算法竞赛小专题系列--

杜教筛--以及积性函数的前世今生

罗勇军 2020.2.9

本系列是这本算法教材的扩展资料：《算法竞赛入门到进阶》.罗勇军、郭卫斌. 清华大学出版社
本文 web 地址：https://blog.csdn.net/weixin_43914593
PDF 下载地址：https://github.com/luoyongjun999/code 其中的补充资料
如有建议，请联系：（1）QQ 群，567554289；（2）作者 QQ，15512356

战疫期间，清华大学出版社免费开放了电子书。《算法竞赛入门到进阶》的电子书免费阅读：
https://lib-nuanxin.wqxuetang.com/#/Book/3209109

目录

本文详细讲解了与杜教筛有关的各种知识，包括积性函数、欧拉函数、整除分块、狄利克雷卷积、莫比乌斯反演、杜教筛等。以及它们的：理论知识、典型例题、关键代码、练习题。

本文的目标是：让**一名数论小白，也能最终看懂和应用杜教筛**。

0 杜教筛简介

0.1 杜教筛的核心内容

杜教筛用途：在低于线性时间里，高效率求**一些**积性函数的前缀和。

杜教筛算法 = 整除分块 + 狄利克雷卷积 + 线性筛。

杜教筛公式：$g(1)S(n)=\sum_{i=1}^{n}h(i)-\sum_{i=2}^{n}g(i)S(\left\lfloor\frac{n}{i}\right\rfloor)$

0.2 杜教筛之起源

“杜教筛”，一个亲切的算法名字，从 2016 年后流行于中国的 OI 圈。

杜教，即信息学竞赛的大佬杜瑜皓。本文作者好奇“杜教筛”这个名称的来源，在 QQ 上联系到他，他说，虽然算法的原理不是他的发明，但确实是他最早应用在 OI 竞赛中。至于为什么被称为“杜教筛”并广为传播，可能是一个迷了。

这个算法的原始出处，据 skywalkert（唐靖哲，唐老师）的博客说[①]，“这种黑科技大概起源于 Project Euler 这个网站[②]，由 xudyh（杜瑜皓）引入中国的 OI、ACM 界，目前出现了一些 OI 模拟题、OJ 月赛题、ACM 赛题是需要这种技巧在低于线性时间的复杂度下解决一类积性函数的前缀和问题。”

Project Euler 网站也是一个 OJ，题库里面有很多高难度的数学题。不过这个 OJ 并不需要提交代码，只需要提交答案即可。这个网站被推崇的一大原因是：如果提交的答案正确，就有权限看别人对这一题的解题方法，以及上传的代码。

经典的杜教筛题目，例如洛谷 P4213，求数论函数的前缀和。

洛谷 P4213 https://www.luogu.com.cn/problem/P4213
给定一个正整数 n，$n\leqslant 2^{31}-1$，求：

$$ans1=\sum_{i=1}^{n}\phi(i)\ ,\quad ans2=\sum_{i=1}^{n}\mu(i)$$

题目中的 $\phi(i)$ 是欧拉函数，$\mu(i)$ 是莫比乌斯函数。欧拉函数、莫比乌斯函数在算法竞赛中很常见，它们都是积性函数。题目求这 2 个积性函数的前缀和，由于给的 n 很大，显然不能直接用暴力的、复杂度 $O(n)$ 的线性方法算，需要用杜教筛，复杂度是 $O(n^{\frac{2}{3}})$。

关于“杜教筛”的理论知识介绍，影响较大的有：

（1）2016 年信息学奥林匹克中国国家队候选队员论文集，“积性函数求和的几种方法 任之洲”，其中提到杜教筛：“这个算法在国内信息学竞赛中一般称为杜教筛，可以高效地完成一些常见数论函数的计算。”

（2）2016 年，skywalkert（唐老师）的博客。

0.3 本文的结构

由于杜教筛涉及的知识比较多，不容易讲清楚。所以本文将循循善诱地完成这一困难的任务，本文的顺序是：

（1）积性函数的定义和一些性质。积性函数的常见计算。

（2）以欧拉函数为例，讲解积性函数的计算。欧拉函数有什么用、如何求、它的求解和积性函数有什么关系、为什么用到杜教筛。

（3）整数分块。

（4）狄利克雷卷积。

---

①https://blog.csdn.net/skywalkert/article/details/50500009

② https://projecteuler.net/archives

（5）莫比乌斯函数和莫比乌斯反演。

（6）杜教筛。杜教筛公式、复杂度证明、欧拉函数和莫比乌斯函数的杜教筛实现。

1 积性函数

如果读者没有深入学过数论，第一次看到这些词语：积性函数、欧拉函数、狄利克雷卷积、莫比乌斯函数、莫比乌斯反演、杜教筛，是不是感到高深莫测、头皮发怵？

即使读者学过一些数论，但是还没有系统读过初等数论的书，那么在阅读本文之前，最好也读一本，比如《初等数论及其应用》Kenneth H.Rosen 著，夏鸿刚译，机械工业出版社。这本书很易读，非常适合学计算机的人阅读：（1）概览并证明了初等数论的理论知识；（2）理论知识的各种应用，可以直接用在算法题目里；（3）大量例题和习题；（4）与计算机算法编程有很多结合。花两天时间通读很有益处。

本文所有的数学定义都引用自这本书。

1.1 积性函数定义

定义[①]：

（1）定义在所有正整数上的函数称为算术函数（或数论函数）。

（2）如果算术函数 f 对任意两个互质（互素）的正整数 p 和 q，均有 $f(pq)=f(p)f(q)$，称为积性函数（multiplicative functions，或译为乘性函数）[②]。

（3）如果对任意两个正整数 p 和 q，均有 $f(pq)=f(p)f(q)$，称为完全积性函数。

性质：

【定理 1.1】积性函数的和函数也是积性函数。如果 f 是积性函数，那么 f 的和函数 $F(n)=\sum_{d|n} f(d)$ 也是积性函数。

其中 d|n 表示 d 整除 n。

和函数在杜教筛中起到了关键的作用。

1.2 积性函数的基本问题

（1）计算积性函数 f 的第 n 项 f(n)。

（2）计算积性函数 f 在 1～n 范围内所有项：f(1)、f(2)、...、f(n)。

（3）计算积性函数 f 前 n 项的和：$\sum_{i=1}^{n} f(i)$，即前缀和。这就是杜教筛的目标。

后面以欧拉函数和莫比乌斯函数为例，解答了这几个问题。

1.3 常见积性函数

下面的积性函数，在狄利克雷卷积中常常用到。

$I(n)$：恒等函数，$I(n)=1$

$id(n)$：单位函数，$id(n)=n$

$I_k(n)$：幂函数，$I_k(n)=n^k$

$\varepsilon(n)$：元函数，$\varepsilon(n)=\begin{cases}1, & n=1\\ 0, & n>1\end{cases}$

