# 数学知识



## 质数

在大于 1 的整数中，如果只包含 1 和它本身这两个约数，就被称为质数，或者素数

1. **判定**

**试除法 ( O (n) )**

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

改进后 （O （sqrt n））

```
bool is_prime(int n)
{
    if(n < 2)   return false;
    for(int i = 2;i <= n/i ; i ++ )
        if(n % i == 0)
            return false;
    return true;
}
```

**筛素数法 （O n*ln n）**

现有数列 2,3,4,5,6,7,8,9,10,11,12 ......

现在枚举每个数，然后把每个数的倍数删掉

第一轮：2 , 3 ， ~~4~~，5，~~6~~，7，~~8~~，9，~~10~~，11 , ~~12~~ ......

第二轮：2，3，~~4~~，5，~~6~~，7，~~8~~，~~9~~，~~10~~，11，~~12~~ ......

第三轮：2，3，~~4~~，5，~~6~~，7，~~8~~，~~9~~，~~10~~，11，~~12~~ ......

......

n轮后，没有被删除的数说明在 2 ~ n 中没有它的倍数，即为素数

**但实际上我们只需要将每个质数的倍数的数删掉就行了**

```
int primes[N],cnt; //cnt 质数个数，primes[] 存的是质数
bool st[N];

void get_primes(int n)
{
    for(int i = 2; i <= n; i ++)
    {
        if(!st[i])
        {
            primes[cnt++] = i;
            for(int j = i + i; j <= n; j += i ) st[j] = true; //优化后
        }
    //for(int j = i + i; j <= n; j += i )   st[j] = true; 优化前
    }
}
```

**线性筛法 （O n)**

每次用最小质因数筛

```
int primes[N],cnt; //cnt 质数个数，primes[] 存的是质数
bool st[N];

void get_primes(int n)
{
    for(int i = 2; i <= n; i ++)
    {
        if(!st[i])  primes[cnt ++ ] = i;
        for(int j = 0; j < cnt && primes[j] <= n/i ; j ++ )
        {
            st[primes[j]*i]  = true;
            if(i % primes[j] == 0)  break; //执行此语句时primes[j]一定是i的最小质因子
        }                                   //break是线性的关键
    }
 }
```

一些题目：洛谷P3383 洛谷P3912 洛谷P1217

2. **分解质因数**

**把合数表示为质因数乘积的形式叫做“分解质因数”**

n 中最多只包含一个大于sqrt(n)的质因子，如果超过的话，相乘就会大于等于 n

例如： 18 分解质因数， $18 = 2 * 3 * 3 = 2^1 * 3^2$

O （n）代码

```
void divide(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(n%i == 0)
        {
            int s = 0;
            while(n%i == 0)
            {
                n/=i;
                s++;
            }
            printf("%d %d\n",i,s);
        }
    }
    puts("");
}
```

O （sqrt n）代码

```
void divide(int n) //分解质因数 n ， 并从小到大输出
{
    for (int i = 2; i <= n / i; i ++ ) //枚举到 sqrt（n）即可
        if (n % i == 0)
        {
            int s = 0;  //幂
            while (n % i == 0) n /= i, s ++ ;
            cout << i << ' ' << s << endl;
        }
    if (n > 1) cout << n << ' ' << 1 << endl; //说明此时的n就是目前大于sqrt(n)的质因数
    cout << endl;
}
```

一些题目：洛谷P2043 洛谷P1075 acwing 867

# 约数

**约数：如果一个自然数A能被自然数B整除，那么称B是A的约数。**

例如： 18 能被 1,2,3,6,9,18 整除，这些数都是18的约数

1. **试除法 （O sqrt n）**

```
vector<int> get_dividsors(int n)
{
    vector<int> res; //存放所有约数
    for(int i = 1; i <= n/i ; i++)
    {
        if(n%i == 0)
        {
            res.push_back(i);
            if(i != n/i)    res.push_back(n/i); //特判当 i*i == n 时，不需要重
复加入i
        }
    }
    return res;
}
```

2. **约数个数**

一个数 N 分解质因数为： $N = P_1^{a1} p_2^{a2} \ldots\ldots p_n^{an}$

则这个数的约数个数为 $(a_1+1)(a_2+1)(a_3+1)\ldots\ldots(a_n+1)$ ，P的指数的选法个数 相乘

```
unordered_map<int,int> primes; //哈希表
long long sum = 1;

void count_dividsor(int n)
{
for(int i = 2;i <= n/i; i++) //分解质因数
    {
        while(n%i == 0)
        {
            n/=i;
            primes[i]++; //该质因数出现的次数
        }
    }
    if(n > 1)   primes[n]++; //特判比较大的质因数
    for(auto item:primes)   sum = sum * (item.second + 1) ; //遍历哈希表 求出
答案
}
```

