1. 快速幂算法：时间复杂度为。

lintcode 140，用到了取模的相关知识：

(a + b) % c = (a % c + b % c) % c

(a \* b) % c = (a % c \* b % c) % c

(a - b) % c = (a % c - b % c) % c

如果a, b, c都是正整数，取模时，加法和乘法都没有问题，但减法需要注意有可能出现的负数问题，由于在不同的语言中负数取模的结果不一致，所以为了保险起见，可以如下操作：

(a - b) % c = (a % c - b % c + c) % c

由于c % c =0，a % c和b % c的结果肯定都小于c，两者的差的绝对值肯定小于等于c，所以加上c后，不影响最终的取模值。

1. 辗转相除法：时间复杂度为。

欧几里得算法，就两个数的最大公约数。lintcode 845.

1. 分解质因数：时间复杂度为。lintcode 235
2. 分块检索法：时间复杂度为。将长度为n的区间分成的大小的区间。总共有个小区间，每个小区间统计局部数据。

lintcode 249，分块的区间n为A中最大值，而不是数组的长度。

lintcode 65，两个排序数组的中位数。

算法描述

先将找中点问题转换为找第 k 小的问题，这里直接令k = (n + m) / 2。那么目标是在 logk = log((n+m)/2) = log(n+m) 的时间内找到A和B数组中从小到大第 k 个。

比较 A 数组的第 k/2 小和 B 数组的第 k/2 小的数。谁小，就扔掉谁的前 k/2 个数。

将目标寻找第 k 小修改为寻找第 (k-k/2) 小

回到第 2 步继续做，直到 k == 1 或者 A 数组 B 数组里已经没有数了。

F.A.Q

Q: 如何 O(1) 时间移除数组的前 k/2 个数？

A: 给两个数组一个起始位置的标记参数（相当于一个offset，偏移位），把这个起始位置 + k/2 就可以了。

Q: 不是让我们找中点么？怎么变成了找第 k 小？

A: 找第 k 小如果能在 log(k) 的时间内解决，那么找中点就可以在 log( (n+m)/2 ) 的时间内解决。

Q.如何证明谁的第 k/2 个数比较小就扔掉谁的前 k/2 个数这个理论？

A: 直观的，我们看一个例子

A=[1,3,5,7]

B=[2,4,6,8]

假如我们要找第 4 小。也就是 k = 4。算法会去比较两个数组中第 2 小的数。也就是 A[1]=3 和 B[1]=4 这两个数的大小。然后会发现，3比较小，然后就决定扔掉 A 的前 k/2 = 2 个数。也就是，接下来，需要去找

A=[5,7]

B=[2,4,6,8]

中的第 k-k/2=2 小的数。这里我们扔掉了 [1,3]，扔掉的这些数中，一定不会包含我们要找的第 4 小的数——4。因为从位置上，他们在 A 和 B合并到一起之后，都会排在 4 的前面。

抽象的证明一下：

我们需要回顾一下 Merge Two Sorted Arrays 这道题目。算法的做法是，每一次比较两个数组中比较小的数，然后谁小，谁先被拿出来，放到最后的合并结果中。那么假设 A 和 B中 A[k/2 - 1] <= B[k/2 - 1]（反之同理）。我们会决定扔掉A[0..k/2-1]，因为这些数在 A 与 B 做简单的 Merge 的过程中，会优先于目标第 k 个数现出来。为什么？因为既然A[k/2-1] <= B[k/2-1]，那么当我们用最简单的 Merge Two Sorted Arrays 的算法一个个从A和B里拿数出来的时候，当 A[k/2 - 1] 出来的时候，B[k/2 - 1] 一定还没有被拿出来，那么此时A里出来了 k/2 个数，B里出来的数一定不够 k/2 个（因为第 k/2 个数都还没出来），所以加起来总共出来的数肯定不够k个，所以第k小的数一定还留在AB数组中。

因此我们证明了：扔掉较小的一部分的前 k/2 个数，不会扔掉要找的第 k 小的数。

lintcode 65，求两个排序数组的中位数，要求时间复杂度为log(n+m)

中位数假设为第k个数，则k=(n+m)/2，如果可以在O(1)的时间内把找第k个数，变成找第k/2个数，并且这两个位置对应的值是相同的，则整体的实际复杂度为logk，即log(n+m)，满足要求，这就要求我们要用O(1)的操作丢掉A或B(同时丢弃，无法保证O(1))中的k/2个数，。

A数组的第k/2个数为A[k/2-1]，B数组的第k/2个数为B[k/2-1]，比较这两个数，谁小就丢弃对应的数组，假设A[k/2-1]较小，可以参考归并排序中的合并，在合并两个排序数组时，从A和B的头开始，谁小谁就先入到合并数组中，这样当A[k/2-1]进入到合并的数组后，B[k/2-1]肯定没有进入到合并数组中，这时丢弃A的前k/2个数，则之前找第k个数，就变成了A剩余的元素和B中的元素找k/2个元素，就可以满足要求。