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 $(m,n)=1$ ，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：

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

```
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 $O(n^{\tfrac23})$ 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

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

等式右边数论分块处理，递归计算 s_f 即可。

复杂度证明

设 $A=\{1,2,3,\ldots,\lfloor\sqrt n\rfloor\},B=\{\lfloor\frac n2\rfloor,\ldots,\lfloor\frac{n}{\lfloor\sqrt n\rfloor}\rfloor\}$ 设 $U(n)=A\bigcup B$ 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 $m=\lfloor\frac na\rfloor$ 则任意 $\lfloor\frac mb\rfloor=\lfloor\frac m{ab}\rfloor \in U(n)$

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：

$$ \begin{aligned} T(n)&=O(\sum_{i=1}^{\lfloor\sqrt n\rfloor}(\sqrt i+\sqrt\frac{n}{i}))\\ &=O(\int_{1}^{\lfloor\sqrt n\rfloor}(\sqrt x+\sqrt\frac{n}{x}) \text{ d}x)\\ &=O((x^{\frac 32}+\sqrt n\sqrt x) \mid_{1}^{\lfloor\sqrt n\rfloor})\\ &=O(x^{\frac34}) \end{aligned} $$

设线性预处理了前 $b>\sqrt n$ 项，则复杂度为：

$$ \begin{aligned} T(n)&=O(\sum_{i=1}^{\lfloor\sqrt{\frac nb}\rfloor}\sqrt\frac{n}{i}+b)\\ &=O(\int_{1}^{\lfloor\sqrt{\frac nb}\rfloor}\sqrt\frac{n}{x}\text{ d}x+b)\\ &=O(\frac n{\sqrt b}+b) \end{aligned} $$

取 $b=n^{\frac 23}$ 取得最优复杂度 $O(n^{\frac 23})$

实例

模板题

Luogu P4213 模板】杜教筛Sum51nod 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 $\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)$

$$ \begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i,j)&=\sum_{d=1}^{n}d\sum_{i=1}^{\lfloor\frac nd\rfloor}\sum_{j=1}^{\lfloor\frac nd\rfloor}[(i,j)=1]\\ &=\sum_{d=1}^{n}d\sum_{i=1}^{\lfloor\frac nd\rfloor}\sum_{j=1}^{\lfloor\frac nd\rfloor}\sum_{k\mid i,k\mid j}\mu(k)\\ &=\sum_{d=1}^{n}d\sum_{k=1}^{\lfloor\frac nd\rfloor}\lfloor\frac{n}{kd}\rfloor^2\mu(k)\\ \text{令}T=kd\\ &=\sum_{T=1}^{n}\sum_{d\mid T}d\mu(\frac Td)\lfloor\frac nT\rfloor^2\\ &=\sum_{T=1}^{n}\lfloor\frac nT\rfloor^2\sum_{d\mid T}d\mu(\frac Td)\\ &=\sum_{T=1}^{n}\lfloor\frac nT\rfloor^2\varphi(T)\\ \end{aligned} $$

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

$$ \begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n}\text{lcm}(i,j)&=\sum_{d=1}^{n}d\sum_{i=1}^{\lfloor\frac nd\rfloor}i\sum_{j=1}^{\lfloor\frac nd\rfloor}j[\gcd(i,j)=1]\\ &=\sum_{d=1}^{n}d\times (2\times \sum_{i=1}^{\lfloor\frac nd\rfloor}i\sum_{j=1}^{i}j\sum_{k\mid i,k\mid j}\mu(k)-1)\\ &=\sum_{d=1}^{n}d\times (2\times \sum_{i=1}^{\lfloor\frac nd\rfloor}i\sum_{k\mid i}\mu(k)\sum_{j=1}^{\frac ik}jk-1)\\ &=\sum_{d=1}^{n}d\times (2\times \sum_{i=1}^{\lfloor\frac nd\rfloor}i\sum_{k\mid i}k\mu(k)\frac{\frac ik\times (\frac ik+1)}{2}-1)\\ &=\sum_{d=1}^{n}d\times (2\times \sum_{i=1}^{\lfloor\frac nd\rfloor}\frac{i^2}2\sum_{k\mid i}(\mu(k)\frac ik+\mu(k))-1)\\ &=\sum_{d=1}^{n}d\times (2\times \sum_{i=1}^{\lfloor\frac nd\rfloor}\frac{i^2}2(\varphi(i)+\epsilon(i))-1)\\ &=\sum_{d=1}^{n}d\sum_{i=1}^{\lfloor\frac nd\rfloor}i^2\varphi(i) \end{aligned} $$

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 $(f\ast g)(n)=\sum_{d\mid n}d^2\varphi(d)(\frac{n}{d})^2=n^2\sum_{d\mid n}\varphi(d)=n^3$ 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<cstdlib>
```

```
#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)%=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 $\sum_{i=a}^{b}\frac 1i\sum_{j=1}^{i}\text{lcm}(i,j)$

考虑求前缀和函数 s_n 进行差分，答案为 s_b-s_{a-1} 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）：

$$ \begin{aligned} s_n&=\sum_{i=1}^{n}\frac 1i\sum_{j=1}^{i}\text{lcm}(i,j)\\ &=\sum_{i=1}^{n}\frac 1i\sum_{j=1}^{i}\frac{ij}{\gcd(i,j)}\\ &=\sum_{i=1}^{n}\sum_{d\mid i}\frac 1d\sum_{j=1}^{i}j[\gcd(i,j)=d]\\ &=\sum_{i=1}^{n}\sum_{d\mid i}\frac 1d\sum_{j=1}^{\frac id}jd[\gcd(\frac id,j)=1]\\ &=\sum_{i=1}^{n}\sum_{d\mid i}F(\frac id)\\ 其中 F(n)&=\sum_{j=1}^{n}j[\gcd(n,j)=1]\\ &=\sum_{j=1}^{n}j\sum_{k\mid n,k\mid j}\mu(k)\\ &=\sum_{k\mid n}\mu(k)\sum_{j=1}^{\frac nk}jk\\ &=\sum_{k\mid n}\mu(k)k\frac 12\frac nk(\frac nk+1)\\ &=\frac n2\sum_{k\mid n}\mu(k)(\frac nk+1)\\ &=\frac n2(\varphi(n)+\epsilon(n))\\ 于是 s_n&=\sum_{i=1}^{n}\sum_{d\mid i}\frac d2(\varphi(d)+\epsilon(d))\\ &=\sum_{d=1}^{n}\sum_{i=1}^{\lfloor\frac nd\rfloor}\frac d2(\varphi(d)+\epsilon(d))\\ &=\frac 12(\sum_{d=1}^{n}\lfloor\frac nd\rfloor d\varphi(d)+n)\\ \end{aligned} $$

故问题转化为快速求 $f=\text{id}\cdot \varphi$ 的前缀和，令 $g=\text{id}$ 则 $(f\ast g)(n)=\sum_{d\mid n}d\varphi(d)\frac nd=n\sum_{d\mid n}\varphi(d)=n^2$ 。

