一种求积性函数前缀和的筛法.时间复杂度 $O(\frac{n^{\frac{3}{4}}}{log_2 n})$,空间复杂度 $O(\sqrt{n})$.要求质数 p 处的函数值是个多项式,并且 p^k 处的函数值也很好算.

以下用 $Prime$ 表示素数集合, cnt 表示 $\leq n$ 的质数个数, p_i 表示第 i 个素数 , $R(i)$ 表示 i 的最小质因子,$F(x)$ 是一个积性函数.

除法默认为向下取整.

$Part\ 1$

- 先来考虑求解这样一个式子:

$$\sum_{i=1}^{cnt} F(p_i) = \sum_{x=1}^{n} F(x)[x \in Prime]$$

- 定义二元函数 g ,

$$g(n,j) = \sum_{i=2}^{n} f(i)[i \in Prime \vee R(i) > p_j]$$

- 这里的 f 是假定所有数的计算方法都和素数一样时的函数 F ,比如 $F(x)=\varphi(x), f(x)=x-1$.根据我们对筛法的要求, f 就会是一个完全积性函数.
- 要求的式子就是 $g(n,cnt)$,g 里面只有素数的函数值,所以和原式是相等的.
- 而 n 以内的数最小质因子显然不会超过 $\sqrt{n}$,所以也可以写成 $g(n,m), p_m^2 \geq n$.
- 考虑 $g(n,j)$ 的递推计算.若 $p_j^2 \geq n$,则 $g(n,j)=g(n,j-1)$.因为不会有 $R(i) > p_j$ 的数.否则,

$$g(n,j) = g(n,j-1) - f(p_j) \cdot (g(\frac{n}{p_j}, j-1) - \sum_{k=1}^{j-1} f(p_k))$$

- 上面这个式子什么意思呢?考虑 $g(n,j-1)$ 与 $g(n,j)$ 的差别.根据 g 的定义,不难看出, $g(n,j)$ 比 $g(n,j-1)$ 少的部分,就是那些最小质因子为 p_j 的合数的函数值.所以我们应该把它们减去.
- 这些数都有质因子 p_j ,可以提到括号外面,而 f 是完全积性的,所以就是 $f(p_j)$ 乘上括号内的函数值之和.括号内的数显然不超过 $\frac{n}{p_j}$,并且最小质因子 $\geq p_j$,根据 g 的定义,括号内函数值就是 $g(\frac{n}{p_j}, j-1)$?
- 但是 g 也会统计素数处的函数值, $p_j \cdot p_1, p_j \cdot p_2, \dots, p_j \cdot p_{j-1}$ 这些数(注意不会有 p_j ,因为 g 从 2 开始枚举),在提出 p_j 之后,本不应该被计入(它们的最小质因子是 $p_1, p_2 \dots p_{j-1}$),却被计入了贡献(提出 p_j 后为素数),所以应该将它们减去.
- 于是就得到了 g 的递推计算式.而我们可以将 g 的空间复杂度压到 $O(\sqrt{n})$.
- 第一维,会被用到的其实只有 $\frac{n}{1}, \frac{n}{2}, \dots \frac{n}{n}$ 这些位置.观察递推式,第一维的转移只有 $n \to \frac{n}{p_j}$,而向下取整的除法某种意义上是可结合的,即, $\frac{\frac{x}{a}}{b} = \frac{x}{ab}$,无论怎样除,我们只需要 $g(n,m)$,所以只会用到那 $O(\sqrt{n})$ 个关键位置.
- 第二维,转移只有 $j \to j-1$,而于此同时第一维也在缩小,因为我们只需要 $g(n,m)$,所以可以直接压掉第二维.
- 举个例子, $F(x)=1$, $g(n,m)$ 即求 n 以内的质数个数.下面是部分代码.

```
int sqN=ceil(sqrt(n));
for(int i=1;i<=tot;++i)
    g[i]=w[i]-1;//g(w[i],0)
```

```
for(int j=1;j<=cnt;++j)
    for(int i=1;i<=tot && prime[j]*prime[j]<=w[i];++i)
    {
        int k=w[i]/prime[j];
        if(k<=sqn)
            k=id1[k];
        else
            k=id2[n/k];
        g[i]-=g[k]-j+1;
    }
cout<<g[1]<<endl;
```

- 在代码中, tot 表示关键点总数, $w[i]$ 表示从大到小的第 i 个关键点. $g[i]$ 表示当前的 $g(w[i], j)$,$id_1[k], id_2[k]$ 分别表示 $k, \frac{n}{k}$ 是第几个关键点(把空间降下来).于是就完成了 $\sum_{x=1}^{n} F(x)[x \in Prime]$ 的计算.
- 这一步的时间复杂度是 $O(\frac{n^{\frac{3}{4}}}{log_2 n})$ 的.

$Part\ 2$

- 现在我们想求 $F(x)$ 在所有位置函数值之和 $ans = \sum_{i=1}^{n} F(i)$.设一个二元函数 S ,

$$S(n, j) = \sum_{i=2}^{n} F(i)[R(i) \geq p_j]$$

- 那么 $ans = S(n, 1) + F(1)$.
- 考虑将 S 的计算拆成素数部分与合数部分.素数部分显然是 $g(n, cnt) - \sum_{i=1}^{j-1} F(p_i)$.
- 合数部分,我们枚举一批数的最小质因子为 p_k ,幂次为 e ,那么就可以不重不漏计算 $R(i) \geq p_j$ 的合数.
- 因为我们还枚举了幂次,所以将 p_k^e 提出来后,这批数就都没有质因子 p_k 了,函数值直接与括号外相乘.
- 注意所有的 $p_k^e, e \geq 2$ 作为合数,在这样枚举时都没有被计入,所以要加上.

$$S(n, j) = g(n, cnt) - \sum_{i=1}^{j-1} F(p_i) + \sum_{k=j}^{p_k^2 \leq n} \sum_{e=1}^{p_k^{e+1} \leq n} F(p_k^e) \cdot S(\frac{n}{p_k^e}, k+1) + F(p_k^{e+1})$$

- 不用记忆化,直接递归计算,边界是 $n = 1 \vee p_j > n, S(n, j) = 0$.
- 时间复杂度 $O(\frac{n^{\frac{3}{4}}}{log_2 n})$.