$\sigma(n)$：因子和函数，$\sigma(n)=\sum_{d|n} d$

$d(n)$：约数个数，$d(n)=\sum_{d|n} 1$

2 欧拉函数

---

① 《初等数论及其应用》第 7 章乘性函数。

② 有乘性函数，自然也有加性函数。参考《初等数论及其应用》182 页。

为了彻底了解积性函数的运算，本节以欧拉函数为例进行讲解，这是一个经典的积性函数。并用这个例子引导出为什么要使用杜教筛。

2.1 欧拉函数的定义和性质

定义[①]：设 n 是一个正整数，欧拉函数 $\phi(n)$ 定义为不超过 n 且与 n 互质的正整数的个数。

定义用公式表示： $\phi(n)=\sum_{i=1}^{n}[\gcd(i,n)=1]$

例如：

n = 4 时，1、3 这 2 个数与 4 互质， $\phi(4)=2$ ；

n = 9 时，1、2、4、5、7、8 这 6 个数与 9 互质， $\phi(9)=6$ 。

【定理 2.1】[②]：设 p 和 q 是互质的正整数，那么 $\phi(pq)=\phi(p)\phi(q)$ 。

这个定理说明欧拉函数是一个积性函数。有以下推理：

若 $n=p_1^{a_1}\times p_2^{a_2}...\times p_s^{a_s}$ ，其中 p_1、p_2、...、p_s 互质，a_1、a_2、...、a_s 是它们的幂，则 $\phi(n)=\phi(p_1^{a_1})\times\phi(p_2^{a_2})\times...\times\phi(p_s^{a_s})$ 。

如果 p_1、p_2、...、p_s 互质，那么 p_1^{a1}、p_2^{a2}、...、p_s^{as} 也是互质的。

以 n = 84 为例：

$84=2^2\times3\times7$

$\phi(84)=\phi(2^2\times3\times7)=\phi(2^2)\times\phi(3)\times\phi(7)$

【定理 2.2】设 n 为正整数，那么： $n=\sum_{d|n}\phi(d)$

其中 d|n 表示 d 整除 n，上式对 n 的正因子求和。 例如 n=12，那么 d 是 1、2、3、4、6、12，有： $12=\phi(1)+\phi(2)+\phi(3)+\phi(4)+\phi(6)+\phi(12)$

这个定理说明了 n 与 $\phi(n)$ 的关系：n 的正因子（包括 1 和自身）的欧拉函数之和等于 n。

有很多方法可以证明。这里给出一种直接易懂的证明。

证明：分数 $\frac{1}{n},\frac{2}{n},\cdots\frac{n}{n}$ ，共有 n 个，且互不相等。化简这些分数，得到新的 n 个分数，它们的分母和分子互质，形如 $\frac{a}{d}$ ， $d|n$ 且 a 与 d 互质。在所有 n 个分数中，分母为 d 的分数，数量是 $\phi(d)$ 。所有不同分母的分数，其总数是 $\sum_{d|n}\phi(d)$ ，所以 $n=\sum_{d|n}\phi(d)$ ，得证。

定理 2.2 是欧拉函数特别重要的一个性质，这一性质被用到了杜教筛中。

2.2 求欧拉函数的通解公式

欧拉函数有什么用呢？利用它可以推出欧拉定理[③]，欧拉定理可用于：求大数的模、求解线性同余方程等，在密码学中有重要应用。所以求 $\phi(n)$ 是一个常见的计算。

从定理 2.1 可以推出定理 2.3，定理 2.3 是计算 $\phi(n)$ 的通解公式。

【定理 2.3】设： $n=p_1^{a_1}\times p_2^{a_2}...\times p_s^{a_s}$ 为正整数 n 的质幂因子分解，那么：

$$\phi(n)=n(1-\frac{1}{p_1})(1-\frac{1}{p_2})...(1-\frac{1}{p_k})=n\prod_{i=1}^{k}(1-\frac{1}{p_i})$$

上述公式，有两个特殊情况：

（1）若 n 是质数， $\phi(n)=n-1$ 。这一点很容易理解，请读者思考。

（2）若 $n=p^k$ ，p 是质数，有：

---

①《初等数论及其应用》172 页，欧拉定理。

②有关欧拉函数的所有证明，见《初等数论及其应用》176-180 页。

③欧拉定理：设 m 是一个正整数，*a* 是一个整数且*(a, m)=1*，那么 $a^{\phi(m)}\equiv1(\mathrm{mod}\ m)$ 。

$$\phi(n)=\phi(p^k)=p^k-p^{k-1}=p^{k-1}(p-1)=p^{k-1}\phi(p)$$

这两种情况将用于 2.3.2 节编程求欧拉函数。

所以，欧拉函数 $\phi(n)$ 的求解，归结到了分解质因子这个问题。分解质因子[①]有一个古老的方法，试除法：求 n 的质因子时，逐个检查从 2 到 $\sqrt{n}$ 的所有质数，如果它能整除 n，就是一个因子。试除法的复杂度是 $O(\sqrt{n})$，效率很低。在密码学中，有高达百位以上的十进制数分解质因子，因此发明了很多高效率的方法。不过，在算法竞赛中，数据规模不大，所以一般就用试除法。

下面是求欧拉函数的代码。先用试除法分解质因子，再用公式 $\phi(n)=n\times\prod_{i=1}^{k}(1-\frac{1}{p_i})$ 求得 $\phi(n)$。总复杂度仍然是 $O(\sqrt{n})$。

求单个欧拉函数的代码

```
int euler(int n){
    int ans = n;
    for(int p = 2; p*p <= n; ++ p){ //试除法：检查从 2 到 sqrt(n)的每个数
        if(n%p == 0){               //能整除，p 是一个因子，而且是质因子，请思考
            ans = ans/p*(p-1);      //求欧拉函数的通式
            while(n%p == 0)         //去掉这个因子的幂，并使得下一个 p 是质因子
                n /= p;             //减小了 n
        }
    }
    if(n != 1)                      //情况(1)：n 是一个质数，没有执行上面的分解
        ans = ans/n*(n-1);
    return ans;
}
```

