最大公约数和最小公倍数

定义

若自然数 d 同时是自然数 a 和 b 的约数，则称 d 是 a 和 b 的公约数，在所有 a 和 b 的公约数中最大的一个，称为 a 和 b 的最大公约数，记为 gcd(a, b).

若自然数 m 同时是自然数 a 和 b 的倍数，则称 m 是 a 和 b 的公倍数，在所有 a 和 b 的公倍数中最小的一个，称为 a 和 b 的最小公倍数，记为 lcm(a, b).

定理 1: $\forall a,b \in N, \gcd(a,b)\times lcm(a,b)=a\times b$

证明：

设 $d=\gcd(a,b), a_0=\dfrac{a}{d}, b_0=\dfrac{b}{d}$，根据最大公约数的定义，有 $\gcd(a_0,b_0)=1$，再根据最小公倍数的定义，有 $lcm(a_0,b_0)=a_0\times b_0$.

可得 $lcm(a,b)=lcm(a_0\times d,b_0\times d)=lcm(a_0,b_0)\times d=a_0\times b_0\times d=\dfrac{a\times b}{d}$.

注：$lcm(a,b)=\dfrac{a\times b}{\gcd(a,b)}$ 在计算时，$a\times b$ 的会超过 int 的范围，故一般写成 $lcm(a,b)=\dfrac{a}{\gcd(a,b)}\times b$.

定理 2（欧几里得算法）---辗转相除法：

$\forall a,b\in N, b\neq 0, \gcd(a,b)=\gcd(b,a\%b)$

证明：

若 $a<b$，则 $\gcd(b,a\%b)=\gcd(b,a)=\gcd(a,b)$，成立.

若 $a\ge b$，不妨设 $a=q\times b+r$，其中 $0\le r<b$，显然 $r=a\%b$. 对于 a, b 的任意公约数 d, 由于 $d\mid a 且 d\mid q\times b$，故 $d\mid(a-q\times b)$，即 $d\mid r$，因此 d 也是 b, r 的公约数，反之亦成立. 故 a, b 的公约数集合与 b, a%b 的公约数集合相同，故它们的最大公约数自然也相等.

使用欧几里得算法求最大公约数的时间复杂度为 $O(\log(a+b))$.

代码如下：

```
int gcd(int a,int b)//最大公约数
{
    if(b==0)
        return a;
    return gcd(b,a%b);
    //亦可用下行代码解决
    //return b?gcd(b,a%b):a;
}
```

```
int lcm(int a,int b)//最小公倍数
{
    return a/gcd(a,b)*b;
    //一般不写为a*b/gcd(a,b)
}
```

扩展（C++函数）——在 algorithm 头文件中：

```
    __gcd(a,b);//该函数前面是两个下划线。
```

素数筛

定义：

若一个正整数无法被除了 1 和本身之外的任何自然数整除，则称该数为**质数（或素数）**，否则称该正整数为**合数**.

在整个自然数集合中，质数的数量不多，分布比较稀疏，对于一个足够大的整数 N，不超过 N 的质数大约有 $\frac{N}{\ln N}$ 个，即每 $\ln N$ 个数中大约有 1 个质数.

质数的判定（试除法）：

若一个正整数 N 为合数，则存在一个能整除 N 的数 T，其中 $2 \le T \le \sqrt{N}$.

由此，可通过扫描 $2 \sim \sqrt{N}$ 之间所有的整数，依次检查它们能否整除 N，若都不能整除，则 N 是质数，否则 N 是合数. 试除法的时间复杂度为 $O(\sqrt{N})$，当然需**特判 0 和 1** 这两个整数，它们既不是质数也不是合数.

代码如下：

```
int is_prime(int n)//试除法判定质数
{
    if(n<2)
        return 0;
    for(int i=2;i<=sqrt(n);i++)
        if(n%i==0)
            return 0;
    return 1;
}
```

质数的筛选：

给定一个整数 N，求出 $1 \sim N$ 之间的所有质数，称为质数的筛选问题.

常用筛法:

1. 暴力筛 $O(n\sqrt{n})$

通过用判断素数的函数来枚举每一个数加入到素数表中.

```
int primes[MAXN],cnt;
int is_prime(int n)//试除法判定质数
{
    if(n<2)
        return 0;
    for(int i=2;i<=sqrt(n);i++)
        if(n%i==0)
            return 0;
    return 1;
}

void get_primes(int n)
{
    for(int i=1;i<=n;i++)
        if(is_prime(i)==1)
            primes[cnt++]=i;
}
```

时间复杂度分析:

