min_25筛的推导/实现与复杂度分析.

这是(af)oier spinach退役前的笔记与解题记录,不能保证内容的严谨性,仅供参考

如果您发现文章存在可能的错误,请立刻联系我.

作者正在学文化课,不能保证及时修正,抱歉.

参考文献:

1. zzt: 2018集训队论文.
2. zzt: WC2019讲课课件.
3. zzq blog
4. zhb2047 blog
5. xehoth blog
6. cz_xuyixuan blog
7. 简单易懂的质数筛法-过时数论魔法.ppt,一个APIO的讲课内容,出门左转UOJ用户群.

min_25筛

> min_25提出.复杂度$O(n^{1-\epsilon})$.是近线性复杂度.
>
> 在$n \leq 10^{13}$时可以认为是$O(\frac{n^{\frac{3}{4}}}{log\ n})$的.
>
> 是OI中最好用的筛法(洲阁筛常数太大...而且不好写不好推导...可以说只有理论价值了).

例子

$$n=\prod_{i=1}^{m}{p_i}^{c_i}$$

$$\phi(n,d)=\prod_{i=1}^{m}(p^{c_i}+d)$$

$$\phi(1,d)=1$$

$$\phi(p)=G(p)=p+d$$

$$\phi(p^c)=T(p,c,d)=p^c+d$$

推导

求$S(n)=\sum_{i=1}^{n}f(i)$.

- 满足$f(p^c)$是一个关于p,c的**低次**多项式多项式.
- $f(n)$是积性函数

考虑 提取最小质因子 加速计算.

$$\sum_{i=1}^{n} f(i) = 1 + \sum_{\substack{2 \le p^c \le n \\ p \in prime}} f(p^c)(1 + \sum_{\substack{2 \le x \le \lfloor \frac{n}{p^c} \rfloor \\ minprime_x > p}} f(x))$$

对于合数x必定有.$minprime_x \le \sqrt{x} \le \sqrt{n}$继续分类.

$$1 + \sum_{\substack{2 \le p^c \le n \\ p \in prime \\ p \le \sqrt{n}}} f(p^c)(1 + \sum_{\substack{minprime_x > p \\ 2 \le x \le \lfloor \frac{n}{p^c} \rfloor}} f(x)) + \sum_{\substack{p \in prime \\ \sqrt{n} < p \le n}} f(p)$$

构造辅助函数$g_{n,m}$与h_n定义如下:

- $g_{n,m}$为n以内,最小质因子大于m的贡献.
- h_n为n以内素数的贡献.

$$g_{n,m} = \sum_{\substack{minprime_x > m \\ 2 \le x \le n}} f(x)$$

$$h_n = \sum_{\substack{p \in prime \\ 2 < p \le n}} f(p)$$

寻找g的递归式

$$g_{n,m} = \sum_{\substack{minprime_x > m \\ 2 \le x \le n}} f(x)$$

$$= \sum_{\substack{p^c \le n \\ p \in prime \\ m < p \le \sqrt{n}}} f(p^c)(1 + \sum_{\substack{minprime_x > p \\ 2 \le x \le \lfloor \frac{n}{p^c} \rfloor}} f(x)) + \sum_{\substack{p \in prime \\ \sqrt{n} < p \le n}}$$

$$= \sum_{\substack{p^c \le n \\ p \in prime \\ m < p \le \sqrt{n}}} f(p^c)(1 + g_{\lfloor \frac{n}{p^c} \rfloor, p}) + h_n - h_{\sqrt{n}}$$

根据定义$\sum_{i=1}^{n} f(i) = S(n) = g_{n,1} + 1$,如果求出了所有需要用到的$h_m(\exists i \quad \lfloor \frac{n}{i} \rfloor = m)$即可递归求出$g$

解释:观察递归式,根据$\lfloor \frac{\lfloor \frac{n}{a} \rfloor}{b} \rfloor = \lfloor \frac{n}{ab} \rfloor$用到的$h_m$必然是$\lfloor n/i \rfloor$的形式.或者有$m \le \sqrt{n}$,需要求解的$h_m$是$O(\sqrt{n})$量级的

如果求出了h(素数对于前缀和的贡献).那么暴力递归就能求出g了,这里并不需要记忆化,记忆化了也不能优化复杂度.

