

筛法

埃氏筛

列出所有数字，从小到大枚举，将枚举数的所有倍数筛掉。复杂度 $O(n\log\log n)$ 证明见[这里](#)

```
void sieve(int n){
    int i,j;
    isnp[0]=isnp[1]=1;
    for(i=2;i<=n;i++){
        if(isnp[i])continue;
        pri[cnt++]=i;
        for(j=i;j<=n;j+=i)isnp[j]=1;
    }
}
```

欧拉筛（线性筛）

埃氏筛会将一个合数被其所有质因数都筛一遍，很浪费时间。

考虑优化，让每个合数都只被最大的非本身的因数（和最小质因数共同）筛到一遍。

故首先枚举所有数 i 再枚举所有 i 的素倍数 $t=pri_j \times i$ (i 与 $pri[j]$ 共同) 将 t 筛掉，且当 $pri_j \mid i$ 时退出枚举。此举的正确性在于：

- i 的最小质因数为 $pri[j]$
- $\forall k > j, i \times pri[k]$ 会被比 i 更大的 $\frac{i}{pri[j]} \times pri[k]$ 与 $pri[j]$ 共同筛掉。

因此，欧拉筛的每个数都只被筛了一次，复杂度 $O(n)$

模板题

```
void sieve(int n){
    int i,j;
    isnp[0]=isnp[1]=1;
    for(i=2;i<=n;i++){
        if(!isnp[i])pri[cnt++]=i;
        for(j=0;j<cnt;j++){
            if(pri[j]*i>n)break;
            isnp[pri[j]*i]=1;
            if(i%pri[j]==0)break;
        }
    }
}
```

```
}
```

除了筛素数，欧拉筛还可以线性地筛一些[积性函数](#)

欧拉函数

欧拉函数 $\varphi(n)$ 表示小于等于 n 且 $\gcd(i,n)=1$ 的 i 个数。

欧拉函数是积性的，也就是对任意 n,m 满足 $\gcd(n,m)=1$ 有 $\varphi(n \times m) = \varphi(n) \times \varphi(m)$ [有一个不错的证法](#)

处理边界情况：

- 当 $n=1$ 的时候，规定 $\varphi(1)=1$
- 当 $n=p$ 的时候 $\varphi(n)=p-1$
- 当 $n=p^k$ 的时候 $\varphi(n)=p^{k-1}(p-1)$

因为欧拉函数是积性的，如果将 n 质因数分解为 $n = \prod_i p_i^{k_i}$ 可以得到：

$$\varphi(n) = \prod_i p_i^{k_i-1} (p_i - 1) = n \prod_i \frac{p_i - 1}{p_i}$$

```
void sieve(int n){
    int i,j;
    isnp[0]=isnp[1]=1;
    phi[1]=1;
    for(i=2;i<=n;i++){
        if(!isnp[i])pri[cnt++]=i,phi[i]=i-1;
        for(j=0;j<cnt;j++){
            if(pri[j]*i>n)break;
            isnp[pri[j]*i]=1;
            if(i%pri[j]==0){
                phi[pri[j]*i]=phi[i]*pri[j];
                break;
            }else{
                phi[pri[j]*i]=phi[i]*phi[pri[j]];
            }
        }
    }
}
```

莫比乌斯函数

[这里](#)讲过了，不再赘述。

杜教筛

杜教筛想要解决的问题是，对于数论函数 f 要在小于线性的复杂度求出前缀和 $S_f(n) = \sum_{i=1}^n f(i)$

可以应用杜教筛的前提是，存在一个易求前缀和的数论函数 g 使得狄利克雷卷积 $f \ast g$ 易求前缀和。当两个函数都可以 $O(1)$ 地求出在某点的前缀和时，通过预处理一定数量的前缀和，求出 f 在某处的前缀和复杂度是可以达到 $O(n^{\frac{2}{3}})$ 的。

具体推导过程如下：（设 f, g, h 的前缀和函数分别为 s_f, s_g, s_h ）

$$\begin{aligned} s_h(n) &= \sum_{i=1}^n h(i) = \sum_{d \mid i} f\left(\frac{i}{d}\right) g(d) \\ &= \sum_{d=1}^n \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} g(d) f(t) \\ &= \sum_{d=1}^n g(d) s_f\left(\lfloor \frac{n}{d} \rfloor\right) \\ &= g(1) s_f(n) + \sum_{d=2}^n g(d) s_f\left(\lfloor \frac{n}{d} \rfloor\right) \quad \parallel s_f(n) = \frac{s_h(n) - \sum_{d=2}^n g(d) s_f\left(\lfloor \frac{n}{d} \rfloor\right)}{g(1)} \end{aligned}$$

