数论笔记(六)

前置知识

迪利克雷卷积及其常用结论

- $\varphi * 1 = id$
- $\mu * 1 = \epsilon$
- $\mu * id = \varphi$
- $1 * 1 = d$（因数个数）
- $id * 1 = \sigma(\text{因数和})$

莫比乌斯反演及其常用结论

①$[gcd(i,j)=1]=\sum_{d|gcd(i,j)}\mu(d)$我们也可以简记为$\sum_{d|n}\mu(d)=[n=1]$

②$d(i*j)=\sum_{x|i}\sum_{y|j}[gcd(x,y)=1]$, 其中$d(i)$为$i$的约数个数

③$\sum_{i=1}^{n}\sum_{j=1}^{m}[gcd(i,j)=k]=\sum_{i=1}^{[\frac{n}{k}]}\sum_{j=1}^{[\frac{m}{k}]}[gcd(i,j)=1]$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}[gcd(i,j)=1](n<m)$$

$$=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|gcd(i,j)}\mu(d)$$

$$=\sum_{d=1}^{n}\mu(d)*[\frac{n}{d}]*[\frac{m}{d}], \text{可以数论分块}$$

④ $\sum_{i=1}^{n}\sum_{j=1}^{m} lcm(i,j)=\sum_{d=1}^{n} d\sum_{k=1}^{[\frac{n}{d}]}\mu(k)*k^2\sum_{i=1}^{[\frac{n}{dk}]} i\sum_{j=1}^{[\frac{m}{dk}]} j$

常用的自然幂数和

$$S_{id}(n)=\frac{n(n+1)}{2}$$
$$S_{id2}(n)=\frac{n(n+1)(2n+1)}{6}$$
$$S_{id3}(n)=[\frac{n(n+1)}{2}]^2$$

杜教筛

用途

可以以低于线性时间复杂度来求积性函数前缀和。

（常见积性函数：因子个数$d(x)=\sum_{i|n}1$,因子和$\sigma(x)=\sum_{i|n}i$,
欧拉函数$\varphi(x)=\sum_{i=1}^{x}[gcd(x,i)==1]$,以及莫比乌斯函数$\mu(x)$）

前导

回忆一下迪利克雷卷积的定义$f*g(n)=\sum_{d|n}f(d)g(\frac{n}{d})$

对于数论函数$f(n),g(n),h(n)$,存在$f*g=h$，我们令F,G,H为f,g,h的前缀和，

则有

$$H(x)=\sum_{n\le x}h(n)=\sum_{n\le x}\sum_{d|n}f(d)g(\frac{n}{d})\text{（迪利克雷卷积定义）}$$
$$=\sum_{d=1}^{x}\sum_{n=1}^{\lfloor x/d\rfloor}f(d)g(n)\text{（枚举}d\text{）}$$
$$=\sum_{n\le x}f(n)G(\lfloor x/n\rfloor)=\sum_{n\le x}g(n)F(\lfloor x/n\rfloor)\text{（转化为}g\text{或}f\text{的前缀和）}$$

代入$x=n$可得$g(1)F(n)=H(n)-\sum_{i=2}^{n}g(i)F(\lfloor n/i\rfloor)$

也就是说如果我们能快速求出$H(x),G(x)$，就可以快速求出$F(x)$了

另外的推导

$$\text{设积性函数}f\text{的前缀和}\sum_{i=1}^{n}f(i)=S(n)$$
$$\text{再找一个积性函数}g\text{，则考虑它们的狄利克雷卷积的前缀和}$$
$$\sum_{i=1}^{n}(f*g)(i)=\sum_{i=1}^{n}\sum_{d|i}f(d)g(\frac{i}{d})=\sum_{d=1}^{n}g(d)\sum_{i=1}^{[\frac{n}{d}]}f(i)=\sum_{d=1}^{n}g(d)S([\frac{n}{d}])$$
$$\text{拿到核心式子}g(1)S(n)=\sum_{i=1}^{n}(f*g)(i)-\sum_{i=2}^{n}g(i)S([\frac{n}{i}])$$