$$h_n = \sum_{\substack{p\in prime\\ 2<p\le n}} f(p)$$

由于$f(p^c)$是$G(p,c)$,一个低次多项式,把$f(p)$拆成$\sum_{i=0}^{k} c_i p^i$的形式.我们只需要计算这个.(素数k次幂和)

$$h_n = \sum_{\substack{p\in prime\\ 2<p\le n}} p^k$$

定义

$$L_{n,m} = \sum_{i=2}^{n} [(i\in prime)\ or\ (minprime_i > prime_m)]\ i^k$$

即范围$(1,n]$ 内满足质数k次幂与,$minprime_x > p_m$(最小质因子足够大,大于第m个素数)的 数的k次幂和.

也可以说是埃氏筛第m轮筛除后剩余的数字的k次方和.

考虑一个素数筛的过程(欧拉筛的过程,用最小质因子进行筛除)

解释:考虑最小质因子为p_m的数$x = p_m\cdot y \le n \Rightarrow y\le \lfloor \frac{n}{p_m} \rfloor$

但是$p_m^k\ L_{\lfloor \frac{n}{p_m} \rfloor,m-1}$里面包含了满足$x = q\cdot p_m \le n, minprime_q < p_m$ 的x,它们并不应该在这一轮中筛掉,而应该在之前用更小的质数筛除.

再考虑这种q,必定是$q = p_i < p_m, p_i\in prime$的,不然$q$已经被筛掉了.所以补上一个$pre_{m-1}$,这样找到的$x = p_m\cdot y$就是我们本轮需要筛除的数了.

这样做扣除且仅扣除了所有满足$x\notin prime, x\le n, minprime_x = p_m$的$x$所以筛除的正确性得到了保证.

$$L_{n,m} = L_{n,m-1} - p_m^k(L_{\lfloor \frac{n}{p_m} \rfloor,m-1} - pre_{m-1})$$

$$pre_m = \sum_{i=1}^{m} prime_i^k$$

如果$p_m^2 > n$了那么不需要转移...有$L_{n,m} = L_{n,m-1}$理由如下.

设最小质因子大于p_j的最小合数为x必定有$x\ge p_j^2 > n$使用p_j不能再不重复地筛掉$[1,i]$内的非质数.(可以考虑埃试筛的过程.这时,用p_j去筛$[1,n]$中的倍数.这些合数已经被更小的质数筛过了,应该从p_j^2开始筛)

复杂度

杜教筛的复杂度.

$$\sum_{i=1}^{n}(f*g)(i)=\sum_{i=1}^{n}\sum_{d|i}f(d)g(\frac{i}{d})=\sum_{d=1}^{n}f(d)\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}g(i)$$

$$let\ S(n)=\sum_{i=1}^{n}g(i)$$

$$\sum_{i=1}^{n}(f*g)(i)=\sum_{d=1}^{n}f(d)S(\lfloor\frac{n}{d}\rfloor)$$

$$S(n)=\sum_{i=1}^{n}(f*g)(i)-\sum_{d=2}^{n}f(d)S(\lfloor\frac{n}{d}\rfloor)$$

由于采用了记忆化搜索,每个状态只会向下递归一次,我们可以考虑总转移次数来估计复杂度.即考虑每个状态会依赖多少状态,这会导致算法到hashmap中查询之前的计算结果.考虑x依赖$S_x=\{d_1,d_2,d_3\dots d_n\}$.那么对复杂度的贡献为$|S_x|$,而$S_x$是$\{\lfloor\frac{x}{d}\rfloor\mid d\le n,d\in N\}$从而$|S_x|=O(\sqrt{x})$.然后进行分类计算,$d,\lfloor\frac{n}{d}\rfloor$中至少有一个不超过$\sqrt{n}$,枚举这个数进行计算.