等式右边数论分块处理，递归计算 s_f 即可。

复杂度证明

设 $A = \{1, 2, 3, \dots, \lfloor \sqrt{n} \rfloor\}$, $B = \{\lfloor \frac{n}{2} \rfloor, \dots, \lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \rfloor\}$ 设 $U(n) = A \cup B$ 易看出 $|U(n)|$ 是 $O(\sqrt{n})$ 级别的。同时，对于任意 $m \in U(n)$ 有 $U(m) \subseteq U(n)$ 证明：设 $m = \lfloor \frac{n}{a} \rfloor$ 则任意 $\lfloor \frac{m}{b} \rfloor = \lfloor \frac{n}{ab} \rfloor \in U(n)$

设计算出 $s_f(n)$ 复杂度为 $T(n)$ 则根据上述结论，为计算出 $s_f(n)$ 只需要在记忆化过程中总共计算出 $s_f(i), i \in U(n)$ 即可，故考虑枚举次数，有等式：

$$\begin{aligned} T(n) &= O\left(\sum_{i=1}^{\lfloor \sqrt{n} \rfloor} (\sqrt{i} + \sqrt{\frac{n}{i}})\right) \\ &= O\left(\int_1^{\lfloor \sqrt{n} \rfloor} (\sqrt{x} + \sqrt{\frac{n}{x}}) dx\right) = O\left(x^{\frac{3}{2}} + \sqrt{n} \ln x \mid_1^{\lfloor \sqrt{n} \rfloor}\right) = O\left(x^{\frac{3}{2}}\right) \end{aligned}$$

设线性预处理了前 $b > \sqrt{n}$ 项，则复杂度为：

$$\begin{aligned} T(n) &= O\left(\sum_{i=1}^{\lfloor \sqrt{\frac{nb}{b}} \rfloor} (\sqrt{\frac{n}{i}} + b)\right) \\ &= O\left(\int_1^{\lfloor \sqrt{\frac{nb}{b}} \rfloor} (\sqrt{\frac{n}{x}} + b) dx\right) = O\left(\frac{n}{\sqrt{b}} + b\right) \end{aligned}$$

取 $b = n^{\frac{2}{3}}$ 取得最优复杂度 $O(n^{\frac{2}{3}})$

实例

模板题

[Luogu P4213 模板】杜教筛Sum](#) [51nod 124451nod 1239](#)

三个类似的题，计算 $[1, 2^{31}-1]$ 范围内 φ, μ 的前缀和。很显然有 $\varphi \ast 1 = \text{id}, \mu \ast 1 = \epsilon$ 直接筛即可。

GCD 二维前缀和

51nod 1237

计算 $\sum_{i=1}^n \sum_{j=1}^n \gcd(i,j)$ □

```
$$ \begin{aligned} \sum_{i=1}^n \sum_{j=1}^n \gcd(i,j) &= \sum_{d=1}^n d \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} [(i,j)=1] \\ &= \sum_{d=1}^n d \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{k \mid i, k \mid j} \mu(k) \\ &= \sum_{d=1}^n d \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{T=1}^{\lfloor \frac{n}{dk} \rfloor} \sum_{d \mid T} \mu\left(\frac{Td}{d}\right) \sum_{d \mid T} \mu\left(\frac{Td}{d}\right) \sum_{d \mid T} \mu\left(\frac{Td}{d}\right) \\ &= \sum_{T=1}^n \sum_{d \mid T} \mu\left(\frac{Td}{d}\right) \sum_{d \mid T} \mu\left(\frac{Td}{d}\right) \sum_{d \mid T} \mu\left(\frac{Td}{d}\right) \end{aligned} $$
```

对 T 数论分块即可，其中需要快速计算欧拉函数前缀和。

LCM 二维前缀和

51nod 1238

计算 $\sum_{i=1}^n \sum_{j=1}^n \text{lcm}(i,j)$ □

```
$$ \begin{aligned} \sum_{i=1}^n \sum_{j=1}^n \text{lcm}(i,j) &= \sum_{d=1}^n d \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} [\gcd(i,j)=1] \\ &= \sum_{d=1}^n d \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{k \mid i, k \mid j} \mu(k) \\ &= \sum_{d=1}^n d \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \\ &= \sum_{d=1}^n d \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \mu(k) \sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \sum_{j=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \\ &= \sum_{d=1}^n d \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \mu(k)^2 \sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \sum_{j=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \\ &= \sum_{d=1}^n d \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \mu(k)^2 \sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \sum_{j=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \\ &= \sum_{d=1}^n d \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \mu(k)^2 \sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \sum_{j=1}^{\lfloor \frac{n}{dk} \rfloor} \mu(k) \end{aligned} $$
```

