# 信息竞赛中的数学

- 数论
- 线性代数
- 组合数学
- 概率论
- 博弈论
- 几何
- ……

# 数论基础选讲（一）

山东省实验中学
宁华

# 数论

- 数论是纯粹数学的分支之一，主要研究**整数**的性质。

- 卡尔·弗里德里希·高斯曾说：“数学是科学的皇后，数论是数学的皇后。”

# 数论四大定理

- 威尔逊定理
- 欧拉定理
- 孙子定理
- 费马小定理

- 并称数论四大定理。

# 信息竞赛中的数论基础

- 约数
- 余数
- 质数

# 1、整除、约数、倍数

- 整除：若整数a除以非零整数b的余数为0， 我们就说a能被b整除(或说b能整除a)，记作：b|a 。

- a叫做b的倍数，b叫做a的约数（或因数）。

# 显而易见的性质

- （1）1能整除任何数，任何数都能整除0
- （2）若a|b，a|c，则a|(b+c)，a|(b-c)
- （3）若a|b，则对任意整数c，a|(b*c)
- （4）传递性：若a|b，b|c，则a|c

# 例题：求n的正约数集合

- 1 <= n <= 10^15

- 样例输入：
- 12
- 样例输出：
- 1 2 3 4 6 12

# 试除法

- 1..n
- O(n)
- TLE

# 试除法

- 分析：
- 注意到约数总是成对出现：若k是n的约数,则(n/k)也是n的约数。
- 在一对约数中，必有一个不大于sqrt(n)，另一个不小于sqrt(n)。
- 注意点：完全平方数的情况，sqrt(n)是单独出现的。

- 因此枚举1..sqrt(n)就能求出n的所有约数。O( sqrt(n) )

- 试除法的推论：一个整数N的约数个数上界为2*sqrt(n)

# 例题：[CF 762A] k-th divisor

- 求n的第k小的约数。如果不存在输出-1。空间限制256M，时限2S。
- $1 \le n \le 10^{15}$, $1 \le k \le 10^{9}$

Examples
Input
4 2
Output
2
Input
5 3
Output
-1
Input
12 5
Output
6

Note

In the first example, number 4 has three divisors: 1, 2 and 4. The second one is 2.

In the second example, number 5 has only two divisors: 1 and 5. The third divisor doesn't exist, so the answer is -1.

# 分析

- $1 \le n \le 10^{15}$, $1 \le k \le 10^{9}$

- 虽然k可以高达10^9
- 但是对于$1 \le n \le 10^{15}$
- 一个整数n的约数个数上界为2*sqrt(n)
- 所以n不会有10^9这么多个因子

# 方法1：注意完全平方数

```
#define ll long long
const int N=3e7+10;
ll n,k,q[N],tot;
int main()
{
    cin>>n>>k;
    ll t=sqrt(n);
    for(ll i=1;i<=t;i++)if(n%i==0)q[++tot]=i;
    for(ll i=t-(t*t==n);i>=1;i--)if(n%i==0)q[++tot]=n/i;
    if(tot<k)cout<<-1<<endl;
    else cout<<q[k]<<endl;
    return 0;
}
```

# 方法2：set 去重

```
cin>>n>>k;
set<ll>divisors;
ll t=sqrt(n);
for(ll i=1;i<=t;i++)
    if (n%i==0)
    {
        divisors.insert(i);//把i压入set
        divisors.insert(n/i);//把另一半也压进去
    }
if(k>divisors.size())puts("-1");
else
{
    set<ll>::iterator iter=divisors.begin();//使其指向容器的第一个元素
    for (ll i=1;i<=k-1;i++,iter++) ;
    cout<<*iter<<endl;//注意输出的时候要加上*
}
```

# 例题：求1~n每个数的正约数集合

- n<=5*10^5
- 样例输入：
- 5
- 样例输出：
- 1
- 1 2
- 1 3
- 1 2 4
- 1 5

# 例题：求1~n每个数的正约数集合

- 试除法 O( n*sqrt(n) ) 复杂度过高
- 怎么办？

# 例题：求1~n每个数的正约数集合

- 试除法 O( n*sqrt(n) ) 复杂度过高
- 可以反过来考虑，对于每个数d，1~n中以d为约数的数就是d的倍数：d, 2d, 3d, ... , [n/d]*d
- 倍数法

# 倍数法

```
const int N=5e5+5;
vector<int>factor[N];
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n/i;j++)
            factor[i*j].push_back(i);
    for(int i=1;i<=n;i++)
    {
        for(int j=0;j<factor[i].size();j++)
            printf("%d ",factor[i][j]);
        puts("");
    }
    return 0;
}
```

- 倍数法 O(N+N/2+N/3+...+N/N)=O(NlogN)
- 倍数法的推论：1~N每个数的约数个数的总和大约为NlogN。

# 最大公约数

- 设a,b是不都为0的整数，c为满足 c|a 且 c|b 的最大整数，则称c是a,b的最大公约数
- 记为gcd(a,b)或简记为(a,b)
- GCD: Greatest Common Divisor

