数论基础模板

全家桶

```
#pragma GCC optimize("Ofast", "inline", "-ffast-math")
#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
//#define int long long
using namespace std;
const int N=2e5+7;
const int mod=998244353;
typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;
typedef __int128 lll;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(ll x){if(x>9) write(x/10);putchar(x%10+'0');}
template<class type_name> inline type_name qr(type_name sample){
    type_name ret=0,sgn=1;
    char cur=getchar();
    while(!isdigit(cur)) sgn=(cur=='-'?-1:1),cur=getchar();
    while(isdigit(cur)) ret=(ret<<1)+(ret<<3)+cur-'0',cur=getchar();
    return sgn==-1?-ret:ret;
}

int n,inv[N],frac[N],ifrac[N];

int cnt=0,np[N],pri[N],phi[N],mu[N];
void init(){   //线性筛求质数、欧拉函数、莫比乌斯函数
    np[1]=phi[1]=mu[1]=1;
    for(int i=2;i<N;++i){
```

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

//--------------------------------------------
inline ll gcd(ll a,ll b){  //最大公约数
    return b?gcd(b,a%b):a;
}

inline ll lcm(ll a,ll b){  //最小公倍数
    return a*b/gcd(a,b);
}

inline int exgcd(int a,int b,int &x,int &y){  //扩展欧几里得
    //对于ax+by=c等价于ax ≡c mod b
    if(!b){
        x=1;y=0;
        return a;
    }
    int d=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}

inline int inv_exgcd(int a,int b){int x,y;exgcd(a,b,x,y);return (x%b+b)%b;}

/*
int mul(int a,int b,int p){  //普通的快速乘
    int res=0;
    while(b){
        if(b&1) res=(res+a)%p;
        a=(a+a)%p;
        b>>=1;
    }
    return res;
}*/
int mul(int a, int b, int p){ //利用自然溢出的快速乘
    ull c=(ull)a*b-(ull)((ld)a/p*b+0.5L)*p; //自动溢出
    return c<p?c:c+p; //转化为正数
}

inline ll fpow(ll a,ll b,ll p=mod){  //快速幂
    ll res=1;
```

```
    while(b){
        if(b&1) res=(__int128)res*a%p;
        a=(__int128)a*a%p;
        b>>=1;
    }
    return res;
}
//------------------------------------------
inline int C(int n,int m){  //求组合数
    if(n<m) return 0;
    return frac[n]*ifrac[m]%mod*ifrac[n-m]%mod;
}

inline int A(int n,int m){  //求排列数
    if(n<m) return 0;
    return frac[n]*ifrac[n-m]%mod;
}

inline int Lucas(int x,int y){  //卢卡斯定理求大组合数
    if(!y) return 1;
    return C(x%mod,y%mod)*Lucas(x/mod,y/mod)%mod;
}

inline int katelan(int x){   //求卡特兰数;
    //前几项为1,2,5,14,132,429,1430,4862,16796,58786,208012,742900
    //常用公式:1.f(n)=C(2n,n)/(n+1)
    //          2.f(n)=f(n-1)*(4*n-2)/(n+1)
    //          3.for(int i=0;i<=n-1;++i) f[n]+=f[i]*f[n-i-1];
    //          4.f(n)=C(2n,n)-C(2n,n-1)
    //常见形式：二叉树计数、乘法加括号、AB排列问题、多边形分割
    return C(x<<1,x)*inv[x+1]%mod;
}

//------------------------------------------
int BSGS(int a,int b,int p){ //解高次同余方程 a^x ≡b(mod p)
   if((a%p==0&&b%p!=0)||(a%p!=0&&b%p==0)) return -1;
    map<int,int>mp;
    int t=ceil(sqrt(p*1.0));
    for(int i=0,s=1;i<t;++i,s=s*a%p){
        int x=b*s%p;
        mp[x]=i;
    }
    int ans=0,tmp=fpow(a,t,p);
    for(int i=0,k=1;i<=t;++i,k=k*tmp%p){
        if(mp.count(k)&&i*t-mp[k]>=0){
            ans=i*t-mp[k];
            return ans;
        }
    }
    return -1;
}

int exBSGS(int a,int b,int p){  //扩展BSGS
    if(b==1||p==1) return 0;  //特殊情况,x=0时最小解
    int g=gcd(a,p),k=0,na=1;
    while(g>1){
        if(b%g!=0) return -1;  //无法整除则无解
        ++k;b/=g;p/=g;na=na*(a/g)%p;
```

```
        if(na==b) return k; //na=b说明前面的a次数为0，只需返回k
        g=gcd(a,p);
    }
    int f=BSGS(a,b*inv_exgcd(na,p)%p,p);
    if(f==-1) return -1;
    return f+k;
}