我们找到一个合适的积性函数g，就可以快速算出$(f*g)$以及g的前缀和，就可以用数论分块去求解。

框架

```
ll GetSum(int n) { // 算 f 前缀和的函数
  ll ans = f_g_sum(n); // 算 f * g 的前缀和
  // 以下这个 for 循环是数论分块
  for(ll l = 2, r; l <= n; l = r + 1) { // 注意从 2 开始
    r = (n / (n / l));
    ans -= (g_sum(r) - g_sum(l - 1)) * GetSum(n / l);
    // g_sum 是 g 的前缀和
    // 递归 GetSum 求解
  } return ans;
} //复整体复杂度O(n^(3/4))
```

luoguP4213 【模板】杜教筛（Sum）

给定t组n,球$ans_1 = \sum_{i=1}^{n} \varphi(i)$以及$ans_2 = \sum_{i=1}^{n} \mu(i)$

$$\text{推导：要利用公式} f * g = h, F(n) = H(n) - \sum_{d=2}^{n} g(d)F(\lfloor n/d \rfloor)$$

$$\text{可以使} \varphi = f, g = 1, h(n) = f * g(n) = \sum_{d|n}^{n} f(d)g(\lfloor n/d \rfloor) = n$$

$$F(n) = \sum_{i=1}^{n} \varphi(i) = \frac{n(n+1)}{2} - \sum_{i=2}^{n} F(\lfloor n/i \rfloor)$$

$$\text{同样的，令} f = \mu, g = 1,\ h = \epsilon \text{即可求出答案。}$$

$$F(n) = 1 - \sum_{i=2}^{n} g(i)F(\lfloor n/i \rfloor)$$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e6+7;

bool is_pri[N];
int pri[N],mu[N],cnt=0;
ll sum1[N],sum2[N],phi[N];
map<ll,ll>Phi;
map<int,int>Mu;

void init(){
    phi[1]=mu[1]=1;
    for(int i=2;i<N;++i){
        if(!is_pri[i]) pri[++cnt]=i,mu[i]=-1,phi[i]=i-1;
        for(int j=1;j<=cnt&&i*pri[j]<N;++j){
            is_pri[i*pri[j]]=1;
            if(i%pri[j]){
                mu[i*pri[j]]=-mu[i];
                phi[i*pri[j]]=phi[i]*(pri[j]-1);
            }else{
```

```
                phi[i*pri[j]]=phi[i]*pri[j];
                break;
            }
        }
    }
    for(int i=1;i<N;++i){
        sum1[i]=sum1[i-1]+mu[i];
        sum2[i]=sum2[i-1]+phi[i];
    }
}

int t,n;

int Djmu(ll x){
    if(x<N) return sum1[x];
    if(Mu.count(x)) return Mu[x];
    int res=1;
    for(ll l=2,r;l<=x;l=r+1){
        r=min(x,x/(x/l));
        res-=(ll)(r-l+1)*Djmu(x/l); //容斥
    }
    return Mu[x]=res;
}

ll Djphi(ll x){
    if(x<N) return sum2[x];  //根号以内直接给答案
    if(Phi.count(x)) return Phi[x]; //记忆化
    ll res=(ll)x*(x+1)/2;  //f*g=id的前缀和
    for(ll l=2,r;l<=x;l=r+1){
        r=min(x,x/(x/l));
        res-=(ll)(r-l+1)*Djphi(x/l);
    }
    return Phi[x]=res;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    init();
    cin>>t;
    while(t--){
        cin>>n;
        cout<<Djphi(n)<<" "<<Djmu(n)<<"\n";
    }
    return 0;
}
```

Min25筛

依然是求积性函数前缀和。设积性函数f在p^c的取值是一个多项式，令$minprime_i$表示能整除i的最小质数，即i的最小质因数。于是有

$$\sum_{i=1}^{n} f(i) = 1 + \sum_{2\le p^c\le n,p\text{是质数}} f(p^c)(1 + \sum_{minp_x>p,2\le x\le \frac{n}{p^c}} f(x))$$

即提出最小质因子加速计算。注意到合数$minp_i \le \sqrt{n}$，所以可拆成

$$\sum_{2\le p^c\le n, p是质数, p\le\sqrt{n}} f(p^c)(1+\sum_{minp_x>p, 2\le x\le\frac{n}{p^c}} f(x))+\sum_{p是质数, \sqrt{n}<p\le n} f(p)$$

令$g_{n,m}=\sum_{minp_x>m, 2\le x\le n} f(x), h_n=\sum_{p是质数, 2<p\le n} f(p)$，$g_{n,0}$为所求解，

$g_{n,m}=\sum_{p^c\le n, p是质数, m<p\le\sqrt{n}} f(p^c)(1+g_{\frac{n}{p^c},p})+h_n-h_{\sqrt{n}}$

能快速求$g_{n,m}$和h_n就能快速求$\sum f_i$

注意:在筛h_i时，要求f_i为完全积性，μ就不可以min25求解。

P5325 【模板】Min_25筛

定义积性函数$f(x)$，且$f(p^k)=p^k(p^k-1)$（p是一个质数），求$\sum_{i=1}^n f(i)$对10^9+7取模。