```
#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，并讨论这是不是一个一一映射。

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 $p\mid x$ □ $p^{k_1}\mid x$ □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。

$$ \begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n}\sigma(ij)&=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{x\mid i}\sum_{y\mid j}\frac{iy}{x}[(x,y)=1]\\ &=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{x\mid i}\sum_{y\mid j}\frac{iy}{x}\sum_{d|x,d|y}\mu(d)\\ &=\sum_{d=1}^{n}\mu(d)\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{x\mid i,d\mid x}\sum_{y\mid j,d\mid y}\frac{iy}{x}\\ &=\sum_{d=1}^{n}\mu(d)\sum_{d\mid x}\sum_{d\mid y}\frac yx\sum_{x\mid i}\sum_{y\mid j}i\\ &=\sum_{d=1}^{n}\mu(d)\sum_{x=1}^{\lfloor\frac nd\rfloor}\sum_{y=1}^{\lfloor\frac nd\rfloor}\frac{yd}{xd}\sum_{i=1}^{\lfloor\frac{n}{xd}\rfloor}ixd\sum_{j=1}^{\lfloor\frac{n}{yd}\rfloor}\\ &=\sum_{d=1}^{n}d\mu(d)\sum_{x=1}^{\lfloor\frac nd\rfloor}\sum_{i=1}^{\lfloor\frac{n}{xd}\rfloor}i\sum_{y=1}^{\lfloor\frac nd\rfloor}\sum_{j=1}^{\lfloor\frac {n}{yd}\rfloor}y\\ &=\sum_{d=1}^{n}d\mu(d)\sum_{x=1}^{\lfloor\frac nd\rfloor}\sum_{i=1}^{\lfloor\frac{n}{xd}\rfloor}i \sum_{j=1}^{\lfloor\frac {n}{d}\rfloor}\sum_{y=1}^{\lfloor\frac n{dj}\rfloor}y\\ &=\sum_{d=1}^{n}d\mu(d)(\sum_{i=1}^{\lfloor\frac nd\rfloor}S_1(\lfloor\frac n{id}\rfloor))^2\\ 其中S_1(n)&=\frac 12(1+n)n \\ \end{aligned} $$

于是可以对 d 数论分块□ $\sum_{i=1}^{n}\frac 12\lfloor\frac ni\rfloor(\lfloor\frac ni\rfloor+1)$ 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□ $O(n^{\frac 34})$ □□另外需要快速计算出 $f=\mathrm{id}\cdot \mu$ 的前缀和，令 $g=\mathrm{id}$ □显有 $f\ast g=\epsilon$□

从上述步骤的第六步到第七步的转换，验证了等式 $\sum_{i=1}^{n}\frac 12\lfloor\frac ni\rfloor(\lfloor\frac ni\rfloor+1)=\sum_{i=1}^{n}i\lfloor\frac ni\rfloor$ □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 $\lfloor\frac ni\rfloor$ 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 $(1+p_m+p_m^2+\cdots+p_m^{k_m})$ 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：

$$ \begin{aligned} & \sum _{i=1}^n \sum _{j=1}^i (i - j) \sum _{g \mid i \land g \mid j} \mu(g) \\ = & \sum _{g=1}^n \mu(g) \sum _{i=1}^{\lfloor n/g \rfloor} \sum _{j=1}^i (i - j)g \\ = & \sum _{g=1}^n \mu(g) \times g \times F \left(\left\lfloor \frac ng \right\rfloor \right) \\ \end{aligned} $$

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

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

其中 S 为 $(\mu \cdot \mathrm{id})$ 的前缀和。

min_25 筛

实例

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

$$ \begin{aligned} & \sum _{i=1}^n f(i!) \\ = & \sum _{i=1}^n \sum _{j=1}^i f(j) \\ = & \sum _{i=1}^n (n-i+1) f(i) \\ = & (n + 1) \sum _{i=1}^n f(i) - \sum _{i=1}^n i \cdot f(i) \\ \end{aligned} $$

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：

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

第二个式子的贡献等价于统计 $[1, n]$ 中 p_i 的倍数之和：

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

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=1590463940

Last update: **2020/05/26 11:32**