信息学竞赛中的数学方法

清华大学计算机系　胡泽聪

目录

基础数论

BASIC NUMBER THEORY

基本概念

- 带余除法
- 模数、取模
- 同余
- 因子
- 互质
- 最大公因数

模意义下的运算

很多题目的基础

加减乘法在模意义下封闭

除法则不同

费马小定理

$$a^{p-1} \equiv 1 \pmod p$$

要求p为质数，且a不是p的倍数。特别地，a可以为0。

证明思路：$a, 2a, 3a, \ldots, (p-1)a$。

欧拉定理

$$a^{\varphi(p)} \equiv 1 \pmod{p}$$

要求a与p互质。

$\varphi(x)$为欧拉函数，定义为小于等于x且与x互质的数的个数。

设$x=\prod p_i^{k_i}$，则

$$\varphi(x)=x\prod\left(1-\frac{1}{p_i}\right)=\prod(p_i-1)p_i^{k_i-1}。$$

对于质数而言，$\varphi(x)=x-1$。

乘法逆元

除法是乘法的逆运算。有$a \cdot \frac{1}{a} = 1$。

由欧拉定理，$a \cdot a^{\varphi(p)-1} \equiv 1 \pmod{p}$。

则$a^{\varphi(p)-1}$为a在模p意义下的乘法逆元，等价于$\frac{1}{a}$，记作a^{-1}。

当然，前提是a和p互质。

一些大质数

- $10^9 + 7$
- $10^9 + 9$
- $10^8 + 7$（不常用）
- 奇怪的生日

快速幂

由于模数通常很大，求逆元需要算的幂次也就很大。

朴素的O(指数)做法无法接受。

快速幂

考虑a^b，将b按二进制位分解。

如果我们知道$a^{2^0}, a^{2^1}, a^{2^2}, \dots$的话……

可以把b的所有为1的二进制位对应的次幂相乘得到a^b。

复杂度$O(\log b)$。

快速幂：代码

```
typedef long long ll;
inline int pw(int a, int b, int mod) {
    int r = 1;
    while (b > 0) {
        if (b & 1) r = (ll)r * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return r;
}
```

快速乘法

有时模数实在太大，以至于两数相乘无法用long long表示。

写高精度乘法显然不划算。

类似快速幂，处理出$a \cdot 2^0, a \cdot 2^1, a \cdot 2^2, \ldots$。

每次只要乘2，一般来说可以接受。

要是还存不下就没办法了。

最大公因数（GCD）

Greatest Common Divisor，记作$\gcd(a, b)$。

一些性质:

- $\gcd(a, 0) = a$
- $\gcd(a, b) = \gcd(a - b, b)$

更相减损术

利用$\gcd(a, b) = \gcd(a - b, b)$。

用大数减小数，得到新的一对a和b，并重复以上过程。

复杂度无法得到保证。

辗转相除法

假设$a > b$，那么更相减损术会一直使$a -= b$，直到$a < b$。

这其实就是$a \bmod b$。

于是就得到了辗转相除法。

复杂度为$O(\log \max(a, b))$。

辗转相除法： 代码

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

高精度加减乘法

用字符串表示数字，模拟竖式计算。

加减法复杂度$O(len)$，乘法复杂度$O(len^2)$，len为数字位数。

常用优化：压位。

高精度除法

类似竖式除法。

1. 在除数末尾补尽量多的0，使得其恰好小于被除数；
2. 进行数次高精度减法，直到被除数小于除数；
3. 令商加上减法次数；
4. 如果除数末尾还有之前补上的0，则除数除以0，商乘以10，跳转到第2步；
5. 剩下的被除数即为余数。

复杂度$O(len^2)$。

高精度GCD

辗转相除?

可能进行$O(len)$次除法运算，高精度除法太慢。

考虑更相减损，加入一些优化:

- 如果a和b都是偶数，直接令两数除以2，并令GCD乘以2;
- 如果a和b一奇一偶，则除去偶数的所有2的因子;
- 如果a和b都是奇数，则令大数减小数，此时必然得到一个偶数。

复杂度如何?

