

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 ）

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

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：

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

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。


```

#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;
}
```

min_25 筛

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=1590414153

Last update: 2020/05/25 21:42