$$T(n)=\sum_{i=1}^{\sqrt{n}}\sqrt{i}+\sum_{i=1}^{\sqrt{n}}\sqrt{\frac{n}{i}}\le\int_0^{\sqrt{n}}\sqrt{x}dx+\int_0^{\sqrt{n}}\sqrt{\frac{n}{x}}dx$$

$$\int_0^{\sqrt{n}}\sqrt{x}dx+\int_0^{\sqrt{n}}\sqrt{\frac{n}{x}}dx=O(n^{\frac{1}{2}\times\frac{3}{2}})+O(n^{\frac{1}{2}}\times n^{\frac{1}{2}\times\frac{1}{2}})=O(n^{\frac{3}{4}})$$

筛法求h_n的复杂度分析.这里由于涉及了对数积分,比较复杂,我们进行一些估计并且利用Wolfram Alpha来计算.

$$\pi(n)=\sum_{x\le n}[x\in prime]=O(\frac{n}{log\,n})$$

$$T(n)=\sum_{i=1}^{\sqrt{n}}\pi(\sqrt{i})+\sum_{i=1}^{\sqrt{n}}\pi(\sqrt{\frac{n}{i}})$$

$$\int_0^{\sqrt{n}}\frac{x}{log\,x}dx+\int_0^{\sqrt{n}}\frac{\sqrt{\frac{n}{x}}}{\log\sqrt{\frac{n}{x}}}=O(\frac{n^{\frac{3}{4}}}{log\,n})$$

- 筛出h_n的复杂度为$O(\frac{n^{\frac{3}{4}}}{log\,n})$.
- DFS计算g这部分是玄学的$O(n^{1-\epsilon})$近线性复杂度,不过$n\le10^{13}$时可以认为是$O(\frac{n^{\frac{3}{4}}}{log\,n})$

代码实现

如何正确地实现`Sieve of Eratosthenes`,`Euler's Sieve`

```
int vis[N],prime[N],cnt;

void sieve(int n){
    vis[1]=1;
    for(int i=2;i*i<=n;i++) if(!vis[i]){
        prime[++cnt]=i;
        for(int j=i*i;j<=n;j+=i) vis[j]=1;
    }
}
```

```
int vis[N],prime[N],cnt,minp[N],pk[N];
void sieve(int n){
    vis[1]=1;
    for(int i=2;i<=n;i++){
        if(!vis[i]){
            prime[++cnt]=i;
            minp[i]=i;
            pk[i]=1;
        }
        for(int j=1;j<=cnt&&i*prime[j]<=n;j++){
            vis[i*prime[j]]=1;
            minp[i*prime[j]]=prime[j];
            if(i%prime[j]==0){
                pk[i*prime[j]]=pk[i]+1;
                break;
            }
            pk[i*prime[j]]=1;
        }
    }
}
```

预处理$\sqrt{n}$以内的质数与$pre_x=\sum_{i=1}^{x}prime_x^k$.

找到$m,\exists i\ \lfloor\frac{n}{i}\rfloor=m$,这些m的h是需要计算的.共$O(\sqrt{n})$项,初始化$h(m)=\sum_{i=2}^{m}i^k$

从小到大枚举$prime_j$,从m=n开始到$m\geq p_j^2$的$m=\lfloor\frac{n}{d}\rfloor$向下枚举,进行筛除操作
$h(m)=h(m)-p_j^k(h(\lfloor\frac{m}{p_j}\rfloor)-pre_{m-1})$

$\sum_{i=1}^{n} f(i) = g_{n,1} + 1$.进入递归求解,这里不需要记忆化.

下面是 `SPOJ DIVCNT K` 的代码,朴素实现,没有使用上面提到的技巧去掉hashmap,也没有什么常数优化.

