现代密码学第三次作业

郑凯文

2021 年 6 月 5 日

1 手写高精度库

为了进行大整数的运算，我写了一个简单的支持任意长无符号整数及其四则运算的高精度库`BigUInt`（其实只是一个struct+运算符重载）。

1.1 内部存储

由于这里的高精度数是通过比特序列表示的，全程用不到10进制，在实现中也完全舍弃了10进制的存在。虽然不奢求与成熟的高精度库比效率，但运算速度也不能过慢，毕竟Miller-Rabin算法存在大量取模操作。提高速度必须进行压位操作以充分发挥64位处理器的优势，如对于十进制数，可以用1000000进制，每位用一个`int`存储。

经过权衡，这里采取的是2^{32}进制，每位用一个`u32`存储，这样`BigUInt`内部就是一个`std::vector<u32>`。不选取`u64`的原因是，有时会用到两个`u32`拼起来的`u64`进行中间运算。

1.2 四则运算

对于加法、减法、乘法，只是模拟了手算列竖式，为最朴素的做法（实际上高精度乘法存在Karatsuba算法这种简单的优化，我在数据结构课上也尝试过FFT，但最终参与运算的512比特也只是16个`u32`，复杂算法中分治带来了额外开销，可能得不偿失）。对于除法和取模，可以用二分法搜索每一位的商，我这里用被除数的高位、当前位拼起来的`u64`和除数的当前位（最高位）相除，得到的商来估计真正的商，相减后的余数作为新的被除数重复此过程，直到只能商0。虽然上面说，每位用$u32$便于中间运算，但高精度除法某次试除时，某位上的余数可能是$-(2^{64}-1)\sim(2^{64}-1)$的范围，`u64`也不足模拟了，所以我额外用一个布尔变量表示符号，虽然麻烦了一点，但只是加了几个条件判断，没有带来过多额外开销。

1.3 正确性测试与性能测试

高精度四则运算尤其是高精度除法，在我实现的版本中，需要考虑许多边界情况与原码、补码的性质。为验证正确性，我分别随机生成长度为10000、5000的比特序列，进行四则运算并

和GMP库的运算结果进行比较，重复10次。

我还生成了同样长度的比特序列，进行1000次四则运算并测速，以此来估计我的实现和GMP的速度差异。结果如下

```
gmp add is 34.083333x times faster
gmp sub is 34.305556x times faster
gmp mul is 9.068272x times faster
gmp div is 47.157559x times faster
```

还是差距很大的，连加减法这种O(n)的运算也相差了几十倍。

2 Miller-Rabin算法

2.1 算法原理

对于一个奇素数n，设$n-1=2^s r$，其中r为奇数，对于任意a满足$2 \le a \le n-2$，必然有

- $gcd(a,n)=1$
- 以下两条中的一条成立
  - 存在某个k满足$0 \le k \le s-1$，$a^{2^k r} \equiv -1 \pmod{n}$
  - 对于任意k满足$0 \le k \le s-1$，$a^{2^k r} \equiv 1 \pmod{n}$

于是取若干a，验证是否满足上述条件。伪代码为

```
Miller-Rabin(n)
    n - 1 = r * 2^s, 其中r为奇数
    for i = 1 to rounds
        选取随机整数a, 使得2 <= a <= n - 2
        if gcd(a, n) != 1
            return ("n is composite")
        x = a^r (mod n)
        if (x == 1) || (x == n - 1)
            continue
        for k = 1 to s - 1
            x = x^2 (mod n)
            if x == n - 1
                break
            if x == 1
                return ("n is composite")
        if x != n - 1
            return ("n is composite")
    return ("n is prime")
```

2.2 运行结果

程序会随机生成一个512比特，最高、最低位为1的长整数，并调用算法判断是否是素数。若不是，继续随机，直到生成一个素数。某次运行时，程序输出为：

```
Testing 11100001000111000100010111101001111111010111011001100101010110011100111110000...
Result: not prime
Searching for prime...
102 numbers searched
Find a prime 10011110001111011010100000000111010111111001110011101111111111010001100...
```

其中完整数字分别在not_prime.txt和prime.txt中。方便的在线验证方式为进制转换 - 在线工具和Integer factorization calculator。