2.3 线性筛（欧拉筛）求 1～n 内所有的欧拉函数

现在要求 1～n 范围内所有的欧拉函数，而不是只求一个欧拉函数。前面求一个欧拉函数的复杂度是 $O(\sqrt{n})$，如果一个一个地求 1～n 范围内所有的欧拉函数，那么 n 个的总复杂度是 $O(n\sqrt{n})$。效率太低。

这个过程可以优化：

（1）利用积性函数的性质 $\phi(p_1p_2)=\phi(p_1)\phi(p_2)$，求得前面的 $\phi(p_1)$ 和 $\phi(p_2)$ 后，可以递推出后面的 $\phi(p_1p_2)$；

（2）求 1～n 范围内的质数，用于上一步骤的递推。

编程时，可以这样操作：

（1）求得 1～n 内每个数的最小质因子。所谓最小质因子，例如 $84=2^2\times3\times7$，84 的最小质因子是 2。这步操作可以用欧拉筛，复杂度是 O(n)的，不可能更快了。

（2）在第（1）步基础上，递推求 1～n 内每个数的欧拉函数。这一步的复杂度也是 O(n)的。

两者相加，总复杂度是 O(n)的。

2.3.1 欧拉筛

欧拉筛是一种线性筛，也就是说，它能在 O(n)的线性时间里求得 1～n 内所有的质数。

欧拉筛是对埃氏筛法的改进。埃氏筛法的原理是：每个合数都是多个质数的积；那么从最小的质数 2 开始，用每一个质数去筛比它大的数，就能筛掉合数。埃氏筛法低效的原因是，一个合数会被它的多个质因子重复筛。请读者自己详细了解埃氏筛法。

欧拉筛法的原理是：一个合数肯定有一个最小质因子；让每个合数只被它的最小质因子筛

[①]试除法很低效，有很多快速分解质因子的方法，参考《初等数论及其应用》93 页。

选一次，以达到不重复筛的目的。

具体操作步骤：

（1）逐一检查 2～n 的所有数。第一个检查的是 2，它是第一个质数。

（2）当检查到第 i 个数时，利用已经求得的质数去筛掉对应的合数 x，而且是用 x 的最小质因子去筛。

下面是代码。代码很短很精妙。

欧拉筛求素数

```
int prime[MAXN];            //保存质数
int vis[MAXN];              //记录是否被筛
int euler_sieve(int n){     //欧拉筛。返回质数的个数。
    int cnt = 0;            //记录质数个数
    memset(vis,0,sizeof(vis));
    memset(prime,0,sizeof(prime));
    for(int i=2;i<=n;i++){           //检查每个数，筛去其中的合数
        if(!vis[i]) prime[cnt++]=i;  //如果没有筛过，是质数，记录。第一个质数是 2
        for(int j=0; j<cnt; j++){    //用已经得到的质数去筛后面的数
            if(i*prime[j] >n)  break;  //只筛小于等于 n 的数
            vis[i*prime[j]]=1;         //关键 1。用 x 的最小质因数筛去 x
            if(i%prime[j]==0)  break;  //关键 2。如果不是这个数的最小质因子，结束
        }
    }
    return cnt;                        //返回小于等于 n 的素数的个数
}
```

代码中难懂的是后面 2 个关键行。在“关键 1”， 其中的 prime[j]是最小质因子，vis[i*prime[j]]=1;表示用最小质因子筛去了它的倍数。在“关键 2”，及时跳出，避免重复筛。

下面用图解来说明，请读者对照代码和图解自己理解。特别注意图 4，它是“关键 2”。

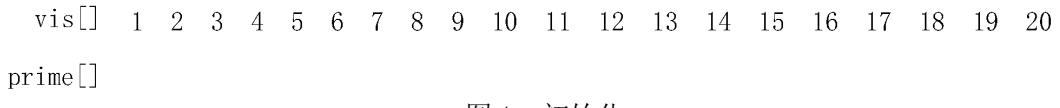

图 1. 初始化

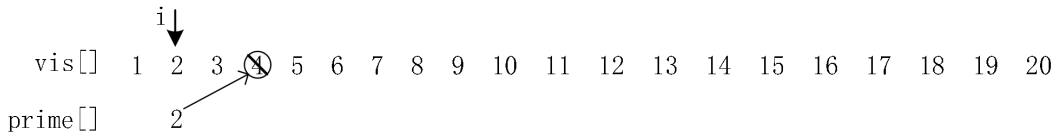

图 2. i=2，用质数 2 筛去 4（**2 是 4 的最小质因子**）

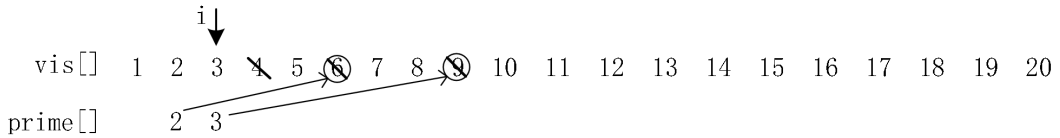

图 3. i=3，用质数 2 筛去 6，用质数 3 筛去 9

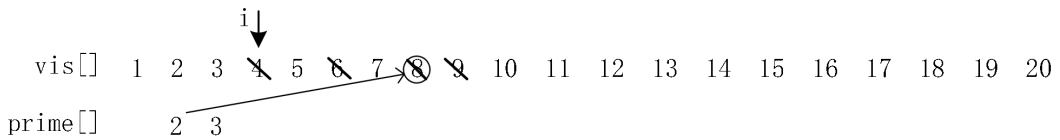

图 4. i=4，用质数 2 筛去 8，**注意质数 3 没有用到，循环 j 被跳出了**

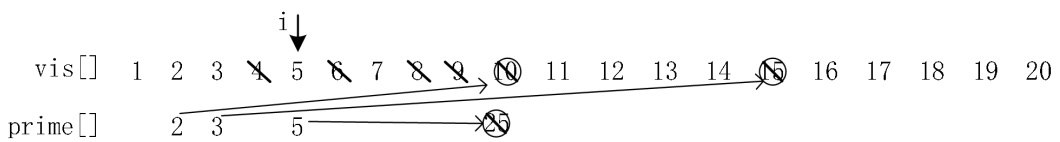

图 5. i=5，用质数 2 筛去 10，用 3 筛去 15，用 5 筛去 25

可以发现，每个数 x 只被筛了一次，而且是被 x 的最小质因子筛去的。这说明筛法是线性的，复杂度 O(n)。

现在用欧拉筛求 1～n 内每个数的最小质因子。只要简单修改上述程序，直接用 vis[MAXN] 记录最小质因子就行。代码修改如下，修改了带注释的后 2 行。

欧拉筛求质数+最小质因子

```
int prime[MAXN];      //记录质数
int vis[MAXN];      //记录最小质因子
int euler_sieve(int n){
    int cnt=0;
    memset(vis,0,sizeof(vis));
    memset(prime,0,sizeof(prime));
    for(int i=2;i<=n;i++){
        if(!vis[i]){ vis[i]=i; prime[cnt++]=i;}    //vis[]记录 x 的最小质因子
        for(int j=0; j<cnt; j++){
            if(i*prime[j] >n)  break;
            vis[i*prime[j]]=prime[j];              //vis[]记录 x 的最小质因子
            if(i%prime[j]==0)
                break;
        }
    }
    return cnt;
}
```