3. **约数之和**

与上面类似，一个数的约数之和为

$$(p_1^0+p_1^1+p_1^2+\ldots+p_1^{a_1})\ldots\ldots(p_n^0+p_n^1+p_n^2+\ldots\ldots+p_n^{a_n})$$

展开后就是一堆乘积，一共

$$(a1+1)(a2+1)(a3+1)\ldots\ldots(an+1)$$

个乘积，每个乘积都是一个约数

```
unordered_map<int,int> primes; //哈希表
long long sum = 1;

void sum_dividsor(int n)
```

```
{
    for(int i = 2;i <= n/i; i++) //分解质因数
    {
        while(n%i == 0)
        {
            n/=i;
            primes[i]++; //该质因数出现的次数
        }
    }
    if(n > 1)   primes[n]++; //特判比较大的质因数
    for(auto item:primes)
    {
        int p  = item.first , a = item.second;
        long long t;
        while(a --) t = (t * p + 1); //求 p1^0 + p1^1+p1^2+...+p1^a1
        sum = sum * t; //求乘积
    }
}
```

4. **欧几里得算法（辗转相除法） 求最大公约数**

```
int gcd(int a,int b) //求a,b两个数的最大公约数
{
    return b ? gcd(b,a % b) : a;
}
```

## 欧拉函数

1. **定义**

1 ~ N 中与 N 互质的数的个数被称为欧拉函数，记为$\phi(N)$。
若在算数基本定理中，$N = p_1^{a_1} p_2^{a_2} \dots p_m^{a_m}$，则：
$\phi(N) = N * \frac{p_1 - 1}{p_1} * \frac{p_2 - 1}{p_2} * \dots * \frac{p_m - 1}{p_m}$

即

$$\phi[N] = N * (1 - 1/p_1) * (1 - 1/p_2) * \dots * (1 - 1/p_m)$$

转化成

$$\phi[N] = N/p_1 * (p_1 - 1) * N/p_2 * (p_2 - 1) * \dots * N/p_m * (p_m - 1)$$

2. **代码实现**

```
int eulers(int n)  //O (sqrt n)
{
    int res = n; //先将结果初始化成n
    for (int i = 2; i <= n / i; i ++ ) //分解质因数
```

```
        if (n % i == 0)
        {
            res = res / i * (i - 1); //转化后的公式
            while (n % i == 0)
            {
                n /= i; //确保不会出现合数因子
            }
        }
    if (n > 1) res = res / n * (n - 1);

    return res;
}
```

### 3. 线性筛法求欧拉函数

对于 1 的欧拉函数 **phi[1]** 有定义：

$$phi[1] = 1$$

对于 **phi[pj * i]** 当 **i % pj = 0** 时，有如下性质:

$$phi[pj * i] = pj * phi[i]$$

当 **i % pj ≠ 0** 时，有如下性质：

$$phi[pj * i] = (pj - 1) * phi[i]$$

具体证明，请百度。。。

然后筛法与线性筛素数基本相同

```
int primes[N],phi[N],cnt;
bool st[N]; //phi存的就是下标的欧拉函数值，primes存的是素数

void get_eulers(int n)
{
    phi[1] = 1;
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++] = i;
            phi[i] = i-1; //质数与它前面的数互质
        }
        for(int j=0;j<cnt && primes[j]<=n/i;j++)
        {
            st[primes[j] * i] = true;
            if(i % primes[j] == 0)
            {
                phi[primes[j] * i] = primes[j] * phi[i];
                break; //线性的关键
            }
            else
                phi[primes[j] * i] = (primes[j] - 1) * phi[i];
        }
    }
}
```

4. **欧拉定理**

   对于**任意互质的**$a$和$n$，有 $a^{\phi(n)} \equiv 1(mod\ n)$

   例如， 5 和 6， $5^{\phi(6)} \equiv 1(mod\ 6)$ 即 $25 \equiv 1(mod\ 6)$

   当n为质数时， $a^{p-1} \equiv 1(mod\ p)$ （费马小定理）

## 快速幂

1. 快速幂顾名思义就是快速的求出某个数的幂

   假如要求一个数 A 的 n 次幂，我们可能会直接调用 pow 函数，但是 pow 函数是浮点运算，比整型运算慢，那么你可能会说直接套循环求，套循环求的话时间复杂度为 O (n)， 如果用快速幂求的话能把时间复杂度降低到 O (log n)