每两步就能令一个数除以2，故减法次数为$O(len)$。

总复杂度$O(len^2)$。

线性同余方程

形如$a \cdot x \equiv b \pmod{p}$，求$x$的值。

一些性质:

- 如果$\gcd(a,p) = 1$，则方程有且仅有一个解$x = a^{-1}b$;
- 方程有解$\Leftrightarrow \gcd(a,p) \mid b$，且解数为$\gcd(a,p)$。

而同余方程与不定方程$a \cdot x = p \cdot y + b$同解。

拓展欧几里得算法

拓展欧几里得算法在求出$\gcd(a,b)$的同时，还能顺带解出不定方程$ax+by=\gcd(a,b)$的一组解。

这个不定方程等价于线性同余方程$a\cdot x\equiv\gcd(a,b)\pmod b$，所以必然有解。

拓展欧几里得算法

考虑$a = 47, b = 30$的辗转相除过程：

- $47 = 30 \times 1 + 17$
- $30 = 17 \times 1 + 13$
- $17 = 13 \times 1 + 4$
- $13 = 4 \times 3 + 1$
- 得到$\gcd(47,30) = 1$

求出GCD之后，将整个过程倒过来：

- $1 = 13 + 4 \times (-3)$
- $4 = 17 + 13 \times (-1)$
- $13 = 30 + 17 \times (-1)$
- $17 = 47 + 30 \times (-1)$

拓展欧几里得算法

求出GCD之后，将整个过程倒过来：

- $1 = 13 + 4\times(-3)$
- $4 = 17 + 13\times(-1)$
- $13 = 30 + 17\times(-1)$
- $17 = 47 + 30\times(-1)$

依次带入得：

$$\begin{aligned}1 &= 13\times1 + \big(17 + 13\times(-1)\big)\times(-3)\\ &= 17\times(-3) + 13\times4\\ &= 17\times(-3) + \big(30 + 17\times(-1)\big)\times4\\ &= 30\times4 + 17\times(-7)\\ &= 30\times4 + \big(47 + 30\times(-1)\big)\times(-7)\\ &= 47\times(-7) + 30\times11\end{aligned}$$

解得$x = -7, y = 11$。

拓展欧几里得算法：代码

```c
int exgcd(int a, int b, int *x, int *y) {
    if (!b) {
        *x = 1, *y = 0;
        return a;
    }
    int r = exgcd(b, a % b, x, y);
    int t = *x;
    *x = *y;
    *y = t - a / b * (*y);
    return r;
}
```

求解线性同余方程

之前说到同余方程与不定方程$a \cdot x = p \cdot y + b$同解。

移项得$a \cdot x - p \cdot y = b$，而$\gcd(a, p) \mid b$。

1. 令$c = \frac{b}{\gcd(a,p)}$；
2. 使用拓展欧几里得算法求解$a \cdot x' + p \cdot y' = \gcd(\mathrm{a}, \mathrm{p})$的解$x', y'$；
3. 则原方程的解为$x = x' \cdot c$、$y = (-y') \cdot c$。