- 类似地可以定义多个数的最大公约数

# 最大公约数

- 一些性质：
  - (a,a)=(0,a)=a
  - 若a|b，则(a,b)=a
  - (a,b)=(b, a mod b)
  - (a,b)=(a,a+b)=(a,ka+b) (?)
  - (ka,kb)=k·(a,b)
  - (a,b,c)=((a,b),c)

- 若(a,b)=1，则称a,b互质（互素）
- **互质的两个数往往有很好的性质**

- $(a,b)=(b, a \bmod b)$
- 证明？

- (a,b)=(b, a mod b)
- 证明：
- 设c=gcd(a,b),那么a可以表示为mc, b可以表示为nc的形式，即a=mc, b=nc
- 令a=kb+r,那么我们就只需要证明gcd(b,r)=c即可。
- ∵r=a−kb=mc−knc,∴gcd(b,r)=gcd(nc,mc−knc)=c*gcd(n,m-kn)
- 我们只需要证明gcd(n,m-kn)=1即可。
- 设gcd(n,m-kn)=d,则n可以表示为xd, m-kn可以表示为yd,则有n=xd, m=kxd+yd, 进而a=mc=(kx+y)dc, b=nc=xdc， ∴ gcd(a,b) = c*d*gcd(kx+y, x)
- 而gcd(a,b)=c, ∴ d*gcd(kx+y, x) = 1 则 d=1，得证。

- (a,b)=(a,a+b)=(a,ka+b) (?)
- 直接证明(a,b) = (a,a+b)很难，但是我们如果能证明a,b的公约数和a,a+b的公约数完全相同就好了。
- 我们假设任意一个a,b的公约数p，p是a,b的约数，那么p也是a+b的约数。公约数相同，则最大公约数也相同，这样我们就完成了(a,b)=(a,a+b)的推导。同样，第二个等式也可依此推出。
- 这个证明利用的是整除的性质和我们的假设：任意的约数。

# 求GCD方法1：

- 求GCD的一般公式：质因数分解

$$a = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$$

$$b = p_1^{b_1} p_2^{b_2} \ldots p_k^{b_k}$$

$$(a,b) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \ldots p_k^{\min(a_k,b_k)}$$

- 例：
- 100=2^2*5^2
- 120=2^3*3^1*5^1
- 则(100, 120)=2^2*3^0*5^1=20

# 求GCD方法2：欧几里德算法

- 又称辗转相除法，是一种迭代求两数gcd的做法
- 由(a,b)=(a,ka+b)的性质：gcd(a,b)=gcd(b, a mod b)

- ```
  int gcd(int a, int b) {
      if (b==0) return a;
      return gcd(b, a % b);
  }
  ```

- 容易证明这么做的复杂度是O(log n)。 (?)
  ```

- 或:
- ```
  int gcd(int a, int b) {
      return b ? gcd(b, a % b) : a;
  }
  ```
  ```

# 非递归实现

```
int gcd(int a, int b)
{
    while(b)
    {
        int t = b;
        b = a % b;
        a = t;
    }
    return a;
}
```

# 求GCD方法3：更相减损法

- 九章算术

# 例题：[CF 664A]Complicated GCD

- 求gcd(a, a+1, a+2, ..., b)
- $1 \le a \le b \le 10^{100}$

- Input
- The only line of the input contains two integers a and b (1 ≤ a ≤ b ≤ 10^100).
- Output
- Output one integer — greatest common divisor of all integers from a to b inclusive.

- Examples
- input
- 1 2
- output
- 1
- input
- 6180339887498948482045868343656381177203091798057661803398874989484820458683436563811772030917980576
- output
- 6180339887498948482045868343656381177203091798057661803398874989484820458683436563811772030917980576

- 分析：
- 注意到gcd(a, a+1) = 1
- 因此a<b时答案为1，否则答案为a

# 例题：[CF 757B]Bash's Big Day

- 给定n个正整数$\{a_i\}$
- 求一个子集S，满足$\gcd(S_1, ..., S_k)>1$，同时$|S|$尽可能大。
- $1\le n,a_i\le 10^5$

- Examples
- input
- 3
- 2 3 4
- output
- 2
- input
- 5
- 2 3 4 6 7
- output
- 3

- 分析：
- gcd>1，说明存在一个正整数d>1，满足d整除S内的所有元素。

- 枚举d=2,3,4,..max{$a_i$}并统计答案
- 复杂度为O(n^2)。

- TLE,有没有什么优化方法呢？

- 分析：
- gcd>1，说明存在一个正整数d>1，满足d整除S内的所有元素。

- 其实对于这种枚举的优化，我们无非是减少枚举层数或者避免枚举一定不会有解的答案，可以发现，我们只需要枚举当前的质因数就可以了。
- 枚举d=2,3,5,7,..pm[k]并统计答案
- 则复杂度为O(nlogn)。

# 例题 NOIP2009 Hankson的趣味题

- 分析

# 例题：[SDOI2009] SuperGCD

- Input
- 共两行： 第一行：一个数A。 第二行：一个数B。
- Output
- 一行，表示A和B的最大公约数。
- Sample Input
- 12
- 54
- Sample Output
- 6
- HINT
- 对于20%的数据，0 < A , B ≤ 10 ^ 18。
- 对于100%的数据，0 < A , B ≤ 10 ^ 10000。