因此我们只需要快速求出 $f(i) = i^2 \varphi(i)$ 的前缀和，然后对 d 数论分块求解。

找到 $g(i) = \text{id}(i)^2$ □ 此时有 $(\text{last } g)(n) = \sum_{d \mid n} d^2 \varphi(d) \sum_{d \mid n} d^2 \varphi(d) = n^3$ □ 利用公式 $\sum_{i=1}^n i^3 = \frac{n^2(n+1)^2}{4}$ □ 本题得解。

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
#include<cstdliblib>
```

```

#include<set>
#include<vector>
typedef long long ll;
using namespace std;
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define mod 1000000007
#define N 2333333
//#define N 1
char isnp[N+10];
int cnt,pri[N+10];
ll phi[N+10],f[N+10],sf[N+10];
void sieve(int n){
    int i,j;
    isnp[0]=isnp[1]=1;
    phi[1]=1;
    for(i=2;i<=n;i++){
        if(!isnp[i])pri[cnt++]=i,phi[i]=i-1;
        for(j=0;j<cnt;j++){
            if(i*pri[j]>n)break;
            isnp[i*pri[j]]=1;
            if(i%pri[j]==0){
                phi[i*pri[j]]=phi[i]*pri[j];
                break;
            }else{
                phi[i*pri[j]]=phi[i]*(pri[j]-1);
            }
        }
    }
    for(i=1;i<=n;i++)f[i]=1LL*i*i%mod*phi[i]%mod;
    for(i=1;i<=n;i++)sf[i]=(sf[i-1]+f[i])%mod;
}
map<ll,ll>m;
ll inv2,inv3;
ll qp(ll a,ll b){
    ll res=1%mod;
    for(;b;b>>=1,a=a*a%mod)if(b&1)res=res*a%mod;
    return res;
}
ll s1(ll n){
    return n%mod*(n%mod+1)%mod*inv2%mod;
}
ll s2(ll n){
    return s1(n)*(2*n%mod+1)%mod*inv3%mod;
}
ll s3(ll n){
    return s1(n)*s1(n)%mod;
}

```

```

ll get(ll n){
    if(n<N)return sf[n];
    if(m.count(n))return m[n];
    ll res=s3(n),i,r;
    for(i=2;i<=n;i=r+1){
        r=n/(n/i);
        (res-=get(n/i)*(s2(r)-s2(i-1)))%=mod;
    }
    return m[n]=res;
}
ll s1(ll l,ll r){
    l%=mod;r%=mod;
    return (r+l)*(r-l+1)%mod*inv2%mod;
}
int main(){
    ll i,r,n,ans=0;
    inv2=qp(2,mod-2);
    inv3=qp(3,mod-2);
    sieve(N);
    scanf("%lld",&n);
    for(i=1;i<=n;i=r+1){
        r=n/(n/i);
        (ans+=s1(i,r)*get(n/i)%mod)%=mod;
    }
    printf("%lld",(ans+mod)%mod);
    return 0;
}

```

平均最小公倍数之和

51nod 1227

求 $\sum_{i=a}^b \frac{1}{\sum_{j=1}^i \text{lcm}(i,j)}$ \square

考虑求前缀和函数 s_n 进行差分，答案为 $s_b - s_{a-1}$ \square 以下推导比较详细（冗杂），是为方便接触较少的同学一步步推导（后续内容会适当省略某些步骤）：

$$\begin{aligned}
 s_n &= \sum_{i=1}^n \frac{1}{\sum_{j=1}^i \text{lcm}(i,j)} \\
 &= \sum_{i=1}^n \frac{1}{\sum_{j=1}^i \frac{ij}{\gcd(i,j)}} = \sum_{i=1}^n \sum_{d \mid i} \frac{1}{d \sum_{j=1}^i \frac{j}{\gcd(i,j)=d}} = \sum_{i=1}^n \sum_{d \mid i} \frac{1}{d \sum_{j=1}^{\frac{id}{d}} j [\gcd(\frac{id}{d},j)=1]} = \sum_{i=1}^n \sum_{d \mid i} F(\frac{id}{d})
 \end{aligned}$$

其中 $F(n) = \sum_{j=1}^n \frac{1}{j [\gcd(n,j)=1]}$ $= \sum_{j=1}^n \sum_{k \mid n, k \mid j} \frac{1}{\mu(k)}$ $= \sum_{k \mid n} \frac{1}{\mu(k)} \sum_{j=1}^{\frac{nk}{k}} \frac{1}{k} = \sum_{k \mid n} \frac{1}{\mu(k)} \frac{1}{k} \frac{1}{2} \frac{nk+1}{k} = \frac{n}{2} \sum_{k \mid n} \frac{1}{\mu(k)k} = \frac{n}{2} (\varphi(n) + \epsilon(n))$ 于是 $s_n = \sum_{i=1}^n \sum_{d \mid i} \frac{1}{d} \frac{1}{2} (\varphi(d) + \epsilon(d)) = \sum_{d=1}^n \sum_{i=1}^{\lfloor \frac{nd}{d} \rfloor} \frac{1}{d} \frac{1}{2} (\varphi(d) + \epsilon(d)) = \frac{1}{2} \sum_{d=1}^n \lfloor \frac{nd}{d} \rfloor \frac{1}{d} (\varphi(d) + \epsilon(d))$

故问题转化为快速求 $f = \sum_{d|n} \varphi(d)$ 的前缀和，令 $g(n) = \sum_{d|n} \varphi(d) \frac{n}{d}$ 则 $f(n) = \sum_{d|n} \varphi(d) \frac{n}{d} = n^2 \sum_{d|n} \frac{1}{d^2}$