之后会写求更复杂度的积性函数前缀和以及部分并非积性的函数(典型的是最大/最小/次大次小质因子)

```
#include <iostream>
#include <cstdio>
#include <ctime>
#include <algorithm>
#include <unordered_map>
#include <bitset>
using namespace std;
typedef unsigned long long Int;
const int N=10000000+10;
int vis[N],prime[N],cnt,pre[N];
inline void init(){
    for(int i=2;i<N;i++){
        pre[i]=pre[i-1]+(vis[i]==0);
        if(!vis[i]) prime[++cnt]=i;
        for(int j=1;j<=cnt&&1LL*i*prime[j]<N;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0) break;
        }
    }
}
namespace SPINACH{
    Int n,k,h[N],m;
    unordered_map<Int,int> id;
    inline Int geth(Int x){ return (k+1)*(x<N?pre[x]:h[id[x]]); }
    Int g(Int n,Int m){
        m++; if(prime[m]>n) return 0;
        Int s=0;
        while(m<=cnt&&1LL*prime[m]*prime[m]<=n){
            Int p=prime[m],pc=p,c=1;
            while(pc<=n){
                s+= (k*c+1)*(1+g(n/pc,m));
                pc=pc*p;c++;
            }
            m++;
        }
        return s+(geth(n)-geth(prime[m-1]));
    }
    Int solve(Int n,Int _k){ id.clear();
        k=_k;
```

```
        Int l=1,q=0;m=0;
        while(l<=n){
            q=n/l; l=n/q+1;
            h[id[q]=++m]=(q-1);
        }
        for(int i=1;i<=cnt&&1LL*prime[i]*prime[i]<=n;i++){
            Int p=prime[i]; l=1;q=0;m=0;
            while(l<=n){
                q=n/l; l=n/q+1;
                if(p*p>q) break;
                h[++m]-=(h[id[q/p]]-1LL*(i-1));
            }
        }
        return 1+g(n,0);
    }
}
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    init();
    int T;cin>>T; Int n,k;
    while(T--){
        cin>>n>>k;
        cout<<SPINACH::solve(n,k)<<'\n';
    }
    return 0;
}
```

这里是使用了小技巧去掉hashmap的写法.效率足够对付大多数题目.多组询问记得遍历$\lfloor \frac{n}{i} \rfloor$清空.

$$at(x)=\begin{cases} pre_0(x) & x\le \sqrt{n} \\ pre_1(x) & x>\sqrt{n} \end{cases}$$

```
#include <iostream>
#include <cstdio>
#include <ctime>
#include <algorithm>
#include <bitset>
using namespace std;
typedef unsigned long long Int
const int N=1000000+10;
int vis[N],prime[N],cnt,pre[N];
inline void init(){
    for(int i=2;i<N;i++){
        pre[i]=pre[i-1]+(vis[i]==0);
        if(!vis[i]) prime[++cnt]=i;
        for(int j=1;j<=cnt&&1LL*i*prime[j]<N;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0) break;
        }
```

```
    }
}
namespace SPINACH{
    Int n,k,h[N];
    Int pre0[N],pre1[N];
    inline Int& at(Int x){ return (x*x<=n)?pre0[x]:pre1[n/x]; }
    inline Int geth(Int x){ return (k+1)*(x<N?pre[x]:at(x)); }
    Int g(Int n,Int m){
        m++; if(prime[m]>n) return 0;
        Int s=0;
        while(m<=cnt&&1LL*prime[m]*prime[m]<=n){
            Int p=prime[m],pc=p,c=1;
            while(pc<=n){
                s+= (k*c+1)*(1+g(n/pc,m));
                pc=pc*p;c++;
            }
            m++;
        }
        return s+(geth(n)-geth(prime[m-1]));
    }
    Int solve(Int _n,Int _k){
        n=_n; k=_k;
        Int l1,q=0;
        while(l<=n){
            q=n/l; l=n/q+1;
            at(q)=(q-1);
        }
        for(int i=1;i<=cnt&&1LL*prime[i]*prime[i]<=n;i++){
            Int p=prime[i]; l=1;q=0;
            while(l<=n){
                q=n/l; l=n/q+1;
                if(p*p>q) break;
                at(q)-=(at(q/p)-1LL*(i-1));
            }
        }
        return 1+g(n,0);
    }
}
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    init();
    int T;cin>>T; Int n,k;
    while(T--){
        cin>>n>>k;
        cout<<SPINACH::solve(n,k)<<'\n';
    }
    return 0;
}
```

例题

- SPOJ divcnt

  系列题目,divcnt即为(counting divisor),约数个数计数问题.有四题,均为min_25提供.

  $\sum_{i=1}^{n}\sigma_0(i)\quad n\leq 2^{64}$

  $\sum_{i=1}^{n}\sigma_0(i^2)\quad n\leq 10^{12}$

  $\sum_{i=1}^{n}\sigma_0(i^3)\quad n\leq 10^{11}$

  $\sum_{i=1}^{n}\sigma_0(i^k)\quad n\leq 10^{10}$

  第一个是Stern Brocot Tree的题,具体做法可以在 参考文献1 中学习,是一个典型的论文题.