# 2、带余除法、同余

- 对于整数a,b，b>0，则存在唯一的整数q,r，满足a=bq+r，其中0≤r<b。

- a ÷ b = q … r

- 其中称q为商、r为余数。
- 余数用 a mod b 表示。

# 同余定理

- 同余概念：最初是由德国数学家高斯提出的。
- 若两数a,b除以m的余数相等，则称a,b模m同余，记做a≡b(mod m)。
- 同余定理：
- 给定一个正整数 m ，如果两个整数 a 和 b 满足m|(a-b)，那么就称整数 a 与 b 对模m同余，记作a≡b(mod m)。
- m|(a-b) ⇔ a≡b(mod m)

- $m|(a-b) \Leftrightarrow a \equiv b \pmod{m}$
- 例如：
- $a=16, b=10, m=3$
- $3|(16-10) \Leftrightarrow 16 \equiv 10 \pmod{3}$

- 如何证明？
- 充分性
- 必要性

- 推论：
- 若$a \equiv b \pmod{c}$，$d \mid c$
- 则$a \equiv b \pmod{d}$

- 例如，$16 \equiv 10 \pmod{6}$, $3 \mid 6$
- $\Rightarrow 16 \equiv 10 \pmod{3}$

- 证明略。

# 性质/推论

- 1. 反身性 a≡a (mod m)
- 2. 对称性 若a≡b(mod m)，则b≡a (mod m)
- 3. 传递性 若a≡b (mod m)，b≡c (mod m)，则a≡c (mod m)
- **4. 线性运算：**
- **若a≡b (mod m)，c≡d(mod m)，**
- **则a±c≡b±d (mod m)，a*c≡b*d (mod m)**

- 5. 除法：若a*c ≡ b*c (mod m) c≠0
- 则 a≡ b (mod m/gcd(c,m))
- 特殊地 ,gcd(c,m)=1 则a ≡ b (mod m)
- 6. 幂运算：如果a ≡ b (mod m)，
- 那么a^n ≡ b^n (mod m)
- 7. 若a ≡ b (mod m)，n|m,则 a ≡ b (mod n)
- 例如，16≡10(mod 6)，3|6 ⇒16≡10(mod 3)
- 8. 若a ≡ b (mod mi) (i=1,2...n) 则 a ≡ b (mod [m1,m2,...mn]) 其中[m1,m2,...mn]表示m1,m2,...mn的最小公倍数

# 最常用

- (a+b)%c=(a%c + b%c) %c
- (a-b)%c=(a%c - b%c) %c
- (a*b)%c=(a%c) * (b%c) %c

- (a^n)%c=(a^(n/2) * a^(n/2))%c (n是偶数)
- (a^n)%c=(a^(n/2) * a^(n/2) * a)%c (n是奇数)

# 如何用同余定理解题？

- 例题：求2001^2003除以13的余数

- 根据同余性质：
- 6. 幂运算：如果a ≡ b (mod m)，那么a^n ≡ b^n (mod m)
- 我们可以得出2001^2003≡12^2003 (mod 13)
- 12^2003还是一个较大的数，很难求出它除以13的余数，这时，我们就要找出12的几次方与1对于模13是同余的。根据试验，可得出12^2≡1 (mod 13)
- 我们把12^2003拆成 (12^2)^1001×12^1，而
- (12^2)^1001×12^1≡1×12(mod 13)≡12(mod 13)
- 这时，我们可以得出2001^2003除以13的余数为12，我们用计算器计算一下，这个答案是对的。

- 求a^b%c这类问题，也可以用欧拉定理求解。请同学们课后研究。

# 如何用同余定理解题？

- 求(11^11+13^13) mod 12

- 求(11^11+13^13)mod 12
- =(11^11 mod 12 + 13^13 mod 12) mod 12
- =(((11^2)^5*11)mod 12 +1)mod 12
- =((1*11)mod 12 + 1 ) mod 12
- =0

# 快速幂求模

- （1）基于分治的快速幂

- (a^n)%c=(a^(n/2) * a^(n/2))%c (n是偶数)
- (a^n)%c=(a^(n/2) * a^(n/2) * a)%c (n是奇数)

```
int a,n,p;
int f(int n)
{
    if(n==0)return 1%p;
    int s=f(n/2);
    s=(1LL*s*s)%p;
    if(n%2)s=1LL*s*a%p;
    return s;
}
int main()
{
    cin>>a>>n>>p;
    cout<<f(n)<<endl;
    return 0;
}
```

- （2）基于二进制拆分的快速幂

# 3、质数与合数

- 若大于1的正整数p仅有两个因子1和p，则称p是一个质数（素数）。
- 否则，若p>1，则称p是一个合数。
- 1不是质数也不是合数
- 若n是一个合数，则n至少有1个质因子。因此其中最小的质因子一定不大于sqrt(n)
- 质数有无穷多个。
- n以内的质数约有n/ln(n)个。

# （1）素数判断及求解

- 素数总是一个比较常涉及到的内容，掌握求素数的方法是一项基本功。

- 常见两个问题：
- 判断一个数是否为素数？
- 求出n以内的素数？

# 第一个问题：判断一个数是否为素数

- 如何判断？

# 傻瓜解法

- 2 ~ n-1
- O(n)