```

#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define mod 1000000007
#define N 1000000
char isnp[N+10];
int pri[N+10],cnt;
ll phi[N+10],sp[N+10];
void sieve(int n){
    int i,j;
    isnp[0]=isnp[1]=1;
    phi[1]=1;
    for(i=2;i<=n;i++){
        if(!isnp[i])pri[cnt++]=i,phi[i]=i-1;
        for(j=0;j<cnt;j++){
            if(pri[j]*i>n)break;
            isnp[i*pri[j]]=1;
            if(i%pri[j]==0){
                phi[i*pri[j]]=phi[i]*pri[j];
                break;
            }else{
                phi[i*pri[j]]=phi[i]*(pri[j]-1);
            }
        }
    }
    for(i=1;i<=n;i++)sp[i]=(sp[i-1]+1LL*i*phi[i])%mod;
}
map<int,ll>m;
ll inv2,inv6;
ll qp(ll a,ll b){
    ll res=1;
    for(;b;b>>=1,a=a*a%mod)if(b&1)res=res*a%mod;
    return res;
}

```

```
}
ll s1(ll l,ll r){
    return (l+r)*(r-l+1)%mod*inv2%mod;
}
ll s2(ll n){
    return n*(n+1)%mod*(2*n+1)%mod*inv6%mod;
}
ll getSF(int n){
    if(n<=N)return sp[n];
    if(m.count(n))return m[n];
    int i,r;
    ll res=s2(n);
    for(i=2;i<=n;i=r+1){
        r=n/(n/i);
        (res-=getSF(n/i)*s1(i,r)%mod)%=mod;
    }
    return m[n]=res;
}
ll getS(int n){
    ll res=n;
    int i,r;
    for(i=1;i<=n;i=r+1){
        r=n/(n/i);
        (res+=n/i*(get(r)-get(i-1))%mod)%=mod;
    }
    return res*inv2%mod;
}
int main(){
    int a,b;
    inv2=qp(2,mod-2);
    inv6=qp(6,mod-2);
    sieve(N);
    scanf("%d%d",&a,&b);
    printf("%d",(getS(b)-getS(a-1)+2*mod)%mod);
    return 0;
}
```

约数之和

51nod 1220

求 $\sum_{i=1}^n \sum_{j=1}^n \sigma(i \times j)$ □

关于约数和 σ 和约数个数 d 分别有结论：

$$\begin{aligned} d(i \times j) &= \sum_{x \mid i} \sum_{y \mid j} [(x,y)=1] \\ \sigma(i \times j) &= \sum_{x \mid i} \sum_{y \mid j} \frac{iy}{x} [(x,y)=1] \end{aligned}$$

关于证明，[这里](#)给出了一种对于第一个式子偏数学的证明，这里给出一个偏感性的证明（本质相同）。

我们需要考虑 (x,y) 代表了 ij 的哪个因数，并讨论这是否是一个一一映射。

因为 $(x,y)=1$ 故任意质因子 p 不可能 $p \mid x, p \mid y$ 设 $p^a \mid i, p^b \mid j$

如果 $p \mid x$ 且 $p^{k_1} \mid x$ 那么对应因数含有 p^{a-k_1} 因子（只是为方便证明第二式起见，可以对应含有 p^{k_1} 因子）；否则设 $p^{k_2} \mid y$ 那么对应因数含有 p^{a+k_2} 因子。很明显，这样构造出来的映射是一一映射。

$$\begin{aligned}
& \sum_{i=1}^n \sum_{j=1}^n \sigma(ij) = \sum_{i=1}^n \sum_{j=1}^n \sum_{x \mid i} \sum_{y \mid j} \frac{iy}{x} [(x,y)=1] \\
& = \sum_{i=1}^n \sum_{j=1}^n \sum_{x \mid i, d \mid x} \sum_{y \mid j, d \mid y} \frac{iy}{x} \\
& = \sum_{d=1}^n \mu(d) \sum_{i=1}^n \sum_{j=1}^n \sum_{x=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{y=1}^{\lfloor \frac{n}{d} \rfloor} \frac{yd}{xd} \sum_{i=1}^{\lfloor \frac{n}{xd} \rfloor} i x \sum_{j=1}^{\lfloor \frac{n}{yd} \rfloor} j y \\