遍历 $O(n)$, 素数判断最坏 $O(\sqrt{n})$, 总时间复杂度为 $O(n\sqrt{n})$

2. 朴素筛 $O(n\log n)$

任何整数 x 的的倍数 2x, 3x, . . . , kx 都不可能是素数, 从 2 往后扫描, 将当前数的所有倍数全部筛掉, 剩下没有被筛掉的数就是质数.

```
int primes[MAXN],cnt,vis[MAXN];

void get_primes(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(vis[i]==0)
            primes[cnt++]=i;
        for(int j=i+i;j<=n;j+=i)
            vis[j]=1;
    }
}
```

时间复杂度分析:

当 $i=2$ 时, 第二层循环共 $\frac{n}{2}$ 次, $i=3$ 时循环 $\frac{n}{3}$ 次, . . . , 总共循环了:

$$\frac{n}{2}+\frac{n}{3}+\cdots+\frac{n}{n}=n(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n})$$

其中 $\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$ 为调和级数, 当 $n\to\infty$ 时, 有: $\sum_{i=1}^{n}\frac{1}{i}=\ln n+\gamma$, 其中 γ 为欧拉常数, 约为 0.577215664. . . 故时间复杂度为:

$$n(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n})\approx n\ln n\approx n\log n\text{, 即 }O(n\log n)$$

3. 埃拉托斯特尼（埃氏）筛 $O(n\log\log n)$

对朴素筛法进行优化, 并不需要将每个数的倍数都筛去, 可只将所有质数的倍数筛去.
若按照朴素筛筛去一个整数的所有倍数, 最终会留下一些质数, 以 p 为例, 说明 p 不是 $2\sim p-1$ 内任何一个数的倍数, 并不需要将 $2\sim p-1$ 内的所有数依次枚举一遍, 只要将其中的质数枚举一遍即可, **合数等于质数之积**.

```
typedef long long ll;
int primes[MAXN],cnt,vis[MAXN];

void get_primes(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(vis[i]==0)
        {
            primes[cnt++]=i;
            for(int j=i+i;j<=n;j+=i)
                vis[j]=1;
        /*
            for(ll j=(ll)i*i;j<=n;j+=i)
                    vis[j]=1;
        */
        }
    }
}
```

这里还有个优化, j 可以从 $i\times i$ 开始枚举, 由于 $i\times(2\sim i-1)$ 已经被 $(2\sim i-1)$ 筛去. 由于数据范围的限制, 会使得 $i\times i$ 可能爆掉 int, 故开成 long long.

时间复杂度分析:

由于只用筛去$2 \sim n-1$的素数倍数即可,故调和级数变为:$\sum_{p \le n} \frac{1}{p}$,其中 p 为质数,当$n \to \infty$时,有:

$$M = \lim_{n \to \infty} (\sum_{p \le n} \frac{1}{p} - \ln(\ln(n))) = \gamma + \sum_{p} [\ln(1 - \frac{1}{p}) + \frac{1}{p}]$$

其中, M 为 $Meissel-Mertens$ **常数**,也称 $Mertens$ **常数**或**质数倒数和常数**,数值约为0.261497⋯,当 $n \to \infty$ 时,$n \times \sum_{p \le n} \frac{1}{p} \approx n \ln(\ln n)$,即为$O(n \log \log n)$

4. 线性(欧拉)筛 $O(n)$

埃氏筛存在一个缺陷,即对于一个合数,可能会被筛很多次,例如$30 = 2 \times 15 = 5 \times 6 \cdots$,改用其最小质因子去筛掉这个合数,就可以保证它只会被筛一次.

从小到大枚举所有的质因子 primes[j].

①当出现 i%primes[j]==0 时,primes[j]一定是 i 的最小质因子, 因此也一定是primes[j]*i 的最小质因子.

②当出现 i%primes[j]!=0 时,说明尚未枚举到 i 的任何一个质因子,也就表示primes[j]小于 i 的任何一个质因子,这时 primes[j]一定是 primes[j]*i 的最小质因子.

无论如何,primes[j]都一定是 primes[j]*i 的最小质因子,且所筛选的质数在$2 \sim n$之间, 因此合数最大为 n,故 primes[j]*i 只需枚举到 n 即可,但由于 primes[j]*i 可能会溢出整数 int 范围,故改成 primes[j]<=n/i 的形式.

```
int primes[MAXN],cnt,vis[MAXN];

void get_primes(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(vis[i]==0)
            primes[cnt++]=i;
        for(int j=0;primes[j]<=n/i;j++)
        {
            vis[primes[j]*i]=1;
            if(i%primes[j]==0)//primes[j]一定是 i 的最小质因子
                break;
        }
    }
}
```

时间复杂度分析：

$2\sim n$ 中，任何一个合数都会被筛去，而且仅用最小质因子去筛，每个合数都有且仅有一个最小质因子，故每个合数只会被筛一次，所以时间复杂度为 $O(n)$。