//--------------------------------------------
const int pp[]={2,3,5,7,11,13,17,19,23,29,61}; //不打表可以直接用这个
bool MillerRabin(int x){ //米勒拉宾质数判定
    if (x<2) return 0;
    int i,j,y=x-1,a,b;
    while(y&1^1) y>>=1; //相当于先化为奇数
    for(i=0;i<11;i++){
        if (x%pp[i]<1)  return x==pp[i]; //不可以是质数p的倍数
    }
    for(i=2;i<11;i++){
        for (a=b=fpow(pp[i],y,x),j=y;j<x&&a>1;j+=j,a=b){
            if (b=mul(b,b,x),b==1&&a!=x-1)  return 0; //二次检测
            //b和x互质，b^2modx=1，要b=1,要么b=x-1,那么b==1时a应该为x-1
        }
        if(a!=1) return 0; //最终a不为1，没有通过二次检验
    }
    return 1;
}

inline bool mr(ll x,ll b){  //另一种写法
    ll k=x-1;
    while(k){
        ll cur=fpow(b,k,x);
        if(cur!=1&&cur!=x-1) return false;
        if((k&1)==1||cur==x-1) return true;
        k>>=1;
    }
    return true;
}

inline bool is_prime(ll x){
    if(x==46856248255981ll||x<2) return false;
    if(x==2||x==3||x==7||x==61||x==24251) return true;
    return mr(x,2)&&mr(x,61);
}

inline ll PollardRho(ll x){ //Pollard Rho算法求素因子
    ll s=0,t=0,c=1ll*rand()%(x-1)+1;
    int stp=0,goal=1;
    ll val=1;
    for(goal=1;;goal<<=1,s=t,val=1){
        for(stp=1;stp<=goal;++stp){
            t=((lll)t*t+c)%x;
            val=(lll)val*abs(t-s)%x;
            if((stp%127)==0){
                ll d=gcd(val,x);
                if(d>1) return d;
            }
        }
        ll d=gcd(val,x);
```

```
        if(d>1) return d;
    }
}

ll max_factor=0;
inline void fac(ll x){ //求最大因子
    if(x<=max_factor||x<2) return;
    if(is_prime(x)){//if(MillerRabin(x)){
        max_factor=max_factor>x?max_factor:x;
        return;
    }
    ll p=x;
    while(p>=x) p=PollardRho(x);
    while((x%p)==0) x/=p;
    fac(x),fac(p);
}

//------------------------------------------

signed main(){
//  init();
//  frac[0]=1;for(int i=1;i<=n;++i) frac[i]=frac[i-1]*i%mod;  //求阶乘
//  ifrac[n]=fpow(frac[n],mod-2);
//  for(int i=n;i>=1;--i) ifrac[i-1]=ifrac[i]*i%mod;  //线性求阶乘的逆元
//  inv[0]=inv[1]=1;for(int i=2;i<=n;++i) inv[i]=inv[mod%i]*(mod-mod/i)%mod;  //
线性求逆元
    srand((unsigned)time(NULL));
    int T=qr(1);
    while(T--){
        ll nn=qr(1ll);
        max_factor=0; fac(nn);
        if(nn==max_factor) puts("Prime");
        else write(max_factor),putchar('\n');
    }
    return 0;
}
```

求解线性同余方程

```
//对于方程ax+by=c,等同于 ax≡c(mod b)，有整数解的充要条件为c%gcd(a,b)=0
//对于方程任意一组解可表示为：x=x0+bt,y=y0-at
inline int exgcd(int a,int b,int &x,int &y){ //扩展欧几里得
    if(!b){
        x=1;y=0;
        return a;
    }
    int d=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}
```

中国剩余定理(CRT)

用于解决同余方程组如下(找到最小整数解x)，保证模数**两两互质**：

$$\begin{cases} x \equiv a_1 \mod m_1 \\ x \equiv a_2 \mod m_2 \\ \ldots\ldots \\ x \equiv a_n \mod m_n \end{cases}$$

我们设$M = \prod_{i=1}^{n} m_i, M_i = \frac{M}{m_i}, M_i t_i \equiv 1 \mod m_i$

那么可以构造一组解为$ans = \sum_{i=1}^{n} a_i M_i t_i$

luogu P1495 【模板】中国剩余定理（CRT）/ 曹冲养猪

```
inline int exgcd(int a,int b,int &x,int &y){
    if(!b){
        x=1;y=0;
        return a;
    }
    int d=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}

inline int inv(int a,int b){
    int d,x,y;
    d=exgcd(a,b,x,y);
    return (x<0)?(x+b):x;
}

void Solve(){
    n=read();
    for(int i=1;i<=n;++i){
        a[i]=read();b[i]=read();
        M*=a[i];
    }
    int ans=0;
    for(int i=1;i<=n;++i){
        m[i]=M/a[i];
        t[i]=inv(m[i],a[i]); //找m[i]在a[i]意义下的逆元
        ans+=b[i]*m[i]%M*t[i]%M;
        ans%=M;
    }
    write(ans);putchar('\n');
}
```

扩展中国剩余定理（EXCRT）

当模数不互质时，中国剩余定理不能解同余方程组，需要按照以下流程求解：

- 读入所有方程组
- 弹出两个方程，先判断有无解

- 无解：报告异常
  - 有解：合并成同一个方程，如何压入方程组
- 执行上述步骤2，直到只剩下一个方程，即为解系。

P4777 【模板】扩展中国剩余定理（EXCRT）