# 一般解法——试除法

- 2 ~ sqrt(n)

- 试除法标准版：大部分人都知道的比较快的方法：判断从2到sqrt(n)是否存在其约数，时间复杂度O(sqrt(n))

# 试除法

```
bool is_prime(int n)
{
    if(n<2) return 0;
    int m=sqrt(n);
    for(int i=2;i<=m;i++)
        if(n%i==0) return 0;
    return 1;
}
```

- O(sqrt(n))

- 试除法高配版：判断2之后，就可以判断从3到sqrt(n)之间的奇数了，无需再判断之间的偶数，时间复杂度O(sqrt(n)/2)

- 试除法尊享版：
- 首先看一个关于质数分布的规律：
- 大于等于5的质数一定和6的倍数相邻。

- 例如5和7，11和13,17和19等等；

```
int isPrime(int n)
{       //返回1表示判断为质数，0为非质数
        float n_sqrt;
        if(n==2 || n==3) return 1;
        if(n%6!=1 && n%6!=5) return 0;
        n_sqrt=floor(sqrt((float)n));
        for(int i=5;i<=n_sqrt;i+=6)
        {
            if(n%(i)==0 | n%(i+2)==0) return 0;
        }
        return 1;
}
```

# 有没有更快的方法？

# Miller-Rabin算法

# 预备知识：费马小定理

- 如果p是一个质数，而a不是p的倍数，则有：
- a^(p-1)%p==1
- 证明：略。

# 逆定理？

- 由费马小定理，我们可以有一个大胆的猜想：
- 如果a^(p-1)%p==1，则p是质数。

- 可惜，这样的猜想是错误的。
- 例如：存在 a=2，p=341 时满足 2^340 ≡ 1 (mod 341)，然而 341 = 11 * 31 却是合数。
- 类似于341这样满足费马小定理式子但不是素数的数，被称作伪素数。

- 所以，我们自然又会萌生出这样一种想法：
- 取不同的 a 多验证几次，这样可以大大降低出错的概率。

- 不过，∀a<561, a^560 ≡ 1 (mod 561)，然而 561 = 3 * 11 *17 。

# 二次探测定理

- 如果 p 是一个素数, 且 0<x<p, 则方程 x^2 % p = 1 的解为:
- x=1 或 x=p-1.

- 证明：
- ∵ x^2≡1(mod p)
- ∴ p | x^2−1
- ∴ p | (x+1)(x−1)
- ∵ p是大于x的质数
- ∴ p=x+1 or p≡x−1 (mod p)，即 x=1 or p−1。

# 这个定理和素数判定有什么用呢？

- 假设需要判断的数是p
- 若p%2==0则p不是素数（p==2特判）。否则p-1一定为偶数
- 我们把p−1分解为 t * 2^k 的形式
- 当p是素数，有a^(t * 2^k)≡1(mod p)
- 然后随机选择一个数a，计算出 a^t(mod p)
- 让其不断的自乘，同时结合二次探测定理进行判断
- 如果我们自乘后的数(mod p)=1，但是之前的数(mod p)≠1或p-1
- 那么这个数就不是素数(违背了二次探测定理)
- 这样乘k次，最后得到的数就是a^(p−1)
- 那么如果最后计算出的数不为1，这个数也不是素数(费马小定理)

# Miller-Rabin随机性素数测试方法

- 这里推荐几个 a 的值： 2，3，5，7，11，61，13，17。用了这几个 a，就连那个被称为强伪素数的 46856248255981 都能被除去。

- 二次探测结合费马小定理，正确率就相当高了。

# 代码：

# 第二个问题：求n以内的素数（或素数个数）

- 例：求10000000以内的素数

# 傻瓜解法

- O(n^2)
- O(n*sqrt(n))

# 快速解法——筛法

- 普通筛法——埃拉托斯特尼筛法/埃氏筛法（Eratosthenes筛法）
- 快速筛法——线性筛/欧拉筛法

# 普通筛

- 基本思想：任意整数x的倍数 2x,3x,... 一定不是素数
- 实现方法：用一个长度为N+1的数组保存信息（0表示素数，1表示非素数），先假设所有的数都是素数（初始化为0），从第一个素数2开始，把2的倍数都标记为非素数（置为1），一直到大于N；然后进行下一趟，找到2后面的下一个素数3，进行同样的处理，直到最后，数组中依然为0的数即为素数。
- 说明：整数1特殊处理即可。

- 举个例子，N=20时，演示如下图：

| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

# 普通筛

```
memset(mark, 0, sizeof(mark));
tot = 0;
for (int i = 2; i <= n; ++i)
{
    if (mark[i]) continue;
    prime[++tot] = i;
    for (int j = i+i; j <= n; j += i)
        mark[j] = 1;
}
```

# 普通筛存在的问题？

| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

# 如何优化？

| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

- 小于 x*x 的 x 的倍数在扫描更小的数时就已经被筛掉了。

- 所以，对于每个数 x ，只需要从 x*x 开始，把 x*x,x*(x+1),x*(x+2),...,x*(n/x)筛掉即可。

# 普通筛+优化

```
memset(mark, 0, sizeof(mark));
tot = 0;
for (int i = 2; i <= n; ++i)
{
    if (mark[i]) continue;
    prime[++tot] = i;
    for (int j = i*i; j <= n; j += i)
        mark[j] = 1;
}
```

