

基于 Miller-Rabin 算法的大素数生成器

-BY PB25071407 安鑫丞

一、作品宣传

这是一个完全自主实现的 1024 位大素数生成工具。项目采用严格的 C99 标准，仅使用标准库函数，实现了从大数存储、运算到素数测试的完整解决方案。代码写作符合标准。

应用价值：

- (1) 密码学基础工具：为 RSA 等加密算法提供核心的素数生成功能。
- (2) 算法学习平台：完整展示大数运算、素性测试的实现原理。
- (3) 性能优化范例：展示了算法优化的多种技术手段

二、亮点展示

(1) 核心亮点

① 大数运算库

1. 实现了 256 进制的大数的表示和运算，包括加减乘模四个运算和快速幂算法。（除法内置于 Barrett 预处理中，未单独列出。）

2. 实现了 256 进制和 2 进制的相互转换。

② Miller-Rabin 算法

1. 10 轮检验，保证算法准确性。

2. 设计了详尽的测试过程输出，便于验证和调试。

③ Barrett 约减算法

1. 将耗时的模运算转换为乘法和移位操作。

2. 适用于多次模数不变的模运算，在快速幂中特别适合。

④ 性能优化

1. 预计算 256*256 乘法表，避免乘法重复运算。

2. 小质数筛选合数，筛去大部分需要检验的数。

3. 除法使用 2 进制运算，大幅减小了时间复杂度和写作难度。

(2) 算法性能

① 速度：平均 8 秒左右生成一个大质数，极端情况下（尝试 800 次）的时间约为 14s。

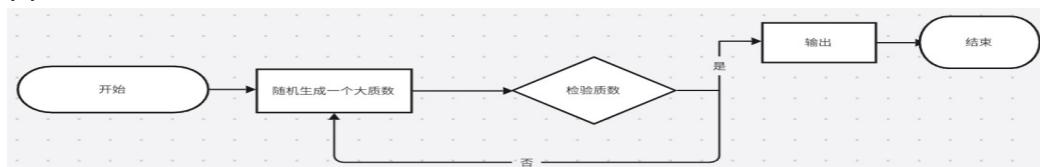
以下是本作品中主要算法的时间复杂度（ n 表示 256 进制大数位数， m 为 2 进制位数）：

时间复杂度	$O(n)$	$O(n^2)$	$O(n^2m)$	$O(m^2)$
算法	加法，减法，大小比较，进制转换	乘法，Barrett 约减	快速幂，单次 Miller-Rabin 检验	Barrett 预处理

② 准确性：错误概率小于 10^{-6} ，满足密码学应用要求。

三、程序说明

(1) 核心框架



(2) 模块

（注：部分函数分为 2 进制版本和 256 进制版本，这里只列出其中一个函数名）

1. 高精度运算：add(), cmp(), sub(), mul(), mol(), mulmod(), div_small(), quickpow()。

2. 质数检验函数：cprime(), check(), try_small(), make()。

3. 随机数函数: `randx()`, `random()`, `makeit()`。
4. 辅助函数: `clean()`, `ctrlcv()`, `outprint()`, `st()`;
5. 进制转换: `ftn()`, `fnt()`。
6. 大数存储: `bignum` (256 进制), `big_binary` (二进制)。

(3) 质数检验

1. 40 次小质数尝试
2. 使用 Miller-Rabin 检验算法

```

1) 计算奇数 q 和整数 k, 使得  $n - 1 = 2^k \times q$ 
2) 随机生成整数 a,  $1 < a < n - 1$ 
3) if  $a^q \bmod n = 1$  or  $n - 1$  then return 可能是;
4) for j = 0 to k-1 do
   if  $(a^{2^j} \bmod n = n - 1)$  then return 可能是;
   if  $(a^{2^j} \bmod n = 1)$  then return 合数;
5) return 合数;

```

(4) 核心函数说明

1. `mul()`乘法
 - 1) 朴素竖式乘法
 - 2) 使用预处理乘法表和统一进位方式优化常数
2. `mol()`及 `Barrett_preset()`模运算

伪代码(m: 模数, b: 基数 (256), k: m 的位数)

```

function Barrett_preset(m,b):
     $\mu = \lfloor b^{(2k)} / m \rfloor$ 
    return (k,  $\mu$ )
function mol(x, m, b, k,  $\mu$ ):
     $q = \lfloor (x / b^{(k-1)}) \times \mu \rfloor / b^{(k+1)}$ 
    r = x - q * m
    if r >= m:
        r = r - m
    if r < 0:
        r = r + m
    return r

```

其中, 计算除法的时候, 先将 256 进制转化为 2 进制, 进行 2 进制的除法运算。运算时, 只需每一位依次比较, 可减处减去并在答案的那一位上记为 1, 最后转化为 256 进制。相较于 256 进制的除法, 时间复杂度和写作难度大大下降。

3. `quickpow()`快速幂

不同于常规递归方法, 使用循环来进行运算, 使用现有的二进制转换来判断需要多乘一个原数的位置。

备注:

- (1) 输出在 `answer.txt` 呈现, 使用 16 进制输出。
- (2) 拓展性较好, 可以通过微调来生成其他位数的大素数; 也可封装为库。