# min25模板

```
//代码计算的是函数f(i)的前n项和
#include<bits/stdc++.h>
#define LL long long
using namespace std;

const int N = 2e5 + 10;//2*sqrt(n)的范围
const LL INV2 = 5e8 +4;//2的逆元(可能用到)
const LL INV6 = 166666668;//6的逆元(可能用到)
const LL MOD = 1e9 + 7;

bool isp[N];
LL n,m,sz,sqrt_n,c0,c1,p[N],w[N],id0[N],id1[N],g0[N],g1[N],sum0[N],sum1[N];
//n是输入的数，sqrt_n保存sqrt(n)，即预处理的数量
//sz是质数个数，isp[i]表示i是否质数，p[i]存储第i个质数
//sum0[i]存储的是前i个质数的f0值之和,sum1[i]存储的是前i个质数的f1值之和
//m是n/k的个数，w[i]存储n/k第i种值(倒序)，id0和id1[i]存储i这个值在w[i]中的下标
//g0[i]和g1[i]等分别存储f在取质数时的多项式中的不同次方项(此处只有两个数组，即假设题目中的f取质数时只有两项)，g0[i]存的是g(w[i],0~sz)，g1[i]存的是g1(w[i],0~sz)
//g0(n,i) = \Sigma_{j=1}^{n}[j是质数 or j的最小质因子>p[i]]*f0(j) 其中f0为f取质数时的第一个次方项
```

```cpp
//g1(n,i) = \Sigma_{j=1}^{n}[j是质数 or j的最小质因子>p[i]]*f1(j) 其中f1为f取质数时的第二个次方项
//c0和c1等保存的是不同次方项的系数(此处只有两个系数，即假设题目中的f取质数时只有两项)

//计算f(p^t)，若要降低常数也可把这个函数用增量法在调用处实现
inline LL f(LL p,LL t)
{
    //...
}

//线性筛，求函数f0、f1在前i个质数处的前缀和
void init(LL n)
{
    sz=0;
    memset(isp,1,sizeof(isp));
    isp[1]=0;
    sum0[0]=0;
    sum1[0]=0;
    for (LL i=2; i<=n; i++)
    {
        if (isp[i])
        {
            p[++sz]=i;
            //计算sum0，即sum0(i) = \Sigma_{j=1}^{i}f0(p[j])
            //...
            //计算sum1，即sum1(i) = \Sigma_{j=1}^{i}f1(p[j])
            //...
        }
```

```
        for (int j=1; j<=sz&&p[j]*i<=n; j++)
        {
            isp[i*p[j]]=0;
            if (i%p[j]==0) break;
        }
    }
}

inline int get_id(LL x) {
    if(x<=sqrt_n) return id0[x];
    else return id1[n/x];
}

//计算原理中的多项式的项，只会计算g0(n/i),g1(n/i)
void sieve_g(LL n)
{
    m=0;
    for (LL i=1,j;i<=n;i=j+1)
    {
        LL k=n/i; j=n/k;
        w[++m]=k;
        if(k<=sqrt_n) id0[k]=m;
        else id1[n/k]=m;

        k%=MOD;
        //计算原理中的g0(w[m],0)，即\Sigma_{j=2}^{w[m]}f0(j)，存在g0[m]中
        //...
        //计算原理中的g1(w[m],0)，即\Sigma_{j=2}^{w[m]}f1(j)，存在g1[m]中
        //...
```

```
    }
    for (int i=1;i<=sz;i++)
        for (int j=1;j<=m&&p[i]*p[i]<=w[j];j++)
        {
            int op=get_id(w[j]/p[i]);
            //根据g0[j]=(g0[j]-f0(p[i])*((g0[op]-sum0[i-1]+MOD)%MOD)%MOD+MOD)%MOD计算
            //...
            //根据g1[j]=(g1[j]-f1(p[i])*((g1[op]-sum1[i-1]+MOD)%MOD)%MOD+MOD)%MOD计算
            //...
        }
}

LL S(LL x,LL y)
{
    if (x<=1||p[y]>x) return 0;//base case
    LL k=get_id(x),res=0;
    res=((c0*g0[k]%MOD+c1*g1[k]%MOD+MOD)%MOD-(c0*sum0[y-1]%MOD+c1*sum1[y-1]%MOD+MOD)%MOD+MOD)%MOD;//质数部分的贡献
    //下面的二重循环统计的是合数部分的贡献
    for(int i=y;i<=sz&&p[i]*p[i]<=x;i++)//枚举合数的最小质因子
    {
        LL t0=p[i], t1=p[i]*p[i];
        for(LL e=1; t1<=x; t0=t1,t1*=p[i],e++)//枚举最小质因子的次数
        {
```

```
                LL fp0=f(p[i],e), fp1=f(p[i],e+1);
                (res+=(fp0*S(x/t0,i+1)%MOD+fp1)%MOD)%=MOD;
            }
    }
    return res;
}

int main()
{
    //freopen("test.in","r",stdin);
    scanf("%lld",&n);
    sqrt_n=sqrt(n);
    init(sqrt_n); sieve_g(n);
    //此处对不同次项的系数c0,c1进行直接赋值
    //...
    //此处计算的是原函数f在取值为1时的函数值，即f(1)，存在f_1中；若是积性函数的话一般有f(1)=1
    //...
    printf("%lld\n",((S(n,1)+f_1)%MOD));
    return 0;
}

```

$$求\sum p \tag{1}$$

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int N = 1000010;
int prime[N], id1[N], id2[N], flag[N], ncnt, m;
```

```
ll g[N], sum[N], a[N], T;
ll n;
int ID(ll x) {
    return x <= T ? id1[x] : id2[n / x];
}
ll calc(ll x) {
    return x * (x + 1) / 2 - 1;
}
ll f(ll x) {
    return x;
}
ll init(ll n){
    T = sqrt(n + 0.5);
    for (int i = 2; i <= T; i++) {
        if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
        for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {
            flag[i * prime[j]] = 1;
            if (i % prime[j] == 0) break;
        }
    }
    for (ll l = 1; l <= n; l = n / (n / l) + 1) {
        a[++m] = n / l;
        if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
        g[m] = calc(a[m]);
    }
    for (int i = 1; i <= ncnt; i++)
        for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)
```

```
            g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] - sum[i - 1]);
}
ll solve(ll x){
    if(x<=1){return x;}
    return n=x,init(n),g[ID(n)];
}
int main() {
    while(1)
    {
        memset(g,0,sizeof(g));
        memset(a,0,sizeof(a));
        memset(sum,0,sizeof(sum));
        memset(prime,0,sizeof(prime));
    memset(id1,0,sizeof(id1));
    memset(id2,0,sizeof(id2));
    memset(flag,0,sizeof(flag));
    ncnt=m=0;
    scanf("%lld", &n);
    printf("%lld\n", solve(n));
    }
}
```