- 埃氏筛法
- $O(\sum_{质数p<=N}(N/p)) = O(NloglogN)$
- 效率已经非常接近于线性，是算法竞赛中常用的质数筛法。

# 普通筛+优化 还存在问题吗？

# 普通筛+优化 还存在问题吗？

- 例如：12
- 既会被2筛掉，又会被3筛掉。

- 产生这个问题的原因？如何解决？

# 线性筛

- 欧拉筛法求素数又称为线性筛法求素数，是一种将求一定范围内的素数的时间复杂度控制在O(n)的一种算法，它跟埃拉特斯尼筛法求素数很像，做了一定的改进。
- 埃拉特斯尼筛法之所以没有达到线性筛的水平是因为它对于同一个合数，要重复操作它的质因子个数次，这是很浪费的，而欧拉筛法求素数则去掉了重复操作，它在当合数i可以整除质数时跳出，对下一个数进行操作。比方说，当i = 4时，不会筛掉12这个数，因为12的最小的质因数是2，12 = 2 * 6，6定义为它相对最大的合数因子，那么4不满足这个条件，所以不在i =4 时筛掉12，而是在i = 6的时候筛掉12.这样就避免了重复操作。

# 线性筛

| i | j | p[j] | tot | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 3 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 % 2 == 0 break; 下面类似的不再赘述 | | | | | | | | | | | | | | | | | | | | | |
| 5 | 0 | 2 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 3 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 5 | 3 | 5 * 5 > 20 break; 下面类似的不再赘述 | | | | | | | | | | | | | | | | | | |
| 6 | 0 | 2 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 2 | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 2 | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 9 | 0 | 2 | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 10 | 0 | 2 | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

- 此过程中保证了两点：
- 1、合数一定被干掉了...
- 2、每个合数都只被干掉了一次

```
for (int i=2;i<=n;i++)
        {
                if (!mark[i])
                        prime[++tot]=i;
                for(int j=1; j<=tot && prime[j]*i<=n; j++)
                {
                        mark[prime[j]*i]=1;
                        if(i%prime[j]==0)break;
                }
        }
```

# 例题 POJ2689 Prime Distance

- 求一个长度不超过1e6的区间[L,R]距离最近和最远的素数对。
- 1 <= L < R < 2^31

# （2）质因数分解

- 输入：
- 120
- 输出：
- 2 3
- 3 1
- 5 1

# （2）质因数分解

- 利用n最多只有1个>sqrt(n)的质因子，可以得到一个O(sqrt(n))的质因数分解算法。

```
vector<int> factor(int x) {
    vector<int> ret;
    for (int i = 2; i * i <= x; ++i)
        while (x % i == 0) {
            ret.push_back(i);
            x /= i;
        }
    if (x > 1) ret.push_back(x);
    return ret;
}
```

# 例题 求一个合数的最大质因子

- 如：13195的质数因子有5,7,13和29。最大的是29。
- 600851475143的最大质数因子是多少？

# 例题 [NOIP 2012普及组 No.1] 质因数分解

- 【题目描述】
- 已知正整数n是两个不同的质数的乘积，试求出较大的那个质数。
- 【输入格式】
- 输入只有一行，包含一个正整数n。
- 【输出格式】
- 输出只有一行，包含一个正整数p，即较大的那个质数。
- 【样例输入】
- 21
- 【样例输出】
- 7
- 【输入输出样例说明】
- 21=3*7，7>3，故输出7。
- 【数据规模】
- 对于60%的数据，6 ≤ n ≤ 1000。
- 对于100%的数据，6 ≤ n ≤ 2*10^9。

# 例题 NYIST OJ-No.34 阶乘因式分解(I)

- 时间限制3000ms，内存限制65535KB，难度2

- 描述

- 给定两个数m,n,其中m是一个素数。
- 将n（0<=n<=10000）的阶乘分解质因数，求其中有多少个m。

- 输入
- 第一行是一个整数s（0 < s <= 100)，表示测试数据的组数
- 随后的s行, 每行有两个整数n，m。

- 输出
- s行，每行对应输出n!中m的个数。

样例输入
3
4 2
100 5
16 2
样例输出
3
24
15

# 例：求 20! 中含有的质因子 2 的个数

- a(2)=[20/2]+[20/4]+[20/8]+[20/16]=18;

- 解释：
  2、4、6、8、10、12、14、16、18、20能被2整除
  4、8、12、16、20能被4整除（即被2除一次后还能被2整除）
  8、16能被8整除（即被2除两次后还能被2整除）
  16能被16整除（即被2除三次后还能被2整除）
  这样就得到了2的阶。

- n!=1*2*3*……*(n-2)*(n-1)*n
- 可以表示成所有和m倍数有关的乘积再乘以其他和m没有关系的
- =(m*2m*3m*.....*km)*ohter
- other是不含因子m的数的乘积
- 因为 km<=n 而k肯定是最大值 所以k=n/m
- =m^k*(1*2*.....*k)*other
- =m^k*k!*other
- 从这个表达式中可以提取出k个m，然后按照相同的方法循环（递归）下去可以求出k!中因子m的个数。 每次求出m的个数的和就是n!中因子m的总个数。