  第二个和第三个,是推式子+反演+卷积优化求和(杜教筛),但是可以使用本文提供的min_25筛通过.

  最后一个是min_25筛的板子题,事实上,这个题也有$O(n^{\frac{2}{3}})$的做法,不过国内没有参考资料,可以自行找min_25提供的算法或者查阅相关论文学习.

- LOJ6053 简单的函数

  给定一个积性函数$f(p^c)=p\oplus c$,求$\sum_{i=1}^{n}f(i)\quad n\leq 10^{10}$,那个$\oplus$是位运算xor

  $$f(p)=\begin{cases}p+1 & p=2\\ p-1 & p>2\end{cases}$$
  $$f(p^c)=p\oplus c$$

  直接套公式(雾)

小心爆long long...小心爆ull,小心爆int128...我这份代码重写三次才过的.

```cpp
#include <iostream>
#include <algorithm>
#include <cmath>
using namespace std;
typedef unsigned long long UInt;
typedef long long Int;
const int N=300000+10;
const Int mod=(Int)(1e9)+7LL;
Int qpow(Int a,Int p){
    if(p==0) return 1;
    Int r=qpow(a,p>>1);
    r=r*r%mod;
    return (p&1)?(r*a%mod):r;
}
Int inv2;
UInt n;
int vis[N],prime[N],cnt;
Int pre0[N],pre1[N];
UInt qwq[N];
double SQRTN=0;
void init(){
    for(int i=2;i<N;i++){
        pre0[i]=pre0[i-1];
        pre1[i]=pre1[i-1];
```

```
        qwq[i]=qwq[i-1];
        if(!vis[i]){
            prime[++cnt]=i;
            pre0[i]++;
            pre1[i]=(pre1[i]+i)%mod;
        }
        for(int j=1;j<=cnt&&prime[j]*i<N;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0) break;
        }
    }
}
// f(p) = p-1 (p>2)
// f(p=2) = (p=2)+1
Int pre0A[N],pre1A[N];
Int pre0B[N],pre1B[N];
inline Int& at(UInt x,int y){ // x*x<=n,overflow uint64!
    if(x<=SQRTN) return y?pre1A[x]:pre0A[x];
    return y?pre1B[n/x]:pre0B[n/x];
}
inline Int geth(UInt x){
    Int ret=2*(x>=2);
    if(x<N) ret+=( (pre1[x]-pre0[x])%mod );
    else ret+=( (at(x,1)-at(x,0))%mod );
    return (ret%mod+mod)%mod;
}
Int g(UInt n,int m){
    m++; if(1ULL*prime[m]>n) return 0;
    Int s=0;
    while(m<=cnt&&1ULL*prime[m]*prime[m]<=n){
        UInt p=prime[m],pc=p,c=1;
        while(pc<=n){
            s=(s+((p^c)%mod)*(1+g(n/pc,m))%mod)%mod;
            c++;pc=pc*p;
        }
        m++;
    }
    Int tmp=( (geth(n)-geth(prime[m-1]))%mod+mod)%mod;
    return (s+tmp)%mod;
}
inline Int s1(Int x){ x%=mod; return x*(x+1)%mod*inv2%mod; }
inline Int sub(Int a,Int b){
    a=(a%mod+mod)%mod;
    b=((-b)%mod+mod)%mod;
    return (a+b)%mod;
}
inline Int mul(Int a,Int b){
    a=(a%mod+mod)%mod;
    b=(b%mod+mod)%mod;
    return a*b%mod;
}
Int solve(){
    UInt l=1,q=0;
```

```
    while(l<=n){
        q=n/l; l=n/q+1;
        at(q,0)=(q-1);
        at(q,1)=(s1(q)+mod-1)%mod;
    }
    for(int i=1;i<=cnt&&1ULL*prime[i]*prime[i]<=n;i++){
        l=1;q=0;Int p=prime[i];
        while(l<=n){
            q=n/l; l=n/q+1;
            if(p*p>q) break;
            //at(q,0)-=(at(q/p,0)-(i-1));
            //at(q,1)-=p*(at(q/p,1)-pre1[i-1]);
            at(q,0)=sub(at(q,0),sub(at(q/p,0),pre0[p-1]));
            at(q,1)=sub(at(q,1),mul(p,sub(at(q/p,1),pre1[p-1])));
        }
    }
    Int ret=1+g(n,0);
    return ret%mod;
}

int main(){
    cin>>n; SQRTN=sqrt(n);
    inv2=qpow(2,mod-2);
    init();
    cout<<solve()<<endl;
    return 0;
}
```

