

数论 3

杜教筛

算法简介

一种 $O(n^{\frac{2}{3}})$ 计算积性函数前缀和的算法。

算法思路

设 f, g 为积性函数 $S(n) = \sum_{i=1}^n f(i)$ 考虑 f, g 的狄利克雷卷积的前缀和

$$\sum_{i=1}^n (f * g)(i) = \sum_{i=1}^n \sum_{d \mid i} f(\frac{i}{d})g(d) = \sum_{d=1}^n \left(g(d) \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} f(k) \right) = \sum_{d=1}^n g(d)S(\lfloor \frac{n}{d} \rfloor)$$

所以有

$$\sum_{i=1}^n (f * g)(i) = g(1)S(n) + \sum_{d=2}^n g(d)S(\lfloor \frac{n}{d} \rfloor)$$

移项得

$$g(1)S(n) = \sum_{i=1}^n (f * g)(i) - \sum_{d=2}^n g(d)S(\lfloor \frac{n}{d} \rfloor)$$

观察式子，发现如果能快速求出 $(f * g)(n)$ 和 $g(n)$ 的前缀和，就可以通过整数分块和记忆化搜索快速求出 $S(n)$

复杂度证明

下面假设 $(f * g)(n)$ 和 $g(n)$ 的前缀和可以 $O(1)$ 求出。

若要求出 $S(n)$ 需要先求出 $S(\lfloor \frac{n}{d} \rfloor) (d=2 \sim n)$

事实上，有 $\{x \mid \exists d \left((2 \leq d \leq n) \wedge \left(\lfloor \frac{n}{d} \rfloor = x \right) \right)\} \subseteq \{1, 2, 3, \dots, \lfloor \sqrt{n} \rfloor\} \cup \{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{3} \rfloor, \lfloor \frac{n}{4} \rfloor, \dots, \lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \rfloor\}$

对 $m \in \{x \mid \exists d \left((2 \leq d \leq n) \wedge \left(\lfloor \frac{n}{d} \rfloor = x \right) \right)\}$ 有 $\{1, 2, 3, \dots, \lfloor \sqrt{m} \rfloor\} \cup \{\lfloor \frac{m}{2} \rfloor, \lfloor \frac{m}{3} \rfloor, \lfloor \frac{m}{4} \rfloor, \dots, \lfloor \frac{m}{\lfloor \sqrt{m} \rfloor} \rfloor\} \subseteq \{1, 2, 3, \dots, \lfloor \sqrt{n} \rfloor\} \cup \{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{3} \rfloor, \lfloor \frac{n}{4} \rfloor, \dots, \lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \rfloor\}$

因为首先 $m < n$ 于是

$$\{1, 2, 3, \dots, \lfloor \sqrt{m} \rfloor\} \subseteq \{1, 2, 3, \dots, \lfloor \sqrt{n} \rfloor\}$$

$$\lfloor \frac{n}{d} \rfloor$$

设 $m = \lfloor \frac{n}{d} \rfloor$ 有

$$\begin{equation} \left\{ \left\lfloor \frac{n}{2d} \right\rfloor, \left\lfloor \frac{n}{3d} \right\rfloor, \left\lfloor \frac{n}{4d} \right\rfloor, \dots, \left\lfloor \frac{n}{\lfloor \sqrt{m} \rfloor d} \right\rfloor \right\} \subset \left\{ \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{3} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, \dots, \left\lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \right\rfloor \right\} \end{equation}$$

所以记忆化搜索只需要求出最开始的状态 $O(\sqrt{n})$ 个，即 $\{1, 2, 3, \dots, \lfloor \sqrt{n} \rfloor\} \cup \{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{3} \rfloor, \lfloor \frac{n}{4} \rfloor, \dots, \lfloor \frac{n}{\lfloor \sqrt{n} \rfloor} \rfloor\}$

根据整数分块，每个状态统计答案的时间复杂度为 $O(\sqrt{n})$ 总时间复杂度为

$$\begin{equation} \sum_{i=1}^{\lfloor \sqrt{n} \rfloor} \left(O(\sqrt{i}) + O\left(\sqrt{\frac{n}{i}}\right) \right) = O\left(\int_{x=1}^{\sqrt{n}} \sqrt{x} + \sqrt{\frac{n}{x}} dx\right) = O\left(n^{\frac{3}{4}}\right) \end{equation}$$

考虑线性筛预处理前 k 个前缀和 $(k \geq \sqrt{n})$

总时间复杂度变为

$$\begin{equation} O(k) + \sum_{i=1}^{\lfloor \sqrt{\frac{n}{k}} \rfloor} O\left(\sqrt{\frac{n}{i}}\right) = O(k) + O\left(\int_{x=1}^{\sqrt{\frac{n}{k}}} \sqrt{\frac{n}{x}} dx\right) = O(k) + O\left(\frac{n}{\sqrt{k}}\right) \end{equation}$$