# 如何求 n! 中含有的某个质因子 p 的个数？

```
int cal(int n, int p) {
        if(n < p) return 0;
        else return n / p + cal(n / p, p);
}
```

# 例题 NYIST OJ-No.70 阶乘因式分解(II)

- 描述
- 给定两个数n，m,其中m是一个素数。将n（0<=n<=2^31）的阶乘分解质因数，求其中有多少个m。
- 输入
- 第一行是一个整数s（0<s<=100)，表示测试数据的组数
- 随后的s行, 每行有两个整数n，m。
- 输出
- s行，每行一个整数，表示每组数据中的 n! 含有的质因子 m 的个数

# 例题 NYIST OJ-No.70 阶乘因式分解(II)

- 样例输入
- 3
- 100 5
- 16 2
- 1000000000  13
- 样例输出
- 24
- 15
- 83333329

# 例题 acw197 阶乘分解

- 数据范围
- 1≤N≤10^6
- 输入样例：
- 5
- 输出样例：
- 2 3
- 3 1
- 5 1
- 样例解释
- 5! = 120 = 2^3 * 3 * 5

# 例题：[CF 776B]Sherlock and his girlfriend

- n个点，标号2..n+1，给这些点染色，要求若a是b的质因子，则a和b的颜色不同。
- 求一种颜色数最少的方案并输出。如果有多种方案需要的最少颜色数相同，则任意输出一种方案。
- 2<=n<=1000

- 输入输出样例
- 输入 #1
- 3
- 输出 #1
- 2
- 1 1 2
- 输入 #2
- 4
- 输出 #2
- 2
- 2 1 1 2

- 分析：注意到这是二分图，一边是质数，一边是合数。
- 把质数都染成1，合数都染成2即可。

# 4、唯一分解定理

- 正整数唯一分解定理（算术基本定理）

- 大约公元前350年，欧几里德在他伟大的十三卷著作《原本》中，用了许多篇幅来讨论素数。特别是他证明了每一个比1大的正整数，要么本身是一个素数，要么可以写成一系列素数的乘积。如果不考虑这些素数在乘法式中的顺序，那么写出来的形式是唯一的。

# 唯一分解定理

- 例如：
- 21 = 3 * 7
- 6936 = 2*2*2*3*17*17 = 2^3 * 3 * 17^2
- 等号右边的表达式被称作是左边数的“素数分解”。
- 这个事实被称为算术基本定理，它告诉我们素数好比化学中的原子——所有整数得以构成的基本砌块。

# 唯一分解定理

- 算术基本定理(The fundamental theorem of arithmetic) 即唯一分解定理可表述为：
- 任何一个大于1的自然数 N，如果N不为质数，那么N可以唯一分解成有限个质数的乘积，N=P1^a1*P2^a2*P3^a3*.....*Pn^an，这里P1<P2<P3.....<Pn均为质数，其中指数ai是正整数。这样的分解称为 *N* 的标准分解式。
- 6936=2^3×3×17^2

# 算术基本定理的推论

- 若正整数n可分解为p1^a1*p1^a2*...*pk^ak，其中pi为两两不同的素数，ai为对应指数
- n的约数集合可以写作：{ p1^b1 * p2^b2 * ... * pk^bk | 0<=bi<=ai }

- n的约数个数为：(1+a1)*(1+a2)*....*(1+ak)
- n的约数的和为：(1+p1+p1^2+...+p1^a1)*...*(1+pk+pk^2+...+pk^ak)

- 如24=2*2*2*3=2^3*3^1
- 则24的约数个数为(1+3)*(1+1)=8
- 约数的和为：(1+2+2^2+2^3)*(1+3)=60
- 实际上24的约数有1,2,3,4,6,8,12,24共8个, 1+2+3+4+6+12+24=60
- 再如180=2*2*3*3*5=2^2*3^2*5^1
- 则180的约数个数为(1+2)*(1+2)*(1+1)=18个。

# 例题 POJ 2603

- Description
- 求十个数乘积的因子个数
- Input
- 十个整数（1<=ai<=10000)
- Output
- 输出十个整数乘积的因子个数

Sample Input
1
2
6
1
3
1
1
1
1
1
Sample Output
9

# 例题 求 N! 的约数的个数

- 样例输入：
- 4
- 样例输出：
- 8
- 样例解释：
- 4!=1*2*3*4=24
- 24的约数有1,2,3,4,6,8,12,24共8个
- 所以输出8

# 例题 求 N! 的约数的个数

- 例：求 20! 的约数的个数

# 例题 求 N! 的约数的个数

- 例：求 20! 的约数的个数
- 设 m = 20! = p1^a1 * p2^a2 * ... * pk^ak
- 只要求出所有质因子pi的幂ai，则：
- ans=(1+a1)*(1+a2)*...*(1+ak)