2.3.2 求 1～n 内所有的欧拉函数

下面用一个模板题给出模板代码，它就是“欧拉筛+欧拉函数公式”。

例题：洛谷 P2158 仪仗队 https://www.luogu.com.cn/problem/P2158

这是欧拉函数的模板题。get_phi()是计算欧拉函数的模板，其中判断了 3 种情况，细节见注释。

情况（1）：若 n 是质数，$\phi(n)=n-1$。

情况（2）：若 $n=p^k$，p 是质数，有：$\phi(n)=p^{k-1}\phi(p)$

情况（3）：若 $n=p_1p_2$，用 $\phi(n)=\phi(p_1)\phi(p_2)$ 递推。

洛谷 P2158 代码

```
#include<bits/stdc++.h>
using namespace std;

const int maxn = 50000;
int vis[maxn];   //记录是否被筛；或者用于记录最小质因子
int prime[maxn]; //记录质数
int phi[maxn];   //记录欧拉函数
int sum[maxn];   //计算欧拉函数的和

void get_phi(){  //模板：求 1～maxn 范围内的欧拉函数
    phi[1]=1;
    int cnt=0;
    for(int i=2;i<maxn;i++) {
        if(!vis[i]) {
            vis[i]=i; //vis[i]=1; //二选一：前者记录最小质因子，后者记录是否被筛
```

```
            prime[cnt++]=i;        //记录质数
            phi[i]=i-1;            //情况（1）：i 是质数，它欧拉函数值=i-1
        }
        for(int j=0;j<cnt;j++) {
            if(i*prime[j] > maxn)  break;
            vis[i*prime[j]] = prime[j]; //vis[i*prime[j]]=1;
                                   //二选一：前者记录最小质因子，后者记录是否被筛
            if(i%prime[j]==0){          //prime[j]是最小质因子
                phi[i*prime[j]] = phi[i]*prime[j];
                                   //情况（2）：i 是 prime[j]的 k 次方
                break;
            }
            phi[i*prime[j]] = phi[i]*phi[prime[j]];
                                  //情况（3）：i 和 prime[j]互质，递推出 i*prime[j]
        }
    }
}

int main(){
    get_phi();        //计算所有的欧拉函数
    sum[1]=1;
    for(int i=2;i<=maxn;i++)   //打表计算欧拉函数的和
        sum[i]=sum[i-1]+phi[i];
    int n;   scanf("%d",&n);
    if(n==1) printf("0\n");
    else     printf("%d\n",2*sum[n-1]+1);
    return 0;
}
```

本节的例子是用欧拉筛求欧拉函数，读者可以认识到：**积性函数都可以用欧拉筛来求。**后面还有一个例子：用欧拉筛求解积性函数莫比乌斯函数。

2.4 欧拉函数前缀和

在欧拉函数问题的最后，回到杜教筛的模板题，求：$\sum_{i=1}^{n}\varphi(i)$。

简单的方法，是用 2.3.2 节计算出每个欧拉函数，再求和，复杂度是 O(n)。

更快的方法，就是本篇的主题“杜教筛”，复杂度是$O(n^{\frac{2}{3}})$。

“杜教筛”需要整除分块、狄利克雷卷积等前置知识。

3 整除分块（数论分块）

整除分块是杜教筛算法的基本思路。

整除分块是为了解决一个整除的求和问题：$\sum_{i=1}^{n}\left\lfloor\frac{n}{i}\right\rfloor$

其中$\left\lfloor\frac{n}{i}\right\rfloor$表示 n 除以 i，并向下取整[①]。n 是给定的一个整数。

如果直接暴力计算，复杂度是 O(n)的，若 n 很大会超时。但是，用整除分块算法来求解，复杂度是 O($\sqrt{n}$)的。

[①]除法运算，向上取整，英文 Ceiling，用符号「 ⌉ 表示；向下取整，英文 Floor，用符号 ⌊ 」表示。

下面以 n = 8 为例，列出每个情况：

i	1	2	3	4	5	6	7	8
8/i	8	4	2	2	1	1	1	1

对第 2 行求和，结果是：$\sum_{i=1}^{8}\left\lfloor\frac{8}{i}\right\rfloor=20$。

观察上面第 2 行 8/i 的值，发现是逐步变小的，并且很多相等，这是整除操作的规律。例如：

8/1 = 8

8/2 = 4

8/3 = 8/4 = 2

8/5 = 8/6 = 8/7 = 8/8 = 1

为了对这些整除的结果进行快速求和，自然能想到，把它们分成一块块的，其中每一块的 8/i 相同，把这一块一起算，就快多了。

设每一块的左右区间是$[l,r]$，上面的例子可以分成 4 块：[1,1]、[2,2]、[3,4]、[5,8]。

每一块内部的求和计算，只要用数量乘以值就行了，例如[5,8]区间的整除和：(8-5+1)*1=4。这个计算的复杂度是 O(1)的。

最后还有两个问题：

（1）把$\left\lfloor\frac{n}{i}\right\rfloor$按相同值分块，一共需要分成几块？或者说，$\left\lfloor\frac{n}{i}\right\rfloor$有几种取值？这决定了算法的复杂度。

【答】分块少于$2\sqrt{n}$种，证明如下：

（a）$i\le\sqrt{n}$时，$\frac{n}{i}$的值有：$\{\frac{n}{1},\frac{n}{2},\frac{n}{3},...,\frac{n}{\sqrt{n}}\}$，$\frac{n}{i}\ge\sqrt{n}$，共$\sqrt{n}$个，此时$\left\lfloor\frac{n}{i}\right\rfloor$有$\sqrt{n}$种取值；

