一类积性函数求和的方法 (翻译)

$\mathcal{H}uxulin$

这个问题在 2-3 年前的 NOI 冬令营上讨论过, 最终我们给出了 $O(\frac{n^{\frac{3}{4}}}{logn})$ 的算法, 然而由于它的常数很大, 所以它通常跑得很慢. 在下面我将给出一个时间复杂度大约在 $O(n^{0.7})$, 空间复杂度为 $O(\sqrt{n})$ 的算法. 在 $N \leq 10^{12}$ 时跑得比洲阁筛快多了.

这个做法是来源于 Hiroaki Yamanouchi. 我是从它的代码里面学到的.

下面的讨论我们将用 $\sqrt{N}$ 来代表最大的整数 W 满足 $W^2 \leq N$; 用 $p, p1, p2, ...$ 表示质数.

参数 $magic$ 可以任意选择, 但我为了方便假设它的值为 $\sqrt{N}$.

考虑一类问题: 积性函数 $f(n)$ 定义为:

$$f(n) = \begin{cases} 1 & n = 1 \\ g(p, e) & n = p^e, e > 0 \\ f(x) \times f(y) & n = xy, (x, y) = 1 \end{cases}$$

当 $f(p) = g(p, 1)$ 时是关于 p 的一个任意低阶多项式.(这并不重要) 给定一个正整数 N, 求 $S = \sum_{i=1}^{N} f(i)$.

考虑对任意的一个 M 分解质因数, 其中 $M \leq N$:

$$M = \prod_{i=1}^{k} p_i^{e_i}$$

$$p_1 < p_2 < ... < p_k, e_i > 0$$

主要的算法是基于下面的结论:

$M' = \frac{M}{P_k}$ 没有大于 $magic$ 的质因子.

所以我们可以用下面的伪代码来计算 S:

Calucate-S

S = f(1)
for each $M' \leq N$ without any prime factor > magic:
  $S+ = f(M') * \sum_{F<p\leq N/M'} f(p)$,where F is largest prime factor of M'
  if (the exponent of F in M') > 1:
    $S+ = f(M')$
  end if
end for
return S

我们枚举的 $f(M')$ 可以通过递推快速的得到. 而 $\sum_{F<p\le N/M'} f(p)$ 可以看做 $S'(N/M')-S'(F)$, 其中 $S'(x)=\sum_{1<p\le x} f(p)$,$p$ 是质数. 现在算法的瓶颈是: 如何快速地得到 $S'(x)$?

观察 X 的取值可能会有哪些, 由于 M' 是不存在大于 $magic$ 的质因子的,F 也就只能取小于等于 $magic$ 的素数. 并且, 如果 $M' > N/magic$, 那么 $N/M' \le magic$. 所以 N/M' 最多也只有 $magic$ 种取值. 所以我们只要算 $magic+N/magic$ 种取值:$1,2,3,...,magic,N/1,N/2,...,N/(N/magic)$. 我们把它叫做集合 NS.

下面的伪代码则为计算 $S'(x)=\sum_{1<p\le p^d} p^d$,$d$ 为非负整数.

```
CALUCATE-S'(x)
for each i in set NS
  Map[i] = Σ_{2≤j≤i} j^d
end for
for p = 2,3,5,7,... (any prime not exceed √N) in increasing order:
  for each i in set NS in decreasing order:
    if i ≥ p*p :
      Map[i] -= (Map[i/p] - Map[p-1]) * p^d
    else break
    end if
  end for
end for
```

这个代码其实是在模拟埃氏筛法的过程: 对每一步, 我们筛掉了质数 p 的倍数对 S' 的贡献, 并且保证了这一步之前没有被筛过.

在执行算法的过程中我们需要把 $Map[1],...,Map[magic]$ 和 $Map[N/1],...,Map[N/(N/magic)]$ 放在两个数组里面来维护. 当枚举 M' 时, 我们不会计算这样的情况:$M'\times F \ge N$ 并且 $M' \mod F^2 \ne 0$.

注意: 这两步可能会影响到整个程序的复杂度.

由于任何一个多项式可以被写作若干个 p^d 的累加和, 对于 d 为了计算任意一个 f 我们可以预处理 $O(d*magic)$ 的值. 对于询问 $S'(x)$, 我们直接拿出来算就可以啦.

空间复杂度是 $O(magic)$, 时间复杂度经过测试是可以大概拟合成 $O(n^{0.7})$.

PS: 后面经过证明复杂度好像是 $O(N^{\frac{2}{3}}(logn)^{\frac{1}{3}})$.

复杂度分析点这里:CodeForces

原文链接点这里:SPOJ