- 20! 会包含哪些质因子呢？

# 例：求 20! 的约数的个数

- 1. 先求出20以内的质数：(2,3,5,7,11,13,17,19)　//怎么求？
- 则 20! 一定且只包含以上质因子

- 2. 再求各个质数pi的阶数(幂)ai：
- 如何求 n! 中含有的某个质因子 p 的个数？

# 例：求 20! 的约数的个数

- 1. 先求出20以内的质数：(2,3,5,7,11,13,17,19)　//怎么求？
- 2. 再求各个质数pi的阶数(幂)ai：

- a(2)=[20/2]+[20/4]+[20/8]+[20/16]=18;
- a(3)=[20/3]+[20/9]=8;
- a(5)=[20/5]=4;
- ...
- a(19)=[20/19]=1;

- 所以 20!=2^18*3^8*5^4*...*19^1

# 例：求 20! 的约数的个数

- 3. 根据算术基本定理的推论得到答案：

- 20!=2^18*3^8*5^4*...*19^1
- 所以：
- 20!的约数个数=(1+18)*(1+8)*……*(1+1)

# 算术基本定理的推论

- 若求A/B的约数个数，
- A可分解为p1^a1*p2^a2*...*pk^ak，
- B可分解为p1^b1*p2^b2*...*pk^bk,
- 则A/B的约数个数为
- (a1-b1+1)*(a2-b2+1)*(a3-b3+1)...*(ak-bk+1).

- 例如：求144/6的约数个数
- 144=2^4*3^2
- 6=2^1*3^1
- 则144/6的约数个数为(4-1+1)*(2-1+1)=8
- 实际上144/6=24的约数有1,2,3,4,6,8,12,24共8个

# POJ 2992

- 输入n、k，求C(n,k)约数的个数

```
while(cin>>n>>k){
    if(2*k>n)  k=n-k;
    for(i=1,m=1;prime[i]<=n,i<t;i++)
        m*=(cal(n,prime[i])-cal(k,prime[i])-cal(nk,prime[i])+1);
    printf("%d/n",m);
}
```

# http://ac.jobdu.com/problem.php?pid=1493

- 题目描述：
- 给定两个正整数a，b（1<=a,b<=100000000）,计算他们公约数的个数。
- 如给定正整数8和16，他们的公约数有：1、2、4、8，所以输出为4。
- 输入：
- 输入包含多组测试数据，每组测试数据一行，包含两个整数a，b。
- 输出：
- 对于每组测试数据，输出为一个整数，表示a和b的公约数个数。
- 样例输入：
- 8 16
- 22 16
- 样例输出：
- 4
- 2

- 解法一：
- 要求所有公约数的个数，我们可以先用辗转相除法求出最大公约数，然后求出这个最大公约数的所有约数的个数，也即a和b的公约数的个数

- 解法二：
- 对a、b分别分解质因子，然后根据算术基本定理，求出公约数个数。

# 例 反质数

- [HAOI2007] BZOJ1053

# 欧拉函数

- 欧拉函数：对于一个正整数n，[1,n] 中与n互质的数的数目，记作φ(n)。

- φ(1)=1
- φ(8)=4 （ 1,3,5,7和8互质 ）

- 此函数以其首名研究者欧拉命名(Euler's totient function)，它又称为φ函数、欧拉商数等。

# 欧拉函数

- （1）对于素数 p ，φ(p) = p -1 。
- 显然，1,2,...,p-1 均与 p 互质

- （2）对于素数 p ，φ(p^k) = p^k - p^(k-1)
- 一共 p^k 个数，去掉 p, 2p, 3p, ...,p^k 与 p^k 不互质，剩下的都与 p^k 互质

# 欧拉函数是积性函数

- 积性函数：如果当a，b互质时，有f(a*b)=f(a)*f(b)，那么称函数f为积性函数。
- 欧拉函数是积性函数，即：
- 若 p, q 互质(gcd(p, q) = 1) ，则有：φ(p*q) = φ(p)*φ(q) 。

- 特殊的，若p，q均为素数，则有：
- φ(p*q) = φ(p)*φ(q)=(p-1)*(q-1)

# 欧拉函数的通式

- $$\varphi(n) = n\prod_{i=1}^{k}(1-\frac{1}{pi})$$
- =n*(1-1/p1)*(1-1/p2)*(1-1/p3)*……*(1-1/pk)
- 其中p1, p2……pk为n的所有质因数。

- 证明：利用容斥原理。
- 设 p 是 n 的质因子，1~n 中 p 的倍数有 p,2p,3p,...,(n/p)p 共 n/p 个。
- 同理，若 q 也是 n 的质因子，则 1~n 中 q 的倍数有 n/q 个。
- 如果把这 n/p+n/q 个数去掉，那么 pq 的倍数被排除了两次，需要加回来一次。
- 因此，1~n 中不与 n 含有共同质因子 p 或 q 的数的个数为：n-n/p-n/q+n/(pq) = n(1-1/p-1/q+1/(pq))=n(1-1/p)(1-1/q)
- 类似地，可以在 N 的全部质因子上使用容斥原理，即可得到：
- φ(n)=n*(1-1/p1)*(1-1/p2)*(1-1/p3)*……*(1-1/pk)

- φ(n)=n*(1-1/p1)*(1-1/p2)*(1-1/p3)*……*(1-1/pk)

