

数论 4

Min_25筛

算法简介

一种 $O\left(\frac{n^{\frac{3}{4}}}{\log n}\right)$ 计算积性函数 $F(x)$ 前缀和的算法。

算法思路

首先给定 Min_{25} 筛的适用条件 $F(p)$ 的值可以拆分为若干个完全积性函数，且 $F(p^k)$ 可以快速计算。

设 $\text{midv}(x)$ 表示 x 的最小素因子。将所有素数从小到大排列，记为 $p_1, p_2, p_3 \cdots$

考虑构造完全积性函数 $f(x)$ 满足 $f(p) = F(p)$ 设 $g(n, k) = \sum_{i=1}^n [\text{midv}(i) > p_k] \text{ or } i \in \text{prime}] f(i)$

考虑从 $g(n, k-1)$ 转移到 $g(n, k)$ 等价于减去 $\text{midv}(i) = p_k$ 且 $i \notin \text{prime}$ 的 $f(i)$ 的贡献。如果 $p_k^2 \geq n$ 则这样的数不存在。

否则将所有满足该条件的 i 提取出一个 p_k 记 $i' = \frac{i}{p_k}$ 于是 $\text{midv}(i') \geq p_k, f(i) = \frac{f(i')}{f(p_k)}$

考虑减去 $f(p_k) g(\lfloor \frac{n}{p_k} \rfloor, k-1)$ 发现 $g(\lfloor \frac{n}{p_k} \rfloor, k-1)$ 多包含了 $\sum_{i=1}^{k-1} f(p_i)$ 于是再补上。

于是有状态转移方程

$$g(n, k) = \begin{cases} g(n, k-1), & p_k^2 \geq n \\ g(n, k-1) - f(p_k) g(\lfloor \frac{n}{p_k} \rfloor, k-1) - \sum_{i=1}^{k-1} f(p_i), & p_k^2 < n \end{cases}$$

由于 $\lfloor \frac{\lfloor \frac{n}{a} \rfloor}{b} \rfloor = \lfloor \frac{n}{ab} \rfloor$ 于是 $g(a, b)$ 中的 a 只有 $O(\sqrt{n})$ 个。

使用刷表法暴力转移上述方程直到 $p_{k+1}^2 > n$ 此时得到 $g(a, k) = \sum_{i=1}^a [i \in \text{prime}] f(i) = \sum_{i=1}^a [i \in \text{prime}] F(i), a \in \lfloor \frac{n}{p_k} \rfloor$

不妨将此时的 $g(n, k)$ 记为 $g(n)$ 由于玄学因素，该过程的时间复杂度为 $O\left(\frac{n^{\frac{3}{4}}}{\log n}\right)$

接下来设 $S(n, k) = \sum_{i=1}^n [mdiv(i) > p_k] F(i)$ 于是目标就是求 $S(n, 0) + F(1) = S(n, 0) + 1$ (积性函数必有 $F(1) = 1$)

将 $S(n, k)$ 的和分为 $\sum_{[i \in \text{prime}, i > p_k]} S(i)$ 和 $\sum_{[i \notin \text{prime}, \text{midv}(i) > p_k]} S(i)$ 两部分。

前面部分有 $\sum_{[i \in \text{prime}, i > p_k]} S(i) = g(n) - \sum_{i=1}^k F(p_i)$ 后面部分可以枚举最小素因子及其幂次，可得状态转移方程

$$S(n,k) = \sum_{i=1}^k F(p_i) + \sum_{p_i^j \leq n, j > k} F(p_i^j) \left(S\left(\left\lfloor \frac{n}{p_i^j} \right\rfloor, i\right) + [j \neq 1] \right)$$

最后补上 $F(p_i^j)[j \neq 1]$ 是因为 $F(p_i^j)S\left(\left\lfloor \frac{n}{p_i^j} \right\rfloor, i\right)$ 不包含 $F(p_i^j)$ 的贡献，但 $F(p_i)$ 的贡献已经计算。