```
int mul(int a,int b,int mod){
    int res=0;
    while(b){
        if(b&1) res=(res+a)%mod;
        a=(a+a)%mod;
        b>>=1;
    }
    return res;
}

inline int gcd(int a,int b){
    return b?gcd(b,a%b):a;
}

inline int lcm(int a,int b){
    return a*b/gcd(a,b);
}

inline int exgcd(int a,int b,int &x,int &y){
    if(!b){
        x=1;y=0;
        return a;
    }
    int d=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}

inline int excrt(){
    int x,y,M=a[1],ans=b[1];
    for(int i=2;i<=n;++i){
        int A=M,B=a[i],C=(b[i]-ans%B+B)%B;  //Ax≡C(mod B)
        int g=exgcd(A,B,x,y),bg=B/g;
        if(C%g!=0) return -1; //判是否无解
        x=mul(x,C/g,bg);
        ans+=x*M; //更新前k个方程组的答案
        M*=bg;  //M为前k个m的lcm
        ans=(ans%M+M)%M;
    }
    return (ans%M+M)%M;
}

void Solve(){
    cin>>n;
    for(int i=1;i<=n;++i) cin>>a[i]>>b[i];
    cout<<excrt()<<"\n";
}
```

GCD的优化

```
inline int gcd(int a,int b){  //普通的求gcd
    return b?gcd(b,a%b):a;
}

inline int GCD(int x,int y){  //二进制算法求gcd
    int i,j;
    if(!x) return y;
    if(!y) return x;
    for(i=0;!(x&1);++i) x>>=1;  //去掉所有2
    for(j=0;!(y&1);++j) y>>=1;
    if(j<i) i=j;
    while(1){
        if(x<y) x^=y^=x^=y; //交换x,y
        if((x-=y)==0) return (y<<i);
        //若x==y,gcd==x==y实际上是辗转减
        while((x&1)==0) x>>=1; //去掉所有2
    }
}
```

快速乘的优化

```
int mul(int a,int b,int p){  //普通的快速乘
    int res=0;
    while(b){
        if(b&1) res=(res+a)%p;
        a=(a+a)%p;
        b>>=1;
    }
    return res;
}

ll mul(ll a,ll b,ll mod){   //利用unsigned 自然溢出的性质，只有O(1)
    ull c=(ull)a*b-(ull)((ld)a/mod*b+0.5L)*mod;
} //其中ull是unsigned long long，ld是long double类型
```

线性筛法

线性筛求欧拉函数

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

```
                phi[i*pri[j]]=pri[j]*phi[i]; //可以手动证一下
                break;
            }else phi[i*pri[j]]=(pri[j]-1)*phi[i]; //互质，积性
        }
    }
}
```

线性筛法求莫比乌斯函数

定义莫比乌斯函数

$$\mu(n)=\begin{cases}1, n=1\\(-1)^k, n=p_1p_2...p_k\\0, 其他\end{cases}$$

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

线性筛法求约数个数

d[i]:表示i的约数个数

num[i]:表示i的最小素因子个数

pri[i]:表示第i个素数

```
void init(){
    np[1]=1;d[1]=1;
    for(int i=2;i<N;i++){
        if(!np[i]){
            pri[++cnt]=i;
            num[i]=1;
            d[i]=2;//只有1和他本身
        }
        for(int j=1;j<=cnt&&pri[j]*i<N;j++){
            np[i*pri[j]]=1;
            if(i%pri[j]){ //互质时直接乘素因子的约数个数即可
                d[i*pri[j]]=d[i]*d[pri[j]]; //d[pri[j]]=2
                num[i*pri[j]]=1;
            }else{ //这里推一推约数个数定理，就是除一个数乘一个数
                num[i*pri[j]]=num[i]+1;
                d[i*pri[j]]=d[i]/(num[i]+1)*(num[i]+2)
                break;
            }
```

```
        }
    }
}
```

线性筛法求约数和

sd[i]：表示i的约数和

sp[i]：表示i的最小素因子的等比数列之和。

pri[i]表示第i个素数

```
void init(){
    np[1]=sd[1]=1;
    for(int i=2;i<N;i++){
        if(!np[i]){
            pri[++cnt]=i;
            sd[i]=i+1;
            sp[i]=i+1;
        }
        for(int j=1;j<=cnt&&pri[j]*i<N;j++){
            np[pri[j]*i]=1;
            if(i%pri[j]){ //如果互质，实际上是在后面乘上(1+pri[j])
                sd[i*pri[j]]=sd[i]*sd[pri[j]];
                sp[i*pri[j]]=pri[j]+1;
            }else{ //看着约数和公式推就好，就是乘一项除一项
                sp[i*pri[j]]=sp[i]*pri[j]+1;
                sd[i*pri[j]]=sd[i]/sp[i]*sp[i*pri[j]];
                break;
            }
        }
    }
}
```