   快速幂的思路是把 $A^k$ 的指数拆分成 $k = 2^0 + 2^1 + ... + 2^{log\ k}$

   则 $A^k = A^{2^0+2^1+...+2^{log\ k}} = A^{2^0}A^{2^1}...A^{2^k}$

   先将 k 的二进制写出，比如 9 的二进制为 1001

   所以 9 拆分后为 $9 = 2^0 + 2^3$ ，那么 $3^9 = 3^{2^0+2^3} = 3^{2^0}3^{2^3}$ ，能大大的减少循环次数，只需要循环二进制的长度就可以求出来

```
int quick_power(int a,int k)
{
    int res = 1;
    while(k)
    {
        if(k & 1)    res = res * a;
        a = a*a; //更新A的值，第一次更新将A^(2^0)更新成A^(2^1),第二次更新成A^(2^2)以此类推
        k >>= 1;
    }
    return res;
}
```

## 扩展欧几里得算法

1. 扩展欧几里得算法故名意思就是欧几里得算法的扩展，先来看看欧几里得算法 (求最大公约数)

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

   先介绍一下**裴蜀定理**：

   对于任意的正整数 $a, b$,一定存在非零整数 $x, y$,使得 $ax + by = gcd(a, b)$

   扩展欧几里得算法就是用来求这对非零整数 $x, y$ 的

   实现：

当$b=0$时，显然 $x=1,y=0$ ，这种情况也就是$gcd(a,b)$开始出栈的时候，然后$gcd$递归时，将x , y位置调换，（追踪a,b），递归时：

$$by+(a\ mod\ b)x=gcd(a,b)$$

$$by+(a-(a/b)*b)x=gcd(a,b)$$

$$\text{其中（}a/b\text{）是向下取整的},\text{所以}a-(a/b)*b\neq 0,\text{而是}a\%b$$

$$ax+b(y-(a/b)*x)=gcd(a,b)$$

即 $y=y-(a/b)*x$ ，所以每次回溯时，更新 $y$ 的值 ，即 `y -= (a/b)*x` 这一步

对于$x,y$的所有取值 , 对 $ax+by=gcd(a,b)$变形

$$ax-\frac{ab}{d}+by+\frac{ab}{d}=gcd(a,b)$$
$$a(x-\frac{b}{d})+b(y+\frac{a}{d})=gcd(a,b)$$
$$\text{所以：}\ \ x=x_0-\frac{b}{d}k,y=y_0+\frac{a}{d}k\quad \text{其中}k\in Z$$

```
int exgcd(int a,int b,int &x,int &y)
{
 if(!b)
 {
     x = 1,y = 0;
     return a;
 }

 int d = exgcd(b,a % b,y,x); //注意 x, y 的位置
 y -= (a/b) * x;

 return d; //d 最大公约数
}
```

一些题目:同余方程 扩展欧几里得算法 乘法逆元

## 中国剩余定理

1. **两两互质的数** $m_1,m_2...m_k$

   有方程组（S）：

$$x\equiv a_1\ (mod\ m_1)$$
$$x\equiv a_2\ (mod\ m_2)$$
$$\vdots$$
$$x\equiv a_k\ (mod\ m_k)$$

设$M=m_1m_2...m_k$ ，$M_i=M/m_i$ ，$t_i=M_i^{-1}$ ， $M_i\ \ mod\ \ m_i$ 的**逆元**

则方程组 $S$ 的通解形式：$x=a_1t_1M_1+a_2t_2M_2+...+a_kt_kM_k$

模$M$的意义下，只有一个解：$x=(\ \sum_{i=1}^{k}a_iM_it_i\ )\ mod\ M$

通俗的将，就是有一个方程组取余的方程组，求未知数 $x$ 的定理

例如：

$$
\begin{aligned}
x\%3 = 1 \\
x\%5 = 1 \\
x\%7 = 1
\end{aligned}
$$

求未知数 $x$ ,这时就可以用中国剩余定理了,但是模上的数一定要**两两互质**

一些题目 [洛谷P1495中国剩余定理](#) [POJ 1006](#) [洛谷P4777](#)