- UOJ Round 13,C:sanrd

  求$\sum_{i=1}^{n} f(i)$,其中$f(i)=second_max(\{p\cdot k\})$即质因子可重集合的次大元素.定义$f(p)=0,f(1)=0$

```
#include <iostream>
#include <algorithm>
#include <ctime>
#include <cmath>
#include <cassert>
using namespace std;
typedef long long Int;
const int N=1000000+10;
int vis[N],prime[N],cnt;
void init(){
    for(int i=2;i<N;i++){
        if(!vis[i]) prime[++cnt]=i;
        for(int j=1;j<=cnt&&i*prime[j]<N;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0) break;
        }
```

```
    }
}
Int pre0[N],pre1[N],n;
Int& at(Int m){ return 1.0*m*m<=n?pre0[m]:pre1[n/m]; }
Int g(Int n,int m){
    m++; if(n<=2||prime[m]>n) return 0;
    Int s=0;
    while(prime[m]*prime[m]<=n&&m<=cnt){
        Int p=prime[m],c=1,pc=p;
        while(pc*p<=n){
            s+=(g(n/pc,m)+p*(at(n/pc)-(m-1)));
            pc=pc*p;c++;
        }
        m++;
    }
    return s;
}
Int solve(Int _n){
    ::n=_n;
    Int l=1,q=0;
    while(l<=n){
        q=n/l; l=n/q+1;
        at(q)=q-1;
    }
    for(int i=1;i<=cnt&&1LL*prime[i]*prime[i]<=n;i++){
        Int p=prime[i];
        l=1; while(l<=n){
            q=n/l; l=n/q+1;
            if(p*p>q) break;
            at(q)-=(at(q/p)-(i-1));
        }
    }
    return g(n,0);
}
int main(){
    init();
    Int l,r;cin>>l>>r;
    cout<<solve(r)-solve(l-1)<<endl;
    return 0;
}
```

UR 13,sanrd现在还不懂为什么....

陈牧歌@PKU的课件里面对于本题的推导有些问题.UOJ官方题解提供了一个类似洲阁筛的做法.

代码抄的[ppl_blog]

[UOJ提交记录]

```
Int g(Int n,int m){
    m++; if(n<=2||prime[m]>n) return 0;
    Int s=0;
    while(prime[m]*prime[m]<=n&&m<=cnt){
        Int p=prime[m],c=1,pc=p;
        while(pc<=n){
            Int r=at(n/pc)-(m-1);
            if(r>0) s+=(g(n/pc,m)+p*(at(n/pc)-(m-1)));
            else s+=g(n/pc,m);
            pc=pc*p;c++;
        }
        m++;
    }
    return s;
}
```

这样就清晰了,$f(p^c\cdot q)=p\quad q\in prime,q\geq p,$

对于这种东西需要**乘一个大于等于p的质数**,所以要求$\lfloor\frac{n}{p^c}\rfloor\geq p\Rightarrow p^c\cdot p\leq n$

如果p不是次大质因子,那么递归去处理,按照$g_{n,m}$的定义,是正确的.

BZOJ 4916

$$\sum_{i=1}^{n}\mu(i^2)\quad\sum_{i=1}^{n}\varphi(i^2)$$

首先第一个是1...然后考虑第二个.

1. 这是个积性函数,质因子贡献独立.
2. 这个函数在质数幂出的取值可以快速求出,其中在质数出的取值是关于p的二次多项式.