发现取 $k \sim n^{\frac{2}{3}}$ 时可以达到最佳时间复杂度 $O\left(n^{\frac{2}{3}}\right)$

另外关于记忆化搜索的答案，建议用哈希表存储。

算法练习

习题一

[洛谷p4213](#)

题意

给定正整数 n 求

$$\text{ans}_1 = \sum_{i=1}^n \varphi(i)$$

$$\text{ans}_2 = \sum_{i=1}^n \mu(i)$$

题解

取 $f = \varphi, g = I$ 则 $(f \text{ last } g) = \text{id}$ 根据杜教筛有

$$(1) S(n) = \sum_{i=1}^n id(i) - \sum_{d=2}^n I(d) S(\lfloor \frac{nd}{d} \rfloor)$$

即

$$S(n) = \frac{n(n+1)}{2} - \sum_{d=2}^n S(\lfloor \frac{nd}{d} \rfloor)$$

取 $f = \mu, g = I$ 则 $(f \ast g) = e$ 根据杜教筛有

$$(1) S(n) = \sum_{i=1}^n e(i) - \sum_{d=2}^n I(d) S(\lfloor \frac{nd}{d} \rfloor)$$

即

$$S(n) = 1 - \sum_{d=2}^n S(\lfloor \frac{nd}{d} \rfloor)$$

```

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();

```

```
while(c==' '||c=='\n'||c=='\r')c=getchar();
return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXP=5e6+5;
bool vis[MAXP];
int prime[MAXP],mu[MAXP],cnt;
LL phi[MAXP];
template <typename T1,typename T2>
struct HASH_Table{
    static const int HASH_MOD=3000017,MAXS=5e6;
    struct cell{
        T1 key;T2 val;
        int next;
    }e[MAXS];
    int head[HASH_MOD],cnt;
    void clear(){mem(head,0);cnt=0;}
    T2 insert(T1 Key,T2 Value){
        int h=Key%HASH_MOD;
        e[++cnt].key=Key,e[cnt].val=Value,e[cnt].next=head[h];
        head[h]=cnt;
        return Value;
    }
    T2 find(T1 Key){
        int h=Key%HASH_MOD;
        for(int i=head[h];i;i=e[i].next){
            if(e[i].key==Key)
                return e[i].val;
        }
        return -1;
    }
};
HASH_Table<int,int> S_Mu;
HASH_Table<int,LL> S_Phi;
void Pre(){
    vis[1]=true,mu[1]=1,phi[1]=1;
    _for(i,2,MAXP){
        if(!vis[i])mu[i]=-1,phi[i]=i-1,prime[cnt++]=i;
        for(int j=0;j<cnt&&i*prime[j]<MAXP;j++){
            vis[i*prime[j]]=true;
            if(i%prime[j])
                mu[i*prime[j]]=-mu[i],phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
```

```

        else{
            mu[i*prime[j]]=0,phi[i*prime[j]]=phi[i]*prime[j];
            break;
        }
    }
}
_for(i,2,MAXP)
mu[i]+=mu[i-1],phi[i]+=phi[i-1];
}
int S_mu(int n){
    if(n<MAXP)
        return mu[n];
    if(S_Mu.find(n)!=-1)
        return S_Mu.find(n);
    int ans=1,lef=2,rig;
    while(lef<=n){
        rig=n/(n/lef);
        ans-=(rig-lef+1)*S_mu(n/lef);
        lef=rig+1;
    }
    return S_Mu.insert(n,ans);
}
LL S_phi(int n){
    if(n<MAXP)
        return phi[n];
    if(S_Phi.find(n)!=-1)
        return S_Phi.find(n);
    LL ans=1LL*n*(n+1)/2;
    int lef=2,rig;
    while(lef<=n){
        rig=n/(n/lef);
        ans-=1LL*(rig-lef+1)*S_phi(n/lef);
        lef=rig+1;
    }
    return S_Phi.insert(n,ans);
}
int main()
{
    int t=read_int(),n;
    Pre();
    while(t--){
        n=read_int();
        space(S_phi(n));
        enter(S_mu(n));
    }
    return 0;
}

```

习题二

洛谷p3768

题意

给定 n, p 计算

$$\sum_{i=1}^n \sum_{j=1}^n ij \text{gcd}(i, j) \bmod p$$

题解

先把 gcd 转化为莫比乌斯函数，有

$$\sum_{i=1}^n \sum_{j=1}^n ij \text{gcd}(i, j) = \sum_{d=1}^n d \sum_{i=1}^n \sum_{j=1}^n ij [(i, j) = d] = \sum_{d=1}^n d^3 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} ij [(i, j) = 1] = \sum_{d=1}^n d^3 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} ij \sum_{k \mid (i, j)} \mu(k)$$

改变枚举顺序，有

$$\sum_{d=1}^n d^3 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} ij \sum_{k \mid (i, j)} \mu(k) = \sum_{d=1}^n d^3 \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} \mu(k) \sum_{k \mid i} \sum_{k \mid j} ij = \sum_{d=1}^n d^3 \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} k^2 \mu(k) \left(\sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} i \right)^2$$