由于 $p_i^2 > n$ 时 $\sum_{p_i^j \leq n, j > k} F(p_i^j) \left(S\left(\left\lfloor \frac{n}{p_i^j} \right\rfloor, i\right) + [j \neq 1] \right) = 0$ 于是只需要枚举 $O\left(\frac{\sqrt{n}}{\log n}\right)$ 个素数即可。

由于玄学因素，该过程不需要记忆化且时间复杂度为 $O\left(\frac{n^{\frac{3}{4}}}{\log n}\right)$

算法例题

例题一

[洛谷p5325](#)

题意

给定积性函数 $F(p_k) = p_k(p_k - 1)$ 计算 F 前 n 项和。

题解

构造完全积性函数 $f_1(x) = x^2, f_2(x) = x$ 于是可以计算出 $g_1(n), g_2(n)$

然后令 $g(n) = g_1(n) - g_2(n) = \sum_{i=1}^n \sum_{i \in \text{prime}} F(i)$ 即可计算出 $S(n, 0)$

```
const int MAXN=1e5+5,Mod=1e9+7,inv2=500000004,inv6=166666668;
namespace Min_25{
    LL n,blk[1];
    int sqr,vis[MAXN],prime[MAXN],p_cnt;
    int sp1[MAXN],sp2[MAXN],sp[MAXN],g1[1],g2[1],g[1];
    int b_cnt,idx1[MAXN],idx2[MAXN];
    int S(LL a,int b){
        if(a<=prime[b])return 0;
        int pos=(a<=sqr)?idx1[a]:idx2[n/a];
        int ans=(g[pos]-sp[b])%Mod;
        for(int i=b+1;i<=p_cnt&&1LL*prime[i]*prime[i]<=a;i++){
            LL pow_p=prime[i];
            for(int j=1,mod_p;pow_p<=a;j++){
                mod_p=pow_p%Mod;
                ans=(ans+1LL*mod_p*(mod_p-1)%Mod*(S(a/pow_p,i)+(j!=1)))%Mod;
                pow_p*=prime[i];
            }
        }
    }
}
```

```

        return (ans+Mod)%Mod;
    }
    int cal(LL n){
        Min_25::n=n;
        sqr=sqrt(n+0.5);
        for(int i=2;i<=sqr;i++){
            if(!vis[i])prime[++p_cnt]=i;
            for(int j=1;j<=p_cnt&& i*prime[j]<=sqr;j++){
                vis[i*prime[j]]=1;
                if(i%prime[j]==0)break;
            }
        }
        _rep(i,1,p_cnt){
            sp1[i]=(sp1[i-1]+1LL*prime[i]*prime[i])%Mod;
            sp2[i]=(sp2[i-1]+prime[i])%Mod;
            sp[i]=(sp1[i]-sp2[i]+Mod)%Mod;
        }
        for(LL lef=1,rig=0;lef<=n;lef=rig+1){
            rig=n/(n/lef);
            blk[++b_cnt]=n/lef;
            int temp=blk[b_cnt]%Mod;
            g1[b_cnt]=(1LL*temp*(temp+1)%Mod*(temp<<1|1)%Mod*inv6+Mod-1)%Mod;
            g2[b_cnt]=(1LL*temp*(temp+1)%Mod*inv2+Mod-1)%Mod;
            if(blk[b_cnt]<=sqr)
                idx1[blk[b_cnt]]=b_cnt;
            else
                idx2[rig]=b_cnt;
        }
        _rep(i,1,p_cnt){
            LL pow_p=1LL*prime[i]*prime[i];
            for(int j=1;j<=b_cnt&&blk[j]>=pow_p;j++){
                LL pos=blk[j]/prime[i];
                pos=(pos<=sqr)?idx1[pos]:idx2[n/pos];
                g1[j]=(g1[j]-1LL*prime[i]*prime[i]%Mod*(g1[pos]-
                sp1[i-1]))%Mod;
                g1[j]=(g1[j]+Mod)%Mod;
                g2[j]=(g2[j]-1LL*prime[i]*(g2[pos]-sp2[i-1]))%Mod;
                g2[j]=(g2[j]+Mod)%Mod;
            }
        }
        _rep(i,1,b_cnt)
            g[i]=(g1[i]-g2[i]+Mod)%Mod;
        return (S(n,0)+1)%Mod;
    }
}
int main()
{
    LL n=read_LL();
    enter(Min_25::cal(n));
    return 0;
}