套一下式子.没了.范围超小,跑得飞起.不过我们还是来考虑一下正解$O(n^{\frac{2}{3}})$的做法吧.如图.没了

$$let\ n=\prod_{i=1}^{m}p_i^{k_i}$$

$$\varphi(n^2)=\varphi(\prod_{i=1}^{m}p_i^{2k_i})=\prod_{i=1}^{m}\varphi(p_i^{2k_i})$$

$$\varphi(p^k)=p^k-p^{k-1}\quad \varphi(p^2k)=p^{2k}-p^{2k-1}=p^k\,\varphi(p^k)$$

$$\varphi(n^2)=n\varphi(n)$$

$$f(n)=n\varphi(n)\quad g(n)=n$$

$$(f*g)(n)=\sum_{d|n}d\varphi(d)\,(\frac{n}{d})=n\sum_{d|n}\varphi(d)=n^2$$

```cpp
#include <iostream>
#include <cstdio>
#include <cctype>
#include <cassert>
#include <algorithm>
const int N=200000+10;
typedef long long Int;
const Int mod=(Int)(1e9)+7LL;
Int qpow(Int a,Int p){
    Int r=1; a=(a%mod+mod)%mod;
    while(p){
        if(p&1) r=r*a%mod;
        a=a*a%mod;
        p>>=1;
    }
    return r;
}
inline Int inv(Int x){ return qpow(x,mod-2); }
int vis[N],prime[N],cnt,phi[N];
struct P{
    Int pre1,pre2;
    P(){ pre1=pre2=0; }
    Int get(){ return (pre2-pre1+mod)%mod; }
}pre[N];
void init(){
    vis[1]=1; phi[1]=1;
    for(int i=2;i<N;i++){
        pre[i]=pre[i-1];
        if(!vis[i]){
            pre[i].pre1=(pre[i].pre1+i)%mod;
            pre[i].pre2=(pre[i].pre2+1LL*i*i)%mod;
            prime[++cnt]=i;
            phi[i]=i-1;
        }
        for(int j=1;j<=cnt&&i*prime[j]<N;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0){
```

```
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
P A[N],B[N];
int n;
inline P& at(int m){ return (1.0*m*m<=n)?A[m]:B[n/m]; }
inline Int s1(Int m){
    static Int inv2=inv(2);
    m%=mod;
    return m*(m+1)%mod*inv2%mod;
}
inline Int s2(Int m){
    static Int inv6=inv(6);
    m%=mod;
    return (2*m+1)%mod*(m+1)%mod*m%mod*inv6%mod;
}
Int geth(int n){
    const P &x=at(n);
    return (x.pre2-x.pre1+mod)%mod;
}
Int g(int n,int m){
    m++;
    if(prime[m]>n) return 0;
    Int ret=0;
    while(1.0*prime[m]*prime[m]<=n){
        Int p=prime[m],pc=p,fpc=p*(p-1)%mod;
        while(pc<=n){
            ret=(ret+fpc*(1+g(n/pc,m))%mod)%mod;
            pc=pc*p;
            fpc=fpc*p%mod*p%mod;
        }
        m++;
    }
    Int tmp=(geth(n)-geth(prime[m-1])+mod)%mod;
    return (ret+tmp)%mod;
}

Int solve(int _n){
    ::n=_n;
    int l=1,r=0,q=0;
    while(l<=n){
        q=n/l; r=n/q;
        P &to=at(q);
        to.pre1=(s1(q)-1+mod)%mod;
        to.pre2=(s2(q)-1+mod)%mod;
        l=r+1;
    }
    for(int i=1;i<=cnt&&
            1.0*prime[i]*prime[i]<=n;i++){
```

```
        Int p=prime[i];
        l=1;r=0;q=0;
        while(l<=n){
            q=n/l; r=n/q;
            if(1.0*p*p>q) break;
            P &to=at(q),&from=at(q/p);
            to.pre1-=p*(from.pre1-pre[p-1].pre1)%mod;
            to.pre1=(to.pre1%mod+mod)%mod;
            to.pre2-=p*p%mod*(from.pre2-pre[p-1].pre2)%mod;
            to.pre2=(to.pre2%mod+mod)%mod;
            l=r+1;
        }
    }
    return 1+g(n,0);
}

int main(){
    init();
    using namespace std;
    int n; cin>>n;
    cout<<1<<endl;
    cout<<solve(n)<<endl;
    return 0;
}

```