- 根据欧拉函数的通式，只需要分解质因数，即可求出欧拉函数

# 求 n 的欧拉函数

```
int euler(int n)
{
    int ans = n;
    for(int i = 2;i * i <= n;i++)//标准的分解质因子的模板
    {
        if(n % i == 0)
        {
            ans = ans / i * (i-1);
            //现在的i为质因子，根据公式(1-(1/i))，也就是乘以(i-1)/i
            while(n % i == 0)
                n /= i;
        }
    }
    if(n > 1)
        ans = ans / n * (n-1);
    return ans;
}
```

- 推论：
- 对于任意 n > 2，2 | φ(n)，因为必存在 pi -1 是偶数。

- 即：除了 1 和 2 以外，其他数的欧拉函数均为偶数。

# 欧拉函数的一些其他性质

- 如果 n % m == 0，那么 : φ(n * m) = m * φ(n)
- 如果 n % m == 0 && φ(n / m)%m == 0，那么 φ(n) = φ(n/m)*m
- 如果 n % m == 0 && φ(n / m)%m != 0，那么 φ(n) = φ(n/m)*(m-1)
- 当n为奇数时，φ(n) = φ(2*n)

- ∀n>1，1~n 中与n互质的数的和为 φ(n)*n/2

- 证明：
- 因为 gcd(n,x)=gcd(n,n-x)，所以与 n 不互质的数 x、n-x 成对出现，平均值为 n/2 。因此，与 n 互质的数的平均值也是 n/2，因此 1~n 中与 n 互质的数的和为 φ(n)*n/2

# n 的所有约数的欧拉函数和等于 n，即：$\sum_{d|n}\varphi(d)=n$

- 例如：
- n=10
- 10的约数有 1,2,5,10
- φ(1)+φ(2)+φ(5)+φ(10)=1+1+4+4=10
- 再如：
- n=20
- 20的约数有 1,2,4,5,10,20
- φ(1)+φ(2)+φ(4)+φ(5)+φ(10)+φ(20)=1+1+2+4+4+8=10

- 证明：
- (1)如果 n = 1
- φ(1)=1
- (2)如果n是质数
- ∑φ(d)=φ(n)+φ(1)=n−1+1=n
- (3)如果n是一个质数的幂,且底数 > 1
- 设n = p^k, p为质数, k > 1
- ∑φ(d)=∑φ(p^i)=p^k−p^(k−1)+p^(k−1)−p^(k−2)+……+p^1−1+1
- 可以发现中间有很多都消掉了，只剩下了
- ∑φ(d)=p^k

- (4)如果n有多个质因子
- 设 n = p1^a1×p2^a2×……×pk^ak
- 枚举每一个因子 d，换另一种表示形式：

$$\sum_{d|n}^{n}\varphi(d)=\sum_{i_1=0}^{a_1}\sum_{i_2=0}^{a_2}\cdots\cdots\sum_{i_k=0}^{a_k}\varphi(p_1^{i_1}p_2^{i_2}\cdots\cdots p_k^{i_k})$$

- 由于每个指数幂间互质，所以原式可以写成：

$$\sum_{d|n}^{n}\varphi(d)=\sum_{i_1=0}^{a_1}\sum_{i_2=0}^{a_2}\cdots\cdots\sum_{i_k=0}^{a_k}[\varphi(p_1^{i_1}p_2^{i_2}\cdots\cdots)\times\varphi(p_k^{i_k})]$$

$$\sum_{d|n}^{n}\varphi(d)=\sum_{i_1=0}^{a_1}\sum_{i_2=0}^{a_2}\cdots\cdots[\varphi(p_1^{i_1}p_2^{i_2}\cdots\cdots)\times\sum_{i_k=0}^{a_k}\varphi(p_k^{i_k})]$$

$$\sum_{d|n}^{n}\varphi(d)=\sum_{i_1=0}^{a_1}\sum_{i_2=0}^{a_2}\cdots\cdots\varphi(p_1^{i_1}p_2^{i_2}\cdots\cdots)\times p_k^{a_k}$$

$$\sum_{d|n}^{n}\varphi(d)=p_1^{a1}\times p_2^{a2}\times\cdots\cdots\times p_k^{a_k}=n$$

# 拓展

- 求1到n中所有数的欧拉函数值
- 如果一个一个求，时间复杂度为O(n*sqrt(n))，如果 n 比较大，复杂度还是挺高的。

- 如何优化？

```
int phi[n+1];
int euler(int n)
{
    for(int i = 1;i <= n;i++)
        phi[i] = i;
    for(int i = 2;i <= Max;i++)//Max就是代表最大值
        if(phi[i] == i)//表明了这是一个质数
            for(int j = i;j <= Max;j += i)
                phi[j] = phi[j]/i*(i-1);
}
```

# 或者

```
void init()
{
    phi[1]=1;
    for(int i=2;i<=N;i++){
        if(!st[i]){
            prime[cnt++]=i;
            phi[i]=i-1;
        }
        for(int j=0;prime[j]*i<=N;j++){
            st[prime[j]*i]=1;
            if(i%prime[j]==0){
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
```

# 例：POJ3090

- SDOI仪仗队