```
#include<bits/stdc++.h>
#define int long long
using  namespace std;
const int N=1e6+7;
const int mod=1e9+7,inv2=500000004,inv3=333333336;

int pri[N],np[N],cnt=0,sp1[N],sp2[N];
int n,sqr,tot,g1[N],g2[N],w[N],ind1[N],ind2[N];

void init(int mx){  //预处理，线性筛
    np[1]=1;
    for(int i=1;i<=mx;++i){
        if(!np[i]){
            pri[++cnt]=i;
            sp1[cnt]=(sp1[cnt-1]+i)%mod;
            sp2[cnt]=(sp2[cnt-1]+1ll*i*i)%mod;
        }
        for(int j=1;j<=cnt&&pri[j]*i<=mx;++j){
            np[pri[j]*i]=1;
            if(i%pri[j]==0) break;
        }
    }
}

int S(int x,int y){  //S(n,x)表示求1到n中所有质因子大于px的函数值之和。
    if(pri[y]>=x) return 0;
    int k=x<=sqr?ind1[x]:ind2[n/x];
    int ans=(g2[k]-g1[k]+mod-(sp2[y]-sp1[y])+mod)%mod;
    for(int i=y+1;i<=cnt&&pri[i]*pri[i]<=x;++i){
        int pe=pri[i];
        for(int e=1;pe<=x;++e,pe=pe*pri[i]){
            int xx=pe%mod;
            ans=(ans+xx*(xx-1)%mod*(S(x/pe,i)+(e!=1)))%mod;
        }
    }
    return ans%mod;
}
```

```
signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n;
    sqr=sqrt(n);
    init(sqr);
    for(int i=1;i<=n;){  //数论分块
        int j=n/(n/i);
        w[++tot]=n/i;
        g1[tot]=w[tot]%mod;
        g2[tot]=g1[tot]*(g1[tot]+1)/2%mod*(2*g1[tot]+1)%mod*inv3%mod;
        g2[tot]--;
        g1[tot]=g1[tot]*(g1[tot]+1)/2%mod-1;
        if(n/i<=sqr) ind1[n/i]=tot;
        else ind2[n/(n/i)]=tot;
        i=j+1;
    }//g1,g2分别表示一次项和二次项，ind1和ind2用来记录这个数在数组中的位置
    for(int i=1;i<=cnt;++i){
        for(int j=1;j<=tot&&pri[i]*pri[i]<=w[j];++j){
            int k=w[j]/pri[i]<=sqr?ind1[w[j]/pri[i]]:ind2[n/(w[j]/pri[i])];
            g1[j]-=pri[i]*(g1[k]-sp1[i-1]+mod)%mod;
            g2[j]-=pri[i]*pri[i]%mod*(g2[k]-sp2[i-1]+mod)%mod;
            g1[j]%=mod;g2[j]%=mod;
            if(g1[j]<0) g1[j]+=mod;
            if(g2[j]<0) g2[j]+=mod;
        }
    }
    cout<<(S(n,0)+1)%mod;
    return 0;
}
```

质数的c次幂前缀和

- 定理1：合数q必有一个$\lfloor(\sqrt{q})\rfloor$以内的因子。

因此利用$\lfloor\sqrt{N}\rfloor$以内的质数就可以筛除N以内的所有质数。

定义$S_{n,m}$为$(\lfloor\sqrt{N},n\rfloor)$以内的素数集合,且$\lfloor\sqrt{N}\rfloor$内有m个质数。h(n,k)为筛除k轮后n以内剩余的数的c次方和。即$h(n,k)=\sum_{x\in S_{n,k}}x^c$

那么$h(N,m)-1+\sum_{p\in P_m}p^c$即为答案。

洲阁筛

也就是扩展Eratosthenes筛，用于解决较一般情况下的积性函数求和问题。复杂度为$O(\frac{n^{3/4}}{logn})$，常数较大。使用条件：$f(x)$为积性函数且$f(p^k)$为关于p,k的多项式。

感觉可以用常数较小的min_25去代替，复杂度是一样的。

习题

[luoguP6021 【模板】质数前缀统计]

给定$n \le 1e10$，求式子$\sum_{i=1}^{\lfloor\sqrt{n}\rfloor} i^2 S(\lfloor\frac{N}{i}\rfloor)$。

[Loj#6235. 区间质数个数]

求$n \le 1e11$以内的素数个数，min25模板题。