（b）$i>\sqrt{n}$时，有$\frac{n}{i}<\sqrt{n}$，此时$\left\lfloor\frac{n}{i}\right\rfloor$也有$\sqrt{n}$种取值。

两者相加，共$2\sqrt{n}$种。所以，整除分块的数量是$O(\sqrt{n})$的。

（2）如何计算？或者说，给定每个块的第一个数 l，能推出这个块的最后一个数 r 吗？

【答】可以证明：$r=n/(n/l)$。证明略[①]。

下面是标程。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    long long n,ans=0;
    cin >> n;
    for(long long l=1,r;l<=n;l=r+1){
        r = n/(n/l);            //计算 r，让分块右移
        ans += (r-l+1)*(n/l);   //求和
        cout << l <<""<< r <<": "<< n/r << endl;  //打印分块
    }
    cout << ans;                //打印和
}
```

整除分块习题：洛谷 P1829、洛谷 P2261、洛谷 P3935。

4 狄利克雷卷积

狄利克雷卷积 Dirichlet Convolution，是计算求和问题的有用工具。狄利克雷卷积是杜教筛用到的核心技术之一。

定义：

[①]证明细节见：https://blog.csdn.net/qq_41021816/article/details/84842956

设 f，g 是算术函数，记 f 和 g 的狄利克雷卷积是 $f*g$，定义为[①]：

$$(f*g)(n)=\sum_{d|n}f(d)g(\frac{n}{d})$$

它是一个求和公式，其中 d|n 表示 d 整除 n，卷积是对正因子求和。

这个卷积是什么含义呢？现在给出一个特殊的例子。定义恒等函数 $I(n)=n$、常数函数 $1(n)=1$，它们的卷积是：

$$(I*1)(n)=\sum_{d|n}I(d)1(\frac{n}{d})=\sum_{d|n}d\cdot 1=\sum_{d|n}d=\sigma(n)$$

把它记为：$I*1=\sigma$

$\sigma(n)$ 是“因子和函数”的符号。例如 n=12，那么 $\sigma(n)=1+2+3+4+6+12=28$。$\sigma(n)$ 是积性函数。

狄利克雷卷积的性质：

（1）交换律：$f*g=g*f$

（2）结合律：$(f*g)*h=f*(g*h)$

（3）分配律：$f*(g+h)=(f*g)+(f*h)$

（4）元函数：$\varepsilon(n)=\left\lfloor\frac{1}{n}\right\rfloor=\begin{cases}1 & n=1\\ 0 & n>1\end{cases}$。对任何 f，有 $e*f=f*e=f$

（5）两个积性函数的狄利克雷卷积仍然是积性函数。

有一些积性函数算术函数在狄利克雷卷积中常常用到，1.3 节已经给出，这里再次列出：

恒等函数：$I(n)=1$

单位函数：$id(n)=n$

幂函数：$I_k(n)=n^k$

元函数：$\varepsilon(n)=\begin{cases}1 & n=1\\ 0 & n>1\end{cases}$

因子和函数：$\sigma(n)=\sum_{d|n}d$

约数个数：$d(n)=\sum_{d|n}1$

5 莫比乌斯函数和莫比乌斯反演

如果读者困惑莫比乌斯函数有什么用，可以先看下一节“5.2 莫比乌斯函数的由来”。

5.1 莫比乌斯函数的定义和性质

莫比乌斯函数以数学家 Möbius 的名字命名[②]。

莫比乌斯函数是狄利克雷卷积的重要工具，它也是数论和组合数学的重要的积性函数。

莫比乌斯函数 $\mu(n)$ 定义为[③]：

$$\mu(n)=\begin{cases}1 & \text{如果n}=1\\ (-1)^r & \text{如果n}=p_1p_2\ldots p_r\text{，其中}p_i\text{是不同的质数}\\ 0 & \text{其他情况}\end{cases}$$

一些例子：

$\mu(1)=1$，$\mu(2)=-1$，$\mu(3)=-1$，$\mu(4)=\mu(2^2)=0$，$\mu(5)=-1$，$\mu(6)=\mu(2\times3)=1$，$\mu(7)=-1$，$\mu(8)=\mu(2^3)=0$；$\mu(330)=\mu(2\times3\times5\times11)=(-1)^4=1$，$\mu(660)=\mu(2^2\times3\times5\times11)=0$。

莫比乌斯函数 $\mu(n)$ 是积性函数。它有一个和欧拉函数的 $n=\sum_{d|n}\phi(d)$ 类似的定理：

---

①参考：https://brilliant.org/wiki/dirichlet-convolution/

②https://brilliant.org/wiki/mobius-function/

③《初等数论及其应用》200 页。

【定理 5.1】莫比乌斯函数的和函数在整数 n 处的值 $F(n)=\sum_{d|n}\mu(d)$，满足：

$$\sum_{d|n}\mu(d)=\begin{cases}1 & n=1\\ 0 & n>1\end{cases}$$

证明：

（1）n=1 时，显然有 $F(1)=\sum_{d|n}\mu(1)=1$。

（2）n>1 时。根据积性函数的定义，有 $F(n)=F(p_1^{a_1})F(p_2^{a_2})...F(p_t^{a_t})$，其中 $n=p_1^{a_1}p_2^{a_2}...p_t^{a_t}$ 是质因子分解。如果能证明 $F(p^k)=0$ 即有 $F(n)=0$。因为当 i≥2 时，$\mu(p^i)=0$，有：

$$F(p^k)=\sum_{d|p^k}\mu(d)=\mu(1)+\mu(p)+\mu(p^2)+...+\mu(p^k)=1+(-1)+0+...+0=0$$

下面是莫比乌斯函数线性筛代码，求 1～n 内的莫比乌斯函数，和欧拉函数线性筛的代码几乎一样，此处不做解释。