## 高斯消元

高斯消元是用来就一组矩阵线性方程组的

例如：

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\
\cdots\cdots\cdots\cdots\cdots\cdots&\cdots\cdots\cdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m,
\end{aligned}
$$

由于还没学线性代数，所以我也不是很清楚，只知道一些大概操作

枚举每一列 c：

1. 找到当前列 c 的值的绝对值最大的行
2. 将这行换到最上面 （换好后第一列是按降序排列的）
3. 将该行整体系数缩小，其中第一个数 （a[i] [c] ）变成 1
4. 将下面所有行的当前列 c 消成 0
5. 你过来消源

直接来例题吧

```
输入一个包含n个方程n个未知数的线性方程组。

方程组中的系数为实数。

求解这个方程组。

下图为一个包含m个方程n个未知数的线性方程组示例：

9a504fc2d5628535be9dcb5f90ef76c6a7ef634a.gif

输入格式
第一行包含整数n。
```

接下来n行，每行包含n+1个实数，表示一个方程的n个系数以及等号右侧的常数。

输出格式
如果给定线性方程组存在唯一解，则输出共n行，其中第i行输出第i个未知数的解，结果保留两位小数。

如果给定线性方程组存在无数解，则输出“Infinite group solutions”。

如果给定线性方程组无解，则输出“No solution”。

数据范围
1≤n≤100,
所有输入系数以及常数均保留两位小数，绝对值均不超过100。

输入样例：
3
1.00 2.00 -1.00 -6.00
2.00 1.00 -3.00 -9.00
-1.00 -1.00 2.00 7.00
输出样例：
1.00
-2.00
3.00

```
~~~C++
#include <iostream>
#include <algorithm>
#include <cmath>

using namespace std;

const int N = 110;
const double eps = 1e-6;

int n;
double a[N][N];

int gauss() //高斯消元
{
    int c, r; //列行
    for (c = 0, r = 0; c < n; c ++ )
    {
        int t = r;
        for (int i = r; i < n; i ++ )
            if (fabs(a[i][c]) > fabs(a[t][c]))
                t = i;

        if (fabs(a[t][c]) < eps) continue; //如果当前列(c)的最大值的绝对值为  0  就跳过

        for (int i = c; i < n + 1; i ++ ) swap(a[t][i], a[r][i]); //交换行(r 和 t)
        for (int i = n; i >= c; i -- ) a[r][i] /= a[r][c];  //把当前行(r)第一个数(a[r][c])变成 1

        for (int i = r + 1; i < n; i ++ )
            if (fabs(a[i][c]) > eps)
```

```
                for (int j = n; j >= c; j -- )
                    a[i][j] -= a[r][j] * a[i][c]; //将当前列的值变成 0 ，
           //因为上一行当前列的值已经为 1 ，所以减去当前行的 c 倍的上一行，即可消为 0
        r ++ ;
    }

    if (r < n) //行没有枚举完 两种情况
    {
        for (int i = r; i < n; i ++ )  //枚举剩下没有枚举完的
            if (fabs(a[i][n]) > eps)   //如果有 b 出现大于 0，则无解
                return 2; //无解
        return 1; //无穷解
    }

    for (int i = n - 2; i >= 0; i -- )
        for (int j = i + 1; j < n; j ++ )
                  a[i][n] -= a[j][n] * a[i][j]; //倒着消元，
 //只需 将当前行(i) 的 b 减去后一行(j)的 b和当前行(i)(除a[i][i]外)每个系数的乘积

    return 0; //唯一解
}

int main()
{
    cin >> n;
    for (int i = 0; i < n; i ++ )
        for (int j = 0; j < n + 1; j ++ )
            cin >> a[i][j];

    int t = gauss();

    if (t == 0)
    {
        for (int i = 0; i < n; i ++ ) printf("%.2lf\n", a[i][n]); //a[i][n]即为解
    }
    else if (t == 1) puts("Infinite group solutions");
    else puts("No solution");

