[toc]

我们说，如果存在一个整数 k，使得 $a = kd$，则称 d 整除 a，记做 $d \mid a$，称 a 是 d 的倍数，如果 $d > 0$，称 d 是 a 的约数。特别地，任何整数都整除 0。

显然大于 1 的正整数 a 可以被 1 和 a 整除，如果除此之外 a 没有其他的约数，则称 a 是素数，又称质数。任何一个大于 1 的整数如果不是素数，也就是有其他约数，就称为是合数。1 既不是合数也不是素数。

素数判定

我们自然地会想到，如何用计算机来判断一个数是不是素数呢？

暴力做法

自然可以枚举从小到大的每个数看是否能整除

```
bool isPrime(a) {
  if (a < 2) return 0;
  for (int i = 2; i < a; ++i)
    if (a % i == 0) return 0;
  return 1;
}
```

这样做是十分稳妥了，但是真的有必要每个数都去判断吗？

很容易发现这样一个事实：如果 x 是 a 的约数，那么 $\frac{a}{x}$ 也是 a 的约数。

这个结论告诉我们，对于每一对 $(x, \frac{a}{x})$，只需要检验其中的一个就好了。为了方便起见，我们之考察每一对里面小的那个数。不难发现，所有这些较小数就是 $[1, \sqrt{a}]$ 这个区间里的数。

由于 1 肯定是约数，所以不检验它。

```
bool isPrime(a) {
  if (a < 2) return 0;
  for (int i = 2; i * i <= a; ++i)
    if (a % i == 0) return 0;
  return 1;
}
```

Miller-Rabin 素性测试

Miller-Rabin 素性测试（Miller–Rabin primality test）是进阶的素数判定方法。 对数 n 进行 k 轮测试的时间复杂度是 $O(k \log^3n)$.

Fermat 素性测试

我们可以根据费马小定理 得出一种检验素数的思路：

它的基本思想是不断地选取在 $[2, n-1]$ 中的基 a，并检验是否每次都有 $a^{n-1} \equiv 1 \pmod n$

```
bool millerRabin(int n) {
  if (n < 3) return n == 2;
  // test_time 为测试次数,建议设为不小于 8
  // 的整数以保证正确率,但也不宜过大,否则会影响效率
  for (int i = 1; i <= test_time; ++i) {
    int a = rand() % (n - 2) + 2;
    if (quickPow(a, n - 1, n) != 1) return 0;
  }
  return 1;
}
```

很遗憾，费马小定理的逆定理并不成立，换言之，满足了 $a^{n-1} \equiv 1 \pmod n$，n 也不一定是素数。