```
bool vis[maxn];
int prime[maxn];
int Mob[maxn];
void Mobius_sieve(){
     int cnt = 0;
     vis[1] = 1;
     Mob[1] = 1;
     for(int i = 2; i <= maxn; i++){
        if(!vis[i])
            prime[cnt++] = i, Mob[i] = - 1;
        for(int j = 0; j < cnt && 1LL * prime[j] * i <= maxn; j++){
            vis[prime[j] * i] = 1;
            Mob[i * prime[j]] = (i % prime[j] ? -Mob[i]: 0);
            if(i % prime[j] == 0)
                break;
        }
    }
}
```

5.2 莫比乌斯函数的由来

看到莫比乌斯函数的定义，读者是不是感到奇怪？它是怎么来的？有什么用？

本节内容，引用《初等数论及其应用》199 页，它给出了莫比乌斯函数的来龙去脉。

设 f 为算术函数，f 的和函数 F 的值为 $F(n)=\sum_{d|n}f(d)$，显然，F 由 f 的值决定。这种关系可以反过来吗？也就是说，是否存在一种用 F 求出 f 的值的简便方法？本节给出这样的公式，看什么样的公式可行。

若 f 是算术函数，F 是它的和函数 $F(n)=\sum_{d|n}f(d)$，按定义展开 F(n)，n=1,2,...,8，有：

F(1) = f(1)
F(2) = f(1) + f(2)
F(3) = f(1) + f(3)
F(4) = f(1) + f(2) + f(4)
F(5) = f(1) + f(5)
F(6) = f(1) + f(2) + f(3) + f(6)
F(7) = f(1) + f(7)
F(8) = f(1) + f(2) + f(4) + f(8)

根据上面方程，可以解得：

f(1) = F(1)
f(2) = F(2) - F(1)

f(3) = F(3) - F(1)
f(4) = F(4) - F(2)
f(5) = F(5) - F(1)
f(6) = F(6) - F(3) - F(2) + F(1)
f(7) = F(7) - F(1)
f(8) = F(8) - F(4)

注意到 f(n)等于形式为 $\pm F(\frac{n}{d})$ 的一些项之和，其中 d|n。从这一结果中，可能有这样一个等式，形式是：

$$f(n)=\sum_{d|n}\mu(d)F(\frac{n}{d})$$

其中 μ 是算术函数。如果等式成立，计算得到：$\mu(1)=1$，$\mu(2)=-1$，$\mu(3)=-1$，$\mu(4)=0$，$\mu(5)=-1$，$\mu(6)=1$，$\mu(7)=-1$，$\mu(8)=0$。（读者已经发现和前一节莫比乌斯函数的值一样。）

又 $F(p)=f(1)+f(p)$，得 $f(p)=F(p)-F(1)$，其中 p 是质数。则 $\mu(p)=-1$。

又因为 $F(p^2)=f(1)+f(p)+f(p^2)$

有：$f(p^2)=F(p^2)-(F(p)-F(1))-F(1)=F(p^2)-F(p)$

这要求对任意质数 p，有 $\mu(p^2)=0$。类似的推理得出对任意质数 p 及整数 k>1，有 $\mu(p^k)=0$。如果猜想 μ 是积性函数，则 μ 的值就由质数幂处的值决定。

这就是莫比乌斯函数的由来。

5.3 莫比乌斯反演

【定理 5.2】莫比乌斯反演(Möbius inversion)公式。若 f 是算术函数，F 为 f 的和函数，对任意正整数 n，满足 $F(n)=\sum_{d|n}f(d)$，则对任意正整数 n，有 $f(n)=\sum_{d|n}\mu(d)F(\frac{n}{d})$。

从定理 5.2 可以推出下面的定理 5.3。

【定理 5.3】设 f 是算术函数，它的和函数为 $F(n)=\sum_{d|n}f(d)$，那么如果 F 是积性函数，则 f 也是积性函数。

莫比乌斯反演，不要求 f 是积性函数。

6 杜教筛

6.1 杜教筛公式

6.1.1 杜教筛公式的推导

杜教筛要解决的是这一类问题：设 $f(n)$是一个数论函数，计算 $S(n)=\sum_{i=1}^{n}f(i)$。

由于 n 很大，要求以低于 O(n)的复杂度求解，所以用前面提到的线性筛是不够的。如果能用到整除分块，每一块的值相等一起算，就加快了速度，也就是构造形如 $\sum_{i=1}^{n}\left\lfloor\frac{n}{i}\right\rfloor$ 的整除分块。

杜教筛的思路，简单地说，是把 $f(n)$构造成能利用整除分块的新的函数，这个构造用到了卷积。

记 $S(n)=\sum_{i=1}^{n}f(i)$。根据 $f(n)$的性质，构造一个 S(n)关于 $S(\left\lfloor\frac{n}{i}\right\rfloor)$ 的递推式，下面是构造方法[①]。

构造 2 个积性函数 h 和 g，满足卷积：$h=g*f$。

① http://fzl123.top/archives/898

根据卷积的定义，有 $h(i)=\sum_{d|i} g(d)f(\frac{i}{d})$，求和：

$$\begin{aligned}\sum_{i=1}^{n} h(i) &= \sum_{i=1}^{n}\sum_{d|i}^{i} g(d)f(\frac{i}{d}) \\ &= \sum_{d=1}^{n} g(d)\sum_{d|i}^{n} f(\frac{i}{d}) \\ &= \sum_{d=1}^{n} g(d)\sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\ &= \sum_{d=1}^{n} g(d)S(\lfloor \frac{n}{d} \rfloor)\end{aligned}$$

注意其中第 2 行变换 $\sum_{i=1}^{n}\sum_{d|i}^{i} g(d)f(\frac{i}{d}) = \sum_{d=1}^{n} g(d)\sum_{d|i}^{n} f(\frac{i}{d})$，网上昵称“杜教筛变换”[①]。

最后一步得到：$\sum_{i=1}^{n} h(i) = \sum_{d=1}^{n} g(d)S(\lfloor \frac{n}{d} \rfloor)$

为计算 S(n)，把式子右边的第一项提出来：

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

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

式中的 i 和 d 无关，为方便看，改写为：

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

这就是杜教筛公式。

6.1.2 杜教筛算法和复杂度

一个积性函数求和问题，如果分析得到了杜教筛公式，工作已经完成了一大半，剩下的是编程实现。

公式 $g(1)S(n) = \sum_{i=1}^{n} h(i) - \sum_{i=2}^{n} g(i)S(\lfloor \frac{n}{i} \rfloor)$ 是 S(n)的递归形式，编程时可以递归实现；递归的时候加上记忆化，让每个 S[i]只需要算一次。

其中的 h(i)和 g(i)，如果构造得好，就容易计算，见后面欧拉函数和莫比乌斯函数的例子。为方便分析计算复杂度，下面略去 h(i)和 g(i)，分析简化后的式子 $S(n) = \sum_{i=2}^{n} S(\lfloor \frac{n}{i} \rfloor)$ 的复杂度。

设 S(n)的复杂度是 T(n)。

递归展开第一层：$S(n) = \sum_{i=2}^{n} S(\lfloor \frac{n}{i} \rfloor) = S(\lfloor \frac{n}{2} \rfloor) + S(\lfloor \frac{n}{3} \rfloor) + \ldots + S(\lfloor \frac{n}{n} \rfloor)$

根据整除分块的原理，右边共有 $O(\sqrt{n})$ 种 $S(\lfloor \frac{n}{i} \rfloor)$。另外，把它们加起来的时候，还有 $O(\sqrt{n})$ 次合并。总时间是 $T(n) = O(\sqrt{n}) + O(\sum_{i=2}^{\sqrt{n}} T(\frac{n}{i}))$。

---

① “杜教筛变换”：https://blog.csdn.net/myjs999/article/details/78906549

再展开一层：$T(\frac{n}{i})=O(\sqrt{\frac{n}{i}})+O(\sum_{k=2}^{\sqrt{\frac{n}{i}}}T(\frac{\frac{n}{i}}{k}))=O(\sqrt{\frac{n}{i}})$，中间第 2 项是高阶小量，把它略去。