分解质因数

一个数n最多有一个大于$\sqrt{n}$的质因子。

1. 枚举所有不超过$\sqrt{n}$ 的质数并试除；
2. 如果剩下的$n \neq 1$，那么这也是一个质因子。

复杂度为$O(\sqrt{n})$。

枚举因子

一个数n的每个大于等于$\sqrt{n}$的因子x必然对应一个小于等于$\sqrt{n}$的因子n/x。

算法和分解质因数基本一样。

因子个数的上界$O(\sqrt{n})$，但实际上达不到这个上界。

筛法

预处理$1 \sim n$的所有质数。

1. 2是质数；
2. 对于每个质数，枚举其不超过n的所有倍数，标记为合数。

就这么简单。

复杂度为$\sum_{i=2}^{n} n/i = O(n \log n)$。

（调和级数$H_n = \sum_{i=1}^{n} 1/i \sim O(\log n)$）

筛法

我们还可以顺便处理出$1\sim n$内每个数的最小质因子。

有了这个信息之后，便可以在$O(\text{因子个数})=O(\log n)$的时间内分解质因数。

基础数论：例题

BASIC NUMBER THEORY: EXAMPLES

NOIP2012 D2T1 同余方程

题面

求$a \cdot x \equiv 1 \pmod{b}$的最小正整数解。 a和b互质。

$2 \le a, b \le 2 \cdot 10^9$。

题解

拓展欧几里得的直接应用。

也可以直接算逆元。

POJ1061 青蛙的约会

题面

假设赤道是长度为L的数轴，坐标为$[0, L-1]$的整数，跳过L之后会从0一侧跳出来。

两只青蛙分别在x和y的位置，每次分别可以跳m和n格。问最少跳几次之后，两只青蛙会相遇。

$1 \le x, y, m, n, L \le 2 \cdot 10^9$。

题解

其实就是求关于k的同余方程$x + km \equiv y + kn \pmod L$ 的最小非负整数解。

化一下即$(m-n)k \equiv y - x \pmod L$。判断是否有解之后解之即可。

HDU2824 The Euler Function

题意

求$\sum_{i=l}^{r}\varphi(i)$。

$1 \leq l \leq r \leq 10^6$。

（此处为原题削弱版）

题解

用筛法处理出每个数的最小质因子，并利用它来在$O(\log n)$时间内计算$\varphi(n)$的值。

POJ2480 Longge`s Problem

题意

求$\sum_{i=1}^{n}\gcd(i,n)$。

$1\le n<2^{31}$。

题解

转换思路：$\gcd(i,n)$的值肯定是n的因子。设其为d。

那有多少$i\in[1,n]$满足$\gcd(i,n)=d$?

同时除以d，即问有多少$i'\in\left[1,\frac{n}{d}\right]$满足$\gcd\left(i',\frac{n}{d}\right)=1$?

而这个个数就是$\varphi\left(\frac{n}{d}\right)$。

所以答案就是$\sum_{d|n}d\cdot\varphi\left(\frac{n}{d}\right)$。

注意在求$\varphi\left(\frac{n}{d}\right)$时应利用$n$的质因子分解结果。总复杂度约为$O(\sqrt{n}\log n)$。

SGU106
The Equation

题意

求$ax+by=c$在$x_1 \le x \le x_2$且$y_1 \le y \le y_2$的条件下有多少组解。

所有数字绝对值均不超过10^8。

题解

先考虑几种特殊情况：

1. $a=b=c=0$;
2. $a=b=0, c\neq 0$（无解）;
3. a和b中有且仅有一个0;
4. $d=\gcd(a,b)\nmid c$（无解）。

对于剩下的情况，先将a,b,c变为非负数，并求出一组可行解(x_0,y_0)。

之后便是求有多少k满足$x_1 \le x_0 + k\cdot\frac{b}{d} \le x_2$且$y_1 \le y_0 + k\cdot\frac{a}{d} \le y_2$，直接做除法取整即可。

POI2011 Strongbox

题意

有一个保险箱，其密码是$[0, n)$之内的整数，且如果a和b都是密码，那么$(a+b) \bmod n$也是密码。

已知$a_1, \dots, a_{k-1}$都不是密码，且a_k是密码。问保险箱最多有多少不同的密码。

$a_i, n \le 10^{14}$，$k \le 250000$。

题解

两个性质：

1. 如果a是密码，那么$\gcd(a, n)$是密码；
2. 如果a和b是密码，那么$\gcd(a, b)$是密码；
3. 密码的集合应当是某个整数s的所有倍数构成的集合。

为使密码尽可能多，应当使得s尽可能小。即寻找$\gcd(a_k, n)$的因子中，不是任意$\gcd(a_i, n)$因子的最小整数。

求出$\gcd(a_k, n)$的所有因子后，标记$\gcd(a_i, \gcd(a_k, n))$。所有有标记数字的因子均打上标记。没有被标记的最小整数即为s。

答案为n/s。

进一步了解……

- 中国剩余定理，用于求解线性同余方程组
- 线性筛法（Eratosthenes筛法），即复杂度为$O(n)$的筛法
- Miller-Rabin素数判定法，复杂度为$O(\log n)$的概率算法
- 离散对数，即求满足$a^x \equiv b \pmod p$的x
- 大步小步算法（Baby-step-giant-step，BSGS），用于求解离散对数

组合数学入门

INTRODUCTORY COMBINATORICS

基本计数原理

- 加法原理
- 乘法原理
- 减法原理

计数问题

统计满足某些条件的物体的个数。

例:

- 求项链的本质不同的染色方案数
- 求图的不同生成树的个数

答案通常很大，需要取模输出。

两个要点:

