ISSN: 2705-0408 (P) ; 2705-0416 (0)

浅谈素数筛优化及其 C++ 程序实现

倪　航　刘万能　朱小龙

（江汉大学光电化学材料与器件教育部重点实验室　湖北武汉　430056）

【摘　要】 简单介绍了素数筛的三种方法：枚举法，埃氏筛法，欧拉筛法。通过C++程序实现，并作比较可知，优化后的枚举法仍然极为耗时，埃氏筛法大大缩减了用时，而欧拉筛法则进一步实现了明显的优化。本文对初学者有一定帮助。

【关键词】 素数筛；埃氏筛；欧拉筛

DOI: 10.18686/jyyxx.v2i3.33347

素数也称质数是指大于 1 并且只能被 1 和自身整除的自然数。素数在生活与工作中有着广泛的应用，比如，大素数用于密码学，利用合数质因数分解的唯一性，素数越大，密码被破译的可能性就越小，齿轮的齿数设为素数，能使磨损更均匀一些。通常人们会关心判断某个正整数是否为素数以及快速的找到大量的素数，人工计算已不太现实，往往通过编程让计算机求解 [1-4]。

用计算机求解的最直接的算法是朴素枚举，对于判断某个正整数 n 是否为素数，最为朴素的做法是枚举小于 n 的所有大于 1 的正整数是否为 n 的因数。

但是显然能够对此方法进行优化。考虑如果 n 为合数，那么 n 一定能够写成 a × b（a, b 为大于 1 的正整数）的形式。不妨设 a≥b，那么只需要枚举较小的因子 b 即可。即若对于所有的正整数 k, $2 \leqslant k \leqslant \sqrt{n}$，$n$ 不能整除 k。这是判断某个特定的正整数是否为素数所使用的方法，但在某个较大的范围中确定存在的素数时，依次判断会花费大量的时间。这时需要使用素数筛法，大体思路是将范围内所有的合数划掉，剩下的就是素数。

已知大于 1 的整数满足唯一分解定理，也就是说一个合数一定能够分解为有限个质因数相乘的形式。这一点是筛法能够存在的核心。最为常用的就是埃氏筛和欧拉筛。

对于单独素数的判断要枚举因子，那么反过来枚举因子的倍数是可行的。若 p 为素数，对于正整数 $i \in [2, \infty]$，$p \times i$ 必然为合数。

埃氏筛的流程：从 2 开始，每一个没有标记的数都判断为素数，然后用这个数去更新给出的上界内所有的数。也就是将这个数所有的倍数标记。这样对于后面每一个数 n，都可以保证它不是前 n-1 个数的倍数（排除 1），这样其实和最基础的判断效果一样，也保证了正确性。

埃氏筛的 c++ 实现关键代码：

```
bool pri[1000]={0};//pri[i]值为1则i为合数，否则i为素数.
void findpri(int n){
    for(int i=2;i<=n;i++){
        if(pri[i]==0){
            for(int j=i*2;j<=n;j+=i){
                pri[j]=1;// 标记不是素数
            }
        }
    }
}
```

外层循环为 n，当外层循环锁定一个数时，设为 i，那么内层的循环次数

$$\left[\frac{n}{i}\right]-1 \quad \frac{n}{i}$$

那么循环总数即为

$$\frac{n}{2}+\frac{n}{3}+\cdots+\frac{n}{n}$$

提取 n 得

$$n\times\left(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}\right)$$

使用大 O 表示法表示时间复杂度，埃氏筛的时间复杂度 O(nlogn)。

欧拉筛则在埃氏筛的基础上进行了优化，优化了时间复杂度。在埃氏筛中，若有一个合数 p, 对 p 进行质因数分解：p = v[1] × v[2] × … × v[k], p 会被 v[1], v[2], …, v[k] 各筛去一次，共筛去 k 次，这其中除了第一次筛除，另外 k-1 次都是多余操作。那么考虑如何避免多余操作：任一合数只被“最小质因数 × 最大因数（非自己）= 这个合数”的途径删掉。

欧拉筛的 c++ 实现关键代码：

```
bool pri[100000005]={0};
int ispri[10000005]={0};
int cnt=0;
voidfindpri(int n){
    pri[1]=1;
for(int i=2;i<=n;i++){
if(pri[i]==0){
            ++cnt;
            ispri[cnt]=i;
      }
for(int j=1;j<=cnt&&ispri[j]*i<=n;j++){
            pri[ispri[j]*i]=1;
if(i%ispri[j]==0){
break;
            }
        }
    }
}
```

欧拉筛法有“if(i%p[j]==0) break;”这个语句，即判断 i 是否整除 p[j]，因为每个数都可以表示为质数的积，所以如果出现了 i%p[j]==0 的情况，那么 i 就不是 i × p[j] 的质因子，而是一个合数，若将 i 表示为两个数：i 最小的质因子 a 与 i/a 的值 b 相乘 (i=a × b)，那么 i × p[j] 也可表示为 (b × p[j]) × a，所以当前就不用筛掉 i × p[j]，等到 i 循环到 b × p[j] 时筛掉 (b × p[j]) × a 即可。

由于每个数只被筛一次，所以欧拉筛的时间复杂度为 O(n)。

接下来用 C++ 程序实现上述三种算法，分别计算出 2 ～ 10^8 之间的素数个数，并对比用时。

运行环境为：

处理器	内存	系统	软件
i7-8550, 1.80GHZ	8GB	Win10, 64 位	Dev-C++4.9.2 版

实际运行速度对比：

算法	运行时间（S）
枚举法	382.98
埃氏筛法	2.999
欧拉筛法	1.406

通过结果对比可知，就算采用优化后的枚举法，仍然是极为耗时的，埃氏筛法大大缩减了用时，让实际应用成为可能，而欧拉筛法则进一步实现了明显的优化。

附：欧拉筛法 C++ 完整代码

```
#include  <bits/stdc++.h>
using  namespace  std;
long  long  ans=0;
bool  pri[100000005];
int  ispri[100000005];
int  cnt=0;
void  findpri(int  n){
    pri[1]=1;
    for(register  int  i=2;i<=n;i++){
        if(pri[i]==0){
            ++cnt;
            ispri[cnt]=i;
        }
        for(register  int j=1;j<=cnt&&ispri[j]*i<=n;j++){
            pri[ispri[j]*i]=1;
            if(i%ispri[j]==0){
                break;
            }
        }
    }
}
int  main(){
findpri (100000000);
for (register  int  i=2;i<=100000000;i++){
if(pri[i]){
ans++;
}
}
cout<<ans<<endl;
return  0;
}
```

作者简介：朱小龙（1976—），男，湖北鄂州人，博士，副教授，研究方向：计算物理。

基金项目：湖北省科技计划项目“聚（3，4-乙烯二氧噻吩）/ 碳纳米复合材料的热电性能优化”，项目编号 2017CFB354。

【参考文献】

[1] 周利荣，胡天磊. Demytko 素数构造算法优化及应用研究 [J]. 电脑编程技巧与维护，2018（6）：54-59.
[2] 马麟浚，黎百恬，王顺庆，等. 埃氏筛法的缺陷和理想最终筛法 [J]. 今日科苑，2013（13）：108-114.
[3] 叶煜，周洪林，任华. 素数筛选法的改进及 C 语言实现 [J]. 计算机与数字工程，2013（6）：899-900+964.
[4] 张国钦，宋伟，马俊兴. 素数筛选法的实现及优化 [J]. 河南教育学院学报：自然科学版，2017（1）：39-41.