设 $dk = T, S(n) = \sum_{i=1}^n i$ 将 $k = \frac{T}{d}$ 代入，有

$$\sum_{d=1}^n d^3 \sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} k^2 \mu(k) \left(\sum_{i=1}^{\lfloor \frac{n}{dk} \rfloor} i \right)^2 = \sum_{T=1}^n S\left(\frac{n}{T}\right) T^2 \sum_{d \mid T} d \mu\left(\frac{T}{d}\right) = \sum_{T=1}^n S\left(\frac{n}{T}\right) T^2 \varphi(T)$$

考虑数论分块 + 杜教筛，设 $F(n) = \sum_{i=1}^n f(i), f(n) = n^2 \varphi(n), g(n) = n^2$ 有

$$(f * g)(n) = \sum_{d \mid n} f(d) g\left(\frac{n}{d}\right) = \sum_{d \mid n} d^2 \varphi(d) \left(\frac{n}{d}\right)^2 = n^2 \sum_{d \mid n} \varphi(d) = n^3$$

根据杜教筛公式，有

$$F(n) = \sum_{i=1}^n i^3 - \sum_{d=2}^n d^2 F\left(\frac{n}{d}\right)$$

再根据 $\sum_{i=1}^n i^3 = \left(\frac{n(n+1)}{2}\right)^2, \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$ 便可以快速计算出 $F(n)$

事实上，杜教筛在计算出 $F(n)$ 的同时也计算出了所有 $F\left(\frac{n}{d}\right)$ 的值。

所以利用记忆化搜索，外层嵌套分块不影响时间复杂度，仍为 $O\left(n^{\frac{2}{3}}\right)$

```

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
int mod,inv2,inv6;
int quick_pow(LL a,LL b,int mod){
    LL t=1;

```

```
while(b){
    if(b&1)
        t=t*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return t%mod;
}

template <typename T1,typename T2>
struct HASH_Table{
    static const int HASH_MOD=3000017,MAXS=5e6;
    struct cell{
        T1 key;T2 val;
        int next;
    }e[MAXS];
    int head[HASH_MOD],cnt;
    void clear(){mem(head,0);cnt=0;}
    T2 insert(T1 Key,T2 Value){
        int h=Key%HASH_MOD;
        e[++cnt].key=Key,e[cnt].val=Value,e[cnt].next=head[h];
        head[h]=cnt;
        return Value;
    }
    T2 find(T1 Key){
        int h=Key%HASH_MOD;
        for(int i=head[h];i;i=e[i].next){
            if(e[i].key==Key)
                return e[i].val;
        }
        return -1;
    }
};

HASH_Table<LL,int> pre_2;
const int MAXP=8e6;
bool vis[MAXP];
int prime[MAXP],pre_1[MAXP],cnt;
void Pre(){
    vis[1]=true,pre_1[1]=1;
    _for(i,2,MAXP){
        if(!vis[i])pre_1[i]=i-1,prime[cnt++]=i;
        for(int j=0;j<cnt&&i*prime[j]<MAXP;j++){
            vis[i*prime[j]]=true;
            if(i%prime[j])
                pre_1[i*prime[j]]=pre_1[i]*(prime[j]-1);
            else{
                pre_1[i*prime[j]]=pre_1[i]*prime[j];
                break;
            }
        }
    }
}
```

```

    _for(i,2,MAXP)
    pre_1[i]=(1LL*i*i%mod*pre_1[i]+pre_1[i-1])%mod;
}
int Pow_1s(LL n){n%=mod;return n*(n+1)%mod*inv2%mod;}
int Pow_2s(LL n){n%=mod;return n*(n+1)%mod*(2*n+1)%mod*inv6%mod;}
int Pow_3s(LL n){n%=mod;LL t=n*(n+1)%mod*inv2%mod;return t*t%mod;}
int S(LL n){
    if(n<MAXP)
        return pre_1[n];
    if(pre_2.find(n)!=-1)
        return pre_2.find(n);
    LL ans=Pow_3s(n),lef=2,rig;
    while(lef<=n){
        rig=n/(n/lef);
        ans=(ans-1LL*(Pow_2s(rig)-Pow_2s(lef-1))*S(n/lef))%mod;
        lef=rig+1;
    }
    return pre_2.insert(n,ans);
}
int cal(LL n){
    LL ans=0,lef=1,rig,t;
    while(lef<=n){
        rig=n/(n/lef);
        t=Pow_1s(n/lef);t=t*t%mod;
        ans=(ans+t*(S(rig)-S(lef-1)))%mod;
        lef=rig+1;
    }
    return (ans+mod)%mod;
}
int main()
{
    mod=read_int();
    inv2=quick_pow(2,mod-2,mod);
    inv6=quick_pow(6,mod-2,mod);
    Pre();
    enter(cal(read_LL()));
    return 0;
}

```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E6%95%B0%E8%AE%BA_3&rev=1594369355

Last update: 2020/07/10 16:22