度为O(n+logs)。

代码

# pku3128 Leonardo's Notebook

题目意思是：一个置换是否可以由另一个置换的平方得来的。一个置换的平方，原来偶数长的循环会被分裂成两段长度相等的循环，而奇数长的循环不会被分裂。题目只是问是否存在，所以只要看所给置换中偶数长的循环是否成对，否则就不能由一个置换的平方得来。

补充：因为如果所给置换的循环是偶数，则肯定是由分裂过来的，那么一定是成对的，否则如果是奇数，那么有可能是原来是奇数，也有可能是原来的偶数分裂成两个奇数循环。

代码

*推荐：（不错的应用）

# pku3590 The shuffle Problem //把n分解成若干个数，使得他们的lcm最大。在所取的数都是素数幂的时候是最大的，所以可以用递归来枚举所有的分解情况，而且由于要输出序最小的，所以对于剩下的数可以直接单独都作为一个循环，这样就可以使得序最小了。此外，这道题目需要注意求最大

的lcm的时候不能用dp来做，因为这个具有后效性，局部最优不一定使得全局最优。
代码