不重、不漏

排列与组合

n个人排成一排的方案数：

$$n! = 1 \cdot 2 \cdot \cdots \cdot n$$

从n个人中选出k个的方案数：

$$\frac{n \cdot (n-1) \cdot \cdots \cdot (n-k+1)}{k!} = \frac{n!}{k!\,(n-k)!}$$

也就是组合数，记作C_n^k或者$\binom{n}{k}$。

Pascal公式

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

即考虑最后一个人选还是不选。

平日所说的杨辉三角就是这个东西。

常用公式

多重集$\{n_1 \cdot a_1, n_2 \cdot a_2, \ldots, n_k \cdot a_k\}$的排列:

$$\binom{n}{n_1}\binom{n-n_1}{n_2}\binom{n-n_1-n_2}{n_3}\cdots\binom{n-\sum_{i=1}^{k-1} n_i}{n_k} = \frac{n!}{n_1!\,n_2!\cdots n_k!}$$

二项式定理:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

常用公式

从$n \times m$的网格的左上角走到右下角，每次只能向右或者向下走，行走路径的方案数：

$$\binom{n+m-2}{n-1}$$

方程$x_1 + \cdots + x_n = m$的非负整数解数：

$$\binom{m+n-1}{n-1}$$

常用公式

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

$$\binom{n}{0} + \binom{n}{2} + \cdots = \binom{n}{1} + \binom{n}{3} + \cdots$$

证明的两种方式

1. 组合学推理
2. 数学推导

尝试一下证明

1.

$$k\binom{n}{k} = n\binom{n-1}{k-1}$$

2. 范德蒙德（Vandermonde）卷积

$$\sum_{k=0}^{n}\binom{m_1}{k}\binom{m_2}{n-k} = \binom{m_1+m_2}{n}$$

模意义下的组合数

直接利用Pascal公式

- $n, k \leq 1000$
- 对模数没有要求

定义直接算逆元

- k较小
- 模数为质数

预处理阶乘逆元

- $n, k \leq 10^6$
- 模数为质数

Catalan数列

卡特兰数$C_n = \frac{1}{n+1}\binom{2n}{n}$

其含义为：n个+1和n个−1构成一个序列a，且任意前缀和≥ 0的方案数。

证明：

- 所有序列方案为$\binom{2n}{n}$；
- 不合法序列中存在一个最小的位置a_k，使得$a_1 + \cdots + a_k < 0$；
- 由于a_k最小，因此$a_1 + \cdots + a_{k-1} = 0$且$a_k = -1$；
- 将$a_1, \ldots, a_k$变为相反数，此时序列有$(n+1)$个+1和$(n-1)$个−1；
- 而不合法序列可以和有$(n+1)$个+1和$(n-1)$个−1的序列一一对应；
- 故方案数为$\binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n}$。

Catalan数列

卡特兰数的另一种表达方式是:

$$C_n = \sum_{i=1}^{n-1} C_i \cdot C_{n-i-1}$$

所以卡特兰数还是二叉树的方案数。

Bell数列

贝尔数B_n为将n个不同元素划分成任意份的方案数。

$$B_n = \sum_{i=0}^{n-1} \binom{n-1}{i} B_{n-1-i}$$

一般不需要计算，只是用来估计复杂度。

容斥原理

集合的补：$\bar{A} = S - A$

设$A_1, \mathrm{A}_2, \ldots, A_m$为集合$S$中具有一些性质的元素的集合，则

$$
\begin{aligned}
&|\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_m}| \\
&= |S| \\
&-\sum |A_i| \\
&+\sum \left|A_i \cap A_j\right| \\
&-\sum \left|A_i \cap A_j \cap A_k\right| \\
&+\cdots \\
&+(-1)^m \sum |A_1 \cap A_2 \cap \cdots \cap A_m|
\end{aligned}
$$

错位排列

满足i不在第i位的排列，记为D_n。

递推公式：

$$D_n = (n-1)(D_{n-1} + D_{n-2})$$

禁位排列

$n \times n$的棋盘上摆放n个车，互相不能攻击，方案数为$n!$。

如果有不能摆放的位置呢?