& = \sum_{d=1}^n d \mu(d) \sum_{x=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{y=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{xd} \rfloor} i \sum_{j=1}^{\lfloor \frac{n}{yd} \rfloor} j \\
& = \sum_{d=1}^n d \mu(d) \left(\sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} S_1 \left(\lfloor \frac{n}{id} \rfloor \right) \right)^2
\end{aligned}$$

其中 $S_1(n) = \frac{1}{2}(1+n)n$

于是可以对 d 数论分块 $\sum_{i=1}^n \frac{1}{2} \lfloor \frac{n}{d} \rfloor (\lfloor \frac{n}{d} \rfloor + 1)$ 线筛预处理一部分，剩余 $O(\sqrt{n})$ 直接算（或者不处理直接暴力 $O(n^{\frac{3}{4}})$ 另外需要快速计算出 $f = \text{id} \cdot \mu$ 的前缀和，令 $g = \text{id}$ 显有 $\text{last } g = \epsilon$

从上述步骤的第六步到第七步的转换，验证了等式 $\sum_{i=1}^n \frac{1}{2} \lfloor \frac{n}{d} \rfloor (\lfloor \frac{n}{d} \rfloor + 1) = \sum_{i=1}^n i \lfloor \frac{n}{d} \rfloor$ 而后者可以视作在 i 处统计了 i 的所有因数之和（对于每一个因数 i 在他共 $\lfloor \frac{n}{d} \rfloor$ 个整数倍的位置都被统计一次），即为 $\sum_{i=1}^n \sigma(i)$ 实现的时候存一个 div_i 表示 i 的最小质因数做贡献的一项 $(1 + p_m + p_m^2 + \dots + p_m^{k_m})$ 的值即可。如果不转化为 σ 的前缀和，也可以直接打表验证积性函数后记录 p^k 下的取值进行线筛。

```

#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;

```

```
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define N 3000000
#define mod 1000000007
char isnp[N+10];
int pri[N+10],cnt,mu[N+10],sm[N+10];
ll sigma[N+10],dv[N+10],sr[N+10];
void sieve(int n){
    int i,j;
    isnp[0]=isnp[1]=1;
    mu[1]=1;sigma[1]=1;
    for(i=2;i<=n;i++){
        if(!isnp[i])pri[cnt++]=i,mu[i]=-1,sigma[i]=dv[i]=1+i;
        for(j=0;j<cnt;j++){
            if(pri[j]*i>n)break;
            isnp[i*pri[j]]=1;
            if(i%pri[j]==0){
                mu[i*pri[j]]=0;
                dv[i*pri[j]]=dv[i]*pri[j]+1;
                sigma[i*pri[j]]=sigma[i]/dv[i]*dv[i*pri[j]];
                break;
            }else{
                mu[i*pri[j]]=-mu[i];
                sigma[i*pri[j]]=sigma[i]*sigma[pri[j]];
                dv[i*pri[j]]=pri[j]+1;
            }
        }
    }
    for(i=1;i<=n;i++){
        sm[i]=(sm[i-1]+i*mu[i])%mod;
        sr[i]=(sr[i-1]+sigma[i])%mod;
    }
}
ll inv2;
ll qp(ll a,ll p){
    ll ans=1;
    for(;p;p>>=1,a=a*a%mod)if(p&1)ans=ans*a%mod;
    return ans;
}
ll s1(int l,int r){
    return (l+r)*(r-l+1LL)%mod*inv2%mod;
}
map<int,ll>m;
ll getRight(int n){
    if(n<=N)return sr[n];
    int i,r;
    ll res=0;
```

```

    for(i=1;i<=n;i=r+1){
        r=n/(n/i);
        res+=n/i*(n/i+1LL)%mod*(r-i+1)%mod*inv2%mod;
    }
    return res%mod;
}
ll getS(int n){
    if(n<=N) return sm[n];
    if(m.count(n)) return m[n];
    int i,r;
    ll res=1;
    for(i=2;i<=n;i=r+1){
        r=n/(n/i);
        (res-=getS(n/i)*s1(i,r)%mod)%=mod;
    }
    return m[n]=res;
}
ll get(int n){
    ll tmp,res=0;
    int i,r;
    for(i=1;i<=n;i=r+1){
        r=n/(n/i);
        tmp=getRight(n/i);
        (res+=tmp*tmp%mod*(getS(r)-getS(i-1))%mod)%=mod;
    }
    return (res+mod)%mod;
}
int main(){
    int n;
    inv2=qp(2,mod-2);
    sieve(N);
    scanf("%d",&n);
    printf("%lld",get(n));
    return 0;
}

```

CCPC 2019 网络赛 E - huntian oy

题意：求

$$\sum_{i=1}^n \sum_{j=1}^i \gcd(i^a - i^b, j^a - j^b) [\gcd(i, j) = 1]$$

其中 a, b 是给定的数，且 a, b 互质。

题解□

首先有结论（我没找到证明）：

$$\gcd(i^a - j^a, i^b - j^b) = i^{\gcd(a, b)} - j^{\gcd(a, b)}$$

推式子：

$$\sum_{i=1}^n \sum_{j=1}^i (i-j) \sum_{g \mid i \wedge g \mid j} \mu(g) = \sum_{g=1}^n \mu(g) \sum_{i=1}^{\lfloor n/g \rfloor} \sum_{j=1}^i (i-j)g = \sum_{g=1}^n \mu(g) \times g \times F \left(\left\lfloor \frac{n}{g} \right\rfloor \right)$$

其中 $F(n) = \sum_{i=1}^n \sum_{j=1}^i (i-j)$ 我们现在需要 $(\mu \cdot \text{id})(n)$ 在分块点处的前缀和，才能分块计算整个式子。注意到 $(\mu \cdot \text{id}) \ast \text{id}(n) = \sum_{d \mid n} \mu(d) \cdot d \cdot \text{dfrac}{nd} = n \cdot (\mu \ast 1)(n) = n \cdot \text{varepsilon}(n)$ 故可以让函数卷上 id 进行杜教筛：

$$\sum_{i=1}^n i \cdot \text{varepsilon}(n) = \sum_{i=1}^n i \times S \left(\left\lfloor \frac{n}{i} \right\rfloor \right)$$

其中 S 为 $(\mu \cdot \text{id})$ 的前缀和。

min_25 筛

min_25 筛可以用来解决具有下列性质的积性函数 f 在单点处的前缀和：

1. 存在完全积性函数 f' 使得质数 p 处有 $f'(p) = f(p)$
2. $f'(p)$ 的前缀和很好计算。
3. $f'(p^k)$ 的值很好算（是多项式或可以很快求出）；

为了方便，我们约定：

- 若没有特别说明，下文中的所有 p 都取质数；
- pri_j 表示第 j 小的质数；
- $\text{minp}(n)$ 表示 n 的最小质因子。

计算质数贡献

我们先考虑解决求所有分块点处质数的贡献，即 $G(m) = \sum_{i=1}^m [i \text{ is prime}] f(i)$ 其中 m 是 n 的数论分块点。

设状态 $g(j, m)$ 表示 $\sum_{i=1}^m [i \text{ is prime or } \text{minp}(i) > \text{pri}_j] f(i)$ 则我们要求的就是 $G(m) = g(\sum_{j \wedge \text{pri}_j \leq m} 1, m)$ 了。这个状态可以理解为 Eratosthenes 筛法中用 pri_j 筛掉它的倍数后，剩余没被筛掉部分的和。

显然，边界条件是 $g(0, m) = \sum_{i=2}^m f(i)$ 请注意一定要去掉 $f(1)$ 因为它不会被筛掉，我们只要最后加上即可。

考虑转移。若 $\text{pri}_j > \sqrt{m}$ 此时筛不掉任何数，故 $g(j, m) = g(j-1, m)$ 故考虑 $\text{pri}_j \leq \sqrt{m}$ 的情况。此时 pri_j 能筛掉的合数都能表示为 pri_j 乘一个不超过 $\lfloor \frac{m}{\text{pri}_j} \rfloor$ 的、最小质因子大于等于 pri_j 的数，即

$$g(j, m) = g(j-1, m) - f(\text{pri}_j) \times [g(j-1, \lfloor \frac{m}{\text{pri}_j} \rfloor) - \sum_{i=1}^{j-1} f(\text{pri}_i)]$$

因为 f 是完全积性函数，所以即使分解的东西不互质也可以直接乘在一起。另外，式子最后的部分用于减去质数的贡献（参考 g 的定义）。

具体实现时，可以用滚动数组存放 g 滚掉它的第一维 j 可以证明这一部分的时间复杂度是 $O\left(\frac{n^{3/4}}{\log n}\right)$ 的。

计算合数贡献

设 $S(n, j) = \sum_{i=1}^m [\mathrm{minp}(i) \geq \mathrm{pri}_j] f(i)$ 注意中间对 $\mathrm{minp}(i)$ 的要求和 G 有所不同。根据定义，我们最后要求的就是 $S(n, 1) + f(1)$

对于 $S(n, j)$ 质数的贡献为 $G(n) - \sum_{i=1}^{j-1} f(\mathrm{pri}_i)$ 而合数的贡献需要枚举其最小质因子及其次数，分为两个互质的部分。转移如下：

$$S(n, j) = G(n) - \sum_{i=1}^{j-1} f(\mathrm{pri}_i) + \sum_{i=j}^{\mathrm{pri}_i \leq n} \sum_{e=1}^{\mathrm{pri}_i^{e+1} \leq n} [f(\mathrm{pri}_i^e) S(\lfloor \frac{n}{\mathrm{pri}_i^e} \rfloor, i+1) + f(\mathrm{pri}_i^{e+1})]$$

这部分不需要记忆化，直接递归的时间复杂度是 $O(n^{1-\epsilon})$ 的，而且跑 $n = 10^{10}$ 的数据也很飞快。

实例

Luogu 5325

题意：积性函数 $f(n)$ 的质数幂次取值 $f(p^k) = p^k(p^k - 1)$ 求前缀和。

题解： $f(p^k) = (p^2)^k - p^k$ 拆成两部分计算质数处贡献，最后合并即可。