```

```
}
```

例题二

LOJ5325

题意

给定积性函数 $F(p_k)=p_k \oplus k$ 计算 F 前 n 项和。

题解

发现 $F(p)=p-1, p \geq 2$ 于是先求出 $g(n)=\sum_{i=1}^n \text{prime}(i)(p-1)$ 然后修改一下 $F(2)$ 处的误差。

最后套 `Min_25` 筛即可。

```
const int MAXN=1e5+5,Mod=1e9+7,inv2=500000004;
namespace Min_25{
    LL n,blk[ $\text{MAXN} \ll 1$ ];
    int sqr,vis[ $\text{MAXN}$ ],prime[ $\text{MAXN}$ ],p_cnt;
    int sp[ $\text{MAXN}$ ],g1[ $\text{MAXN} \ll 1$ ],g2[ $\text{MAXN} \ll 1$ ],g[ $\text{MAXN} \ll 1$ ];
    int b_cnt,idx1[ $\text{MAXN}$ ],idx2[ $\text{MAXN}$ ];
    int S(LL a,int b){
        if(a<=prime[b])return 0;
        int pos=(a<=sqr)?idx1[a]:idx2[n/a];
        int ans=(g[pos]-sp[b])%Mod;
        for(int i=b+1;i<=p_cnt&&1LL*prime[i]*prime[i]<=a;i++){
            LL pow_p=prime[i];
            for(int j=1;pow_p<=a;j++){
                ans=(ans+1LL*(prime[i]^j)%Mod*(S(a/pow_p,i)+(j!=1)))%Mod;
                pow_p*=prime[i];
            }
        }
        return (ans+Mod)%Mod;
    }
    int cal(LL n){
        Min_25::n=n;
        sqr=sqrt(n+0.5);
        for(int i=2;i<=sqr;i++){
            if(!vis[i])prime[++p_cnt]=i;
            for(int j=1;j<=p_cnt&&i*prime[j]<=sqr;j++){
                vis[i*prime[j]]=1;
                if(i%prime[j]==0)break;
            }
        }
    }
}
```

```

    }
    _rep(i, 1, p_cnt)
    sp[i] = (sp[i-1] + prime[i]) % Mod;
    for (LL lef = 1, rig = 0; lef <= n; lef = rig + 1) {
        rig = n / (n / lef);
        blk[++b_cnt] = n / lef;
        int temp = blk[b_cnt] % Mod;
        g1[b_cnt] = (1LL * temp * (temp + 1) % Mod * inv2 + Mod - 1) % Mod;
        g2[b_cnt] = (temp + Mod - 1) % Mod;
        if (blk[b_cnt] <= sqr)
            idx1[blk[b_cnt]] = b_cnt;
        else
            idx2[rig] = b_cnt;
    }
    _rep(i, 1, p_cnt) {
        LL pow_p = 1LL * prime[i] * prime[i];
        for (int j = 1; j <= b_cnt && blk[j] >= pow_p; j++) {
            LL pos = blk[j] / prime[i];
            pos = (pos <= sqr) ? idx1[pos] : idx2[n / pos];
            g1[j] = (g1[j] - 1LL * prime[i] * (g1[pos] - sp[i-1])) % Mod;
            g1[j] = (g1[j] + Mod) % Mod;
            g2[j] = (g2[j] - (0LL + g2[pos] - (i-1))) % Mod;
            g2[j] = (g2[j] + Mod) % Mod;
        }
    }
    _rep(i, 1, p_cnt)
    sp[i] = (sp[i] - i + 2 + Mod) % Mod;
    _for(i, 1, b_cnt)
    g[i] = (g1[i] - g2[i] + 2 + Mod) % Mod;
    return (S(n, 0) + 1) % Mod;
}
}
int main()
{
    LL n = read_LL();
    enter(Min_25::cal(n));
    return 0;
}

```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E6%95%B0%E8%AE%BA_4&rev=1600735847

Last update: 2020/09/22 08:50