求出“至少有”$1 \sim n$个车处于禁位上的方案数，容斥原理。

（将“某个禁位上有车”看做一种属性）

组合数学入门：例题

INTRODUCTORY COMBINATORICS: EXAMPLES

一道数学题

题面

一个集合的子集个数为2^n，那一个集合的所有子集的子集个数之和是多少？

题解

一个集合的大小为k的子集个数为$\binom{n}{k}$。

那么答案为

$$\sum_{k=0}^{n}\binom{n}{k}2^k$$

$$=\sum_{k=0}^{n}\binom{n}{k}2^k1^{n-k}$$

$$=(2+1)^n=3^n$$

POJ3421 X-factor Chains

题意

给定正整数X，称其因数链为序列$X_1, X_2, \dots, X_m$，满足$X_1 = 1, X_m = X$，且$X_i | X_{i+1}\ (1 \le i < m)$。

求X的最长因数链长度与方案数。

$X \le 2^{20}$。

题解

对X做质因数分解，显然最长的方案一定是每次除以一个质因子。

而方案数就是质因子的多重集排列。

URAL1860 Fiborial

题面

定义序列F_n如下：

- $F_0 = 1$
- $F_1 = 1$
- $F_n = n \cdot F_{n-1} \cdot F_{n-2}\ (n \geq 2)$

求F_n的因子数。

$n \leq 10^6$。

题解

其实这题更多程度上是数论题。

分析得出$F_n = \prod_{i=2}^{n} i^{fib(n-i)}$。

利用筛法预处理，求出F_n的质因数分解。

假设各个质因子的次数为$k_1, k_2, \ldots, k_m$，因子个数是多少?

是$\prod(k_i + 1)$。

无名试题1

题面

集合S中有n个元素，你需要选出S的k个子集，满足所有这些子集的交集为空集。

求选择方案数。

$n, k \le 2^{63} - 1$。

题解

每个元素分开考虑。

这个元素不能处于所有k个子集中，但其他任意分配方案都可行。

所以这个元素的分配方案数为$2^k - 1$。

所以答案为$\left(2^k - 1\right)^n$。

《组合数学》习题8.5

题面

n个+1和m个−1，构成一个序列。求前缀和均≥ 0的序列的方案数。

$n \geq m$。

题解

$n = m$的情况即为卡特兰数，$n > m$的分析类似。

答案为:

$$\frac{n-m+1}{n+1}\binom{m+n}{m}$$

SKI

题面

$n \times m$的网格，每个格子有一个高度。一个人从任意网格开始滑雪，并在任意一个网格结束。只能从较高的网格滑向较低的网格。

问所有路径长度的$0 \sim k$次方的和。

$1 \le n, m \le 100$，$1 \le k \le 30$。

所有相邻网格高度不同。

题解

先考虑$k = 1$的情况，即为普通的DP。

假设我们有上一个状态的$0 \sim k$次方的答案，怎么转移到下一个状态?

二项式定理！

$$
\begin{aligned}
f_k' &= \sum_i (x_i + 1)^k \\
&= \sum_i \sum_j \binom{k}{j} x_i^j \\
&= \sum_j \binom{k}{j} \sum_i x_i^j \\
&= \sum_j \binom{k}{j} f_j
\end{aligned}
$$

进一步了解……

- 棋盘多项式，另一种求禁位排列的方法
- 矩阵乘法，可以快速计算一些简单的递推公式
- 快速阶乘，快速计算$n! \bmod p^k$，其中p为质数；可以用来计算大组合数取模，并可以配合中国剩余定理使用
- 置换、Burnside引理与Pölya计数法，用于计算考虑同构时的本质不同方案数
- Prüfer序列，用于唯一描述树的形态，并可用于计算满足一定条件的树的方案数

位运算与*bitset*

BITWISE OPERATIONS AND STD::BITSET

位运算

```
a & b   // bitwise and
a | b   // bitwise or
~a      // bitwise not
a ^ b   // bitwise xor
a << x  // bitwise shift left
a >> x  // bitwise shift right
```

集合的二进制表示

适用于全集大小比较小（通常在32以内）的情况。

用一个`unsigned int`表示一个子集。

二进制位为1代表子集中有这个元素，0代表没有。

集合操作

- 判断元素是否存在:
- 加入元素:
- 删除元素:
- 改变元素状态:
（如果存在则删除，否则加入）
- 与其他集合的交并异或
- 集合的补:
- 与其他集合的差:

```
return a >> pos & 1;

a |= 1U << pos;

a &= ~(1U << pos);

a ^= 1U << pos

a & b; a | b; a ^ b;

~a

a ^ (a & b)
```

枚举子集

```
int x;  // the set
for (int i = x; i > 0; i = (i - 1) & x) {
    // ...
}
```

std::bