1. 算法简介

函数prime\_sieve(int64\_t L, int64\_t N)实现输出第N个素数，素数的上界为L。

此函数使用Eratosthenes筛法，算法复杂度为O(nloglogn)。由于一个合数总是可以分解成若干个质数的乘积，那么只要把所有小于的素数的倍数都去掉，那么剩下的就都是质数了。例如要查找100以内的质数，首先2是质数，把2的倍数去掉；此时3没有被去掉，可认为是质数，所以把3的倍数去掉；再到5，再到7，7之后呢，因为8，9，10刚才都被去掉了，而100以内的任意合数肯定都有一个因子小于10（100的开方），所以去掉2，3，5，7的倍数后剩下的都是质数了。

为了更有效的利用CPU cache，我们利用改进的Eratosthenes筛法，即将整个数列分割成若干个长度为segment\_size(CPU L1缓存大小)的数列。从前到后逐个筛选每个数列内的素数，并统计每个数列内的素数个数，当统计到的素数个数等于N时，便停止整个筛选过程，不再处理后面的数列，从而避免了不必要的时间开销。

1. 实验结果

|  |  |
| --- | --- |
| N | time(s) |
| 10,000,000 | 0.327 |
| 50,000,000 | 1.860 |
| 100,000,000 | 4.025 |
| 200,000,000 | 8.909 |