```
#include <bits/stdc++.h>
using namespace std;

#define fi first
#define se second
#define lch (o << 1)
#define rch (o << 1 | 1)

typedef double db;
typedef long long ll;
typedef unsigned int ui;
typedef pair<int, int> pint;

const int N = 2e5 + 5;
const int MOD = 1e9 + 7;
const int INV2 = 500000004;
const int INV6 = 166666668;
const int INF = 0x3f3f3f3f;
const ll INF_LL = 0x3f3f3f3f3f3f3f3f;
```

```
// 注意两倍空间
int idx[N];
ll val[N];
ll _n, sqrtN;
int nIdx;

int GetId(ll x) {
    if (x <= sqrtN)
        return idx[x];
    else
        return idx[sqrtN + _n / x];
}

void InitId(ll n) {
    // x = n / i 是该分块中能取到最大 / 最右侧的值
    for (ll i = 1; i <= n; i++) {
        ll x = n / i;
        val[++nIdx] = x;
        if (x <= sqrtN)
            idx[x] = nIdx;
        else
            idx[sqrtN + n / x] = nIdx;
        i = n / (n / i);
    }
}

int pri[N];
ll s1[N], s2[N];
bool notPri[N];

void InitPri() {
    for (int i = 2; i < N; i++) {
        if (!notPri[i])
            pri[++pri[0]] = i;
        for (int j = 1; j <= pri[0]; j++) {
            int p = pri[j];
            if (i * p >= N) break;
            notPri[i * p] = 1;
            if (i % p == 0) break;
        }
    }
    for (int i = 1; i <= pri[0]; i++) {
        s1[i] = (s1[i - 1] + 1LL * pri[i] * pri[i]) % MOD;
        s2[i] = (s2[i - 1] + pri[i]) % MOD;
    }
}