合起来：

$$T(n)=O(\sqrt{n})+O(\sum_{i=2}^{\sqrt{n}}T(\frac{n}{i}))$$

$$=O(\sqrt{n})+O(\sum_{i=2}^{\sqrt{n}}(\sqrt{\frac{n}{i}})$$

后一项最大，只考虑它就够了：

$$T(n)=\sum_{i=2}^{\sqrt{n}}O(\sqrt{\frac{n}{i}})\approx O(\int_0^{\sqrt{n}}\sqrt{\frac{n}{x}}dx)=O(n^{\frac{3}{4}})$$

上述计算还能优化，即预处理一部分 s(n)。s(n)是一个积性函数的前缀和，先用线性筛计算一部分，假设已经预处理了前 k 个正整数的 s(n)，且 $k\geq\sqrt{n}$。因为 S(1)到 S(k)已经求出，那么 $\sum_{i=2}^{n}S(\left\lfloor\frac{n}{i}\right\rfloor)$ 这个式子中还没有求出的是 $k<\left\lfloor\frac{n}{i}\right\rfloor\leq n$，也就是要算 $1<i\leq\frac{n}{k}$ 的这一部分。那么复杂度变为：

$$T(n)=\sum_{i=2}^{\frac{n}{k}}O(\sqrt{\frac{n}{i}})\approx O(\int_0^{\frac{n}{k}}\sqrt{\frac{n}{x}}dx)=O(\frac{n}{\sqrt{k}})$$

k 取多大合适呢？线性筛计算也是要花时间的，而且是线性的 O(k)，那么线性筛计算加上杜教筛公式计算，总时间是：

$$T'(n)=O(k)+T(n)=O(k)+O(\frac{n}{\sqrt{k}})$$

差不多 $k=n^{\frac{2}{3}}$ 时，它有最小值，即：$O(n^{\frac{2}{3}})+O(\frac{n}{\sqrt{n^{\frac{2}{3}}}})=O(n^{\frac{2}{3}})$。

它比 $O(n^{\frac{3}{4}})$ 要好。

以上就是杜教筛算法的全部。

综合起来，杜教筛算法可以概况为：用狄利克雷卷积构造递推式，编程时用整除分块和线性筛优化，算法复杂度达到了 $O(n^{\frac{2}{3}})$。

杜教筛算法 = 狄利克雷卷积 + 整除分块 + 线性筛

应用杜教筛时，狄利克雷卷积是基本的工具。观察卷积的定义 $(f*g)(n)=\sum_{d|n}f(d)g(\frac{n}{d})$，那么，如果求解的函数有形如 $\sum_{d|n}f(d)$ 这样的性质，把这个性质与卷积的定义对照，就容易找到 g 和 h。下面用欧拉函数和莫比乌斯函数为例说明。算法的具体实现，见 6.4 节的模板代码。

6.2 欧拉函数前缀和

求欧拉函数前缀和：$\sum_{i=1}^{n}\phi(i)$。

求欧拉函数前缀和的杜教筛公式，需要用到欧拉函数性质：$n=\sum_{d|n}\phi(d)$。

（1）直接套用杜教筛公式

按卷积定义 $h=f*g=\sum_{d|n}f(d)g(\frac{n}{d})$，把 $n=\sum_{d|n}\phi(d)$ 改写为：$id=\phi*I$，那么有 $h=id$，$g=I$。代入杜教筛公式 $g(1)S(n)=\sum_{i=1}^{n}h(i)-\sum_{i=2}^{n}g(i)S(\lfloor\frac{n}{i}\rfloor)$，得：

$$\begin{aligned}S(n)&=\sum_{i=1}^{n}i-\sum_{i=2}^{n}S(\lfloor\frac{n}{i}\rfloor)\\&=\frac{n(n+1)}{2}-\sum_{i=2}^{n}S(\lfloor\frac{n}{i}\rfloor)\end{aligned}$$

（2）自己推导

读者如果有兴趣，也可以自己按部就班地以欧拉函数为例，做一次杜教筛公式推导的练习，推导过程是[①]：

$$n=\phi(n)+\sum_{d|n,d<n}\phi(d)$$

$$\phi(n)=n-\sum_{d|n,d<n}\phi(d)$$

$$\sum_{i=1}^{n}\phi(i)=\sum_{i=1}^{n}(i-\sum_{d|i,d<i}\phi(d))$$

$$\sum_{i=1}^{n}\phi(i)=\frac{n(n+1)}{2}-\sum_{i=1}^{n}\sum_{d|i,d<i}\phi(d)$$

现在改变枚举变量：枚举 $\frac{i}{d}$ 的值，即枚举 i 对 d 的倍数，因为 i≠d，所以从 2 开始：

$$\sum_{i=1}^{n}\phi(i)=\frac{n(n+1)}{2}-\sum_{\frac{i}{d}=2}^{\frac{i}{d}\le n}\sum_{d=1}^{d\le\lfloor\frac{n}{\frac{i}{d}}\rfloor}\phi(d)$$

设 $S(n)=\sum_{i=1}^{n}\phi(i)$，并把 $\frac{i}{d}$ 写成 i'：

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

这样就得到了欧拉函数的“杜教筛公式”，和（1）一样。

6.3 莫比乌斯函数前缀和

求莫比乌斯函数前缀和：$\sum_{i=1}^{n}\mu(i)$

求莫比乌斯函数前缀和的杜教筛公式，方法和上一节欧拉函数几乎一样。

需要用到莫比乌斯函数性质：$\sum_{d|n}\mu(d)=\begin{cases}1 & n=1\\0 & n>1\end{cases}$，为方便书写，改写成 $\sum_{d|n}\mu(d)=[n=1]$。

（1）套用杜教筛公式

按卷积定义 $h=f*g=\sum_{d|n}f(d)g(\frac{n}{d})$，把 $\sum_{d|n}\mu(d)=[n=1]$ 改写为：$\varepsilon=\mu*I$，那么有

[①]推导过程引用自：https://blog.csdn.net/qq_34454069/article/details/79492437

$h=\varepsilon,\ \ g=I$。代入杜教筛公式 $g(1)S(n)=\sum_{i=1}^{n}h(i)-\sum_{i=2}^{n}g(d)S(\left\lfloor\frac{n}{i}\right\rfloor)$，得：

$$S(n)=\sum_{i=1}^{n}\varepsilon(i)-\sum_{i=2}^{n}S(\left\lfloor\frac{n}{i}\right\rfloor)$$
$$=1-\sum_{i=2}^{n}S(\left\lfloor\frac{n}{i}\right\rfloor)$$

（2）自己推导[①]

$$[n=1]=\mu(n)+\sum_{d|n,d<n}\mu(d)$$
$$\mu(n)=[n=1]-\sum_{d|n,d<n}\mu(d)$$
$$\sum_{i=1}^{n}\mu(i)=1-\sum_{i=1}^{n}\sum_{d|i,d<i}\mu(d)$$
$$\sum_{i=1}^{n}\mu(i)=1-\sum_{\frac{i}{d}\le n}^{d\le\left\lfloor\frac{n}{\frac{i}{d}}\right\rfloor}\sum_{d=1}\sum_{i=1}^{n}\mu(d)$$
$$\sum_{i=1}^{n}\mu(i)=1-\sum_{\frac{i}{d}=2}^{}\sum_{d=1}\sum_{i=1}^{n}\mu(d)$$

令 $S(n)=\sum_{i=1}^{n}\mu(i)$，并把 $\frac{i}{d}$ 写成 i'，得：

$$S(n)=1-\sum_{i'=2}^{n}S(\left\lfloor\frac{n}{i'}\right\rfloor)$$

6.4 模板代码

下面给出洛谷 P4213（https://www.luogu.com.cn/problem/P4213）的代码[②]。

注释中说明了杜教筛的三个技术：整除分块、线性筛、狄利克雷卷积。其中整除分块、线性筛的代码和前面有关的代码一样。

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn = 5e6+7; //超过 n^2/3，够大了

int prime[maxn];   //记录质数
bool vis[maxn];    //记录是否被筛;
int mu[maxn];                    //莫比乌斯函数值
ll phi[maxn];                    //欧拉函数值
unordered_map<int,int> summu;    //莫比乌斯函数前缀和
unordered_map<int,ll> sumphi;    //欧拉函数前缀和
inline void init(){              //线性筛预计算一部分答案
    int cnt = 0;
    vis[0] = vis[1] = 1;
    mu[1] = phi[1] = 1;
    for(int i=2;i<maxn;i++){
        if(!vis[i]){
```