```
#include<bits/stdc++.h>
//#define int long long
using namespace std;
typedef long long ll;
const int N=1e6+7;

ll n,h0[N],h1[N];

signed main(){
    scanf("%lld",&n);
    int bk=sqrtl(n);
    for(int i=1;i<=bk;++i){
        h1[i]=n/i;h0[i]=i;
    }
    for(int i=2;i<=bk;++i){
        if(h0[i]==h0[i-1]) continue;
        ll x0=h0[i-1],r=(ll)i*i;
        int u=min(ll(bk),n/((ll)i*i)),uu=min(u,bk/i);
        for(int j=1;j<=uu;++j) h1[j]-=h1[j*i]-x0;
        ll t=n/i;
        for(int j=uu+1;j<=u;++j) h1[j]-=h0[t/j]-x0;
        for(int j=bk;j>=r;--j) h0[j]-=h0[j/i]-x0;
    }
    printf("%lld\n",h1[1]-1);
}
```

P3768 简单的数学题

给定n,p，求$(\sum_{i=1}^{n}\sum_{j=1}^{n} i*j*gcd(i,j))$

$$\sum_{d=1}^{n} d\sum_{i=1}^{n}\sum_{j=1}^{n} ij[gcd(i,j)==d]$$

$$\text{提取因子} = \sum_{d=1}^{n} d^3\sum_{i=1}^{n/d}\sum_{j=1}^{n/d} ij[gcd(i,j)==1]$$

$$\text{反演} = \sum_{d=1}^{n} d^3\sum_{i=1}^{n/d}\mu(i)i^2(1+2+\ldots+\lfloor\frac{n}{di}\rfloor)^2$$

$$\text{令}T=di\text{，原式} = \sum_{T=1}^{n} sum(\frac{n}{T})^2T^2\sum_{d|T} d\mu(\frac{T}{d})$$

由迪利克雷卷积$T^2\sum_{d|T}d\mu(\frac{T}{d})=T^2\varphi(T)$

最终推得$ans=\sum_{T=1}^{n}sum(\frac{n}{T})^2T^2\sum_{d|T}d\varphi(\frac{T}{d})$

前面直接数论分块，后面套一个杜教筛在低于线性时间内求前缀和即可。

```
#include<bits/stdc++.h>
//#define int long long
using namespace std;
const int N=8e6+1;
typedef long long ll;

ll p,n,inv2,inv6;
map<ll,ll>mp;

inline ll fpow(ll a,int b){
    ll res=1;
    while(b){
        if(b&1) res=res*a%p;
        a=a*a%p;
        b>>=1;
    }
    return res;
}

bool np[N];
ll phi[N];
int pri[N],mu[N],cnt=0;

void init(){
    np[1]=mu[1]=phi[1]=1;
    for(int i=2;i<N;++i){
        if(!np[i]){
            pri[++cnt]=i;
            mu[i]=-1;
            phi[i]=i-1;
        }
        for(int j=1;j<=cnt&&pri[j]*i<N;++j){
            np[pri[j]*i]=1;
            if(i%pri[j]==0){
                phi[i*pri[j]]=phi[i]*pri[j];
                mu[i*pri[j]]=0;
                break;
            }else{
                phi[i*pri[j]]=phi[i]*(pri[j]-1);
                mu[i*pri[j]]=-mu[i];
            }
        }
    }
    for(int i=1;i<N;++i) phi[i]=(phi[i-1]+phi[i]*i%p*i%p)%p;
}

inline ll Sum(ll x){x%=p;return 1ll*x*(x+1)%p*inv2%p;}
inline ll Sump(ll x){x%=p;return 1ll*x*(x+1)%p*(x+x+1)%p*inv6%p;}

inline ll SF(ll x){
    if(x<N) return phi[x];
    if(mp[x]) return mp[x];
```

```
    ll res=Sum(x);
    res=res*res%p;
    for(ll l=2,r;l<=x;l=r+1){
        r=x/(x/l);
        ll tt=(Sump(r)-Sump(l-1))%p;
        res=(res-SF(x/l)*tt%p+p)%p;
    }
    return mp[x]=res;
}

signed main(){
    scanf("%lld%lld",&p,&n);
    init();
    inv2=fpow(2,p-2),inv6=fpow(6,p-2);
    ll ans=0;
    for(ll l=1,r;l<=n;l=r+1){
        r=n/(n/l);
        ll tt=Sum(n/l);tt=tt*tt%p;
        ll gg=(SF(r)-SF(l-1))%p;
        ans=(ans+gg*tt%p)%p;
    }
    printf("%lld\n",(ans+p)%p);
    return 0;
}
```

参考资料

https://www.luogu.com.cn/blog/command-block/min25-shai-xiao-ji

https://www.cnblogs.com/A2484337545/p/14682184.html

《简单易懂的质数筛法》 陈牧歌

https://www.luogu.com.cn/blog/cjyyb/solution-p3768