ll f1[N], f2[N];
```

```

// Sieve1(n) = sum [i = 1 -> n] [i is isprime] * f(i)
// 传入的 n 是原始值
void Sieve1(ll n) {
    // 预处理初值
    // 需要注意这里一般不计算 f(1)
    for (int i = 1; i <= nIdx; i++) {
        ll x = val[i] % MOD;
        f1[i] = x * (x + 1) % MOD * (2 * x + 1) % MOD * INV6 % MOD - 1;
        f2[i] = (1 + x) * x % MOD * INV2 % MOD - 1;
    }
    for (int j = 1; j <= pri[0]; j++) {
        ll p = pri[j];
        if (p * p > n) break;
        for (int i = 1; i <= nIdx; i++) {
            ll x = val[i];
            if (p * p > x) break;
            // f(m) = f(idx[m / pri[j]]) - sum[j - 1];
            f1[i] -= (p * p) % MOD * (f1[GetId(x / p)] - s1[j - 1]);
            f2[i] -= p * (f2[GetId(x / p)] - s2[j - 1]);
            f1[i] %= MOD;
            f2[i] %= MOD;
        }
    }
}

// f(p ^ e)
ll F(ll p, int e) {
    ll t = 1;
    while (e--) t *= p;
    t %= MOD;
    return t * (t - 1) % MOD;
}

// Sieve2(n, j) = sum [i = 1 -> n] [minp(i) >= pri[j]] * f(i)
// 传入的 n 是原始值
ll Sieve2(ll n, int j) {
    if (n <= 1 || pri[j] > n) return 0;
    int id = GetId(n);
    // ret = Sieve1(n) - sum [i = 1 -> j - 1] f(i) + ...
    ll ret = (f1[id] - s1[j - 1]) - (f2[id] - s2[j - 1]);

    // ret = ... + 合数部分
    for (int i = j; i <= pri[0]; i++) {
        ll p = pri[i], t = p; // t = p ^ e
        if (p * p > n) break;
        for (int e = 1; p * t <= n; t *= p, e++) {
            ll tmp = F(p, e) * Sieve2(n / t, i + 1) + F(p, e + 1);
            ret = (ret + tmp) % MOD;
        }
    }
}

```

```
return ret;
}

int main() {
    ios::sync_with_stdio(0);

    ll n;
    cin >> n;
    sqrtN = floor(sqrt(n));
    _n = n;

    InitId(n);
    InitPri();
    Sieve1(n);
    ll ans = (Sieve2(n, 1) + 1) % MOD;
    cout << (ans + MOD) % MOD;

    return 0;
}
```

2019 ICPC 徐州网络赛 H - function

题意：令 n 的唯一分解为 $n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$ 定义 $f(n) = k_1 + k_2 + \cdots + k_m$ 求

$$\sum_{i=1}^n f(i)$$

题解：注意到 $f(n!) = \sum_{i=1}^n f(i)$ 故

$$\sum_{i=1}^n f(i!) = \sum_{i=1}^n \sum_{j=1}^i f(j) = \sum_{i=1}^n (n-i+1) f(i) = (n+1) \sum_{i=1}^n f(i) - \sum_{i=1}^n i \cdot f(i)$$

因此实际要求 $\sum_{i=1}^n f(i)$ 和 $\sum_{i=1}^n i \cdot f(i)$

考虑枚举每个质数 p_i 的贡献。若 $p_i^2 \leq n$ 则第一个式子相当于统计 p_i 在 $[1, n]$ 作为约数的出现次数，这个经典问题可以用递归在 $O(\log n)$ 内解决（第二个式子类似）。

现在考虑 $p_i^2 > n$ 的部分，此时 k_i 最多取到 1。于是我们考虑我们要求的两个式子中，这些部分做出的贡献。

第一个式子的贡献等价于统计 $[1, n]$ 中 p_i 的倍数个数：

$$\sum_{i^2 > n} \left\lfloor \frac{n}{i} \right\rfloor [i \text{ is prime}]$$

第二个式子的贡献等价于统计 $[1, n]$ 中 p_i 的倍数之和：

$$\sum_{i^2 > n} \frac{(1 + \lfloor n/i \rfloor) \lfloor n/i \rfloor}{2} \cdot i \cdot [i \text{ is prime}]$$

相当于我们要求分块点处，质数个数的前缀和与质数的前缀和。这两个的求解是 min_25 的经典应用，故

套用 min_25 筛得到分块点处的值即可求解。

From:

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

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:i_dont_know_png:potassium:sieve&rev=1591504721 

Last update: **2020/06/07 12:38**