① 推导过程引用自： https://blog.csdn.net/qq_34454069/article/details/79492437

② 改写自：https://blog.csdn.net/KIKO_caoyue/article/details/100061406

```
            prime[cnt++] = i;
            mu[i] = -1;
            phi[i] = i-1;
        }
        for(int j=0;j<cnt && i*prime[j]<maxn;j++){
            vis[i*prime[j]] = 1;
            if(i%prime[j]){
                mu[i*prime[j]] = -mu[i];
                phi[i*prime[j]] = phi[i]*phi[prime[j]];
            }
            else{
                mu[i*prime[j]] = 0;
                phi[i*prime[j]] = phi[i]*prime[j];
                break;
            }
        }
    }
    for(int i=1;i<maxn;i++){  //最后，mu[]和 phi[]改为记录 1~maxn 的前缀和。
        mu[i] += mu[i-1];
        phi[i] += phi[i-1];
    }
}
inline int gsum(int x){         // g(i)的前缀和
    return x;
}
inline int getsmu(int x){
    if(x < maxn) return mu[x];   //预计算
    if(summu[x]) return summu[x];  //记忆化
    int ans = 1;                //杜教筛公式中的 1
    for(int l=2,r;l<=x;l=r+1){  //用整除分块计算杜教筛公式
        r = x/(x/l);
        ans -= (gsum(r)-gsum(l-1))*getsmu(x/l);
    }
    return summu[x] = ans/gsum(1);
}
inline ll getsphi(int x){
    if(x < maxn) return phi[x];
    if(sumphi[x]) return sumphi[x];  //记忆化，每个 sumphi[x]只用算一次
    ll ans = 1LL*x*(x+1)/2;      //杜教筛公式中的 n(n+1)/2
    for(int l=2,r;l<=x;l=r+1){   //用整除分块计算杜教筛公式，这里算 sqrt(x)次
        r = x/(x/l);
        ans -= (gsum(r)-gsum(l-1))*getsphi(x/l);
    }
    return sumphi[x] = ans/gsum(1);
}
int main(){
    init();  //用线性筛预计算一部分
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
```

```
        printf("%lld %d\n",getsphi(n),getsmu(n));
    }
    return 0;
}
```

6.5 题目

这里列出网上的一些题目。

（1）求 $\sum_{i=1}^{n} i\phi(i)$

题解：http://fzl123.top/archives/898

（2）求 $\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma(ij)$，其中 $n\leqslant 10^9$。

题解：https://www.cnblogs.com/zhugezy/p/11312301.html

（3）hdu 6706

http://acm.hdu.edu.cn/showproblem.php?pid=6706

（4）这里列出了一些可提交的题目：

https://www.cnblogs.com/TSHugh/p/8361040.html

致谢

杜瑜皓：对本文草稿提出细致的修改意见。

华东理工大学队员：傅志凌（oj.ecustacm.cn 管理员）、赵李洋、李震、彭博，讨论算法复杂度。

参考文献

[1] 积性函数求和的几种方法，任之洲，绍兴市第一中学，2016 年信息学奥林匹克中国国家队候选队员论文集

[2] 唐靖哲：https://blog.csdn.net/skywalkert/article/details/50500009

[3] 吉如一：http://jiruyi910387714.is-programmer.com/posts/195270.html

[4] 傅志凌：http://fzl123.top/archives/898