    return 0;
}
~~~
```

## 求组合数

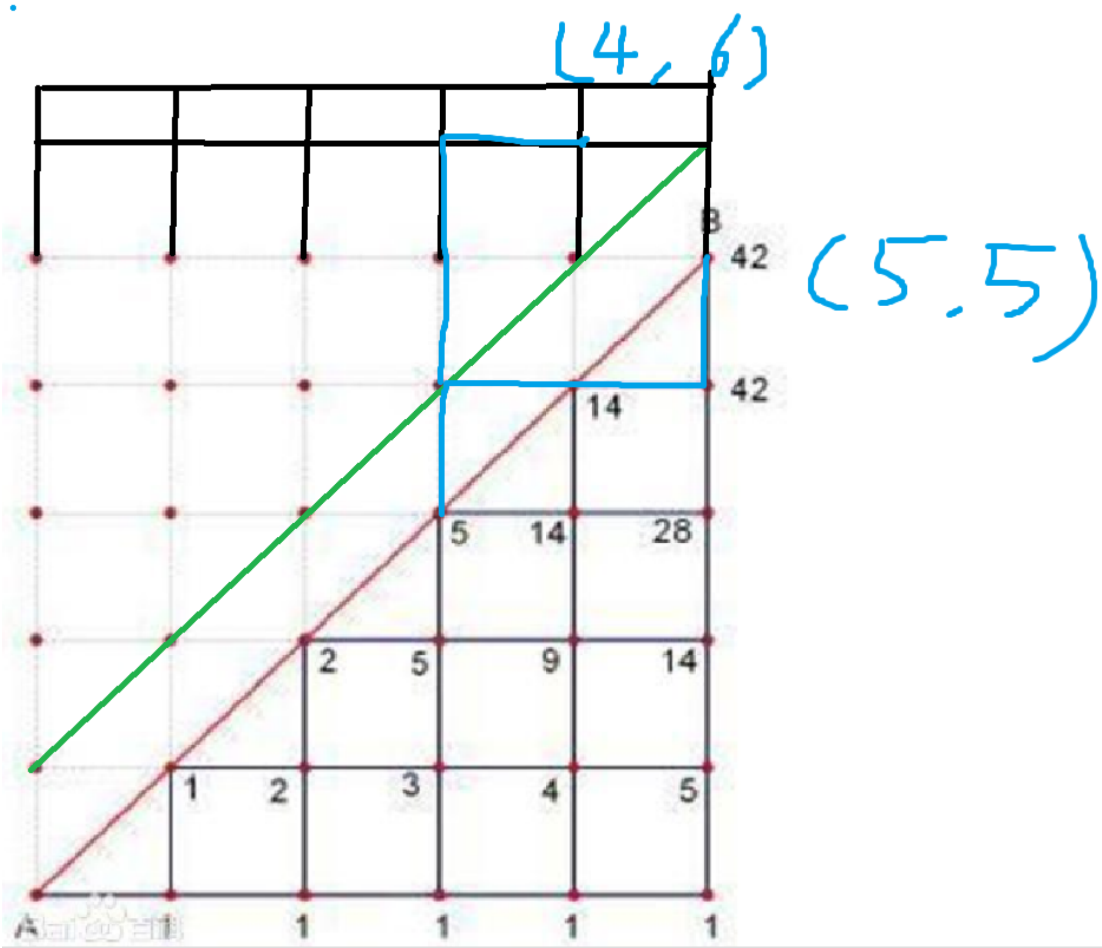
$C_a^b = \frac{a(a-1)\ldots(a-b+1)}{b!} = \frac{a!}{b!(a-b)!}$

在mod上某个数 p 时，除以这个数等于乘上这个数mod p 的逆元

$C_i^j = C_{i-1}^j + C_{i-1}^{j-1}$

## 卡特兰数

1.




从**A点**走到**B点**且**对于任意步数**，往右走的次数不少于往上走的次数的方案数共有多少?

这就是典型的卡特兰数，假设B点坐标为（n,n）从**A点**走到**B点**的方案数一共有$C_{2n}^{n}$种方案 ,而不合法的方案数（即往右走的次数不少于往上走的次数这个条件**不成立**）的方案数为$C_{2n}^{n-1}$

所以卡特兰数 $Catalan(n) = C_{2n}^{n} - C_{2n}^{n-1}$ **=** $\frac{C_{2n}^{n}}{n+1}$

如图，假设B(5,5)，那么经过绿线到达B点的方案数都是不合法的，对过绿线的路径做关于绿线的轴对称，那么经过绿线到达点B，就等效于从A点到达对称路径的终点( 图中为(4,6) )

所以方案数 **ans =** $Catalan(5) = C_{10}^{5} - C_{10}^{4} = 42$

**应用**： 求特定01序列，求合法括号序列，n个节点所能生成的二叉树数量，出栈序列，凸多边形三角划分

题目：[栈](#) [满足条件的01序列](#) [矩阵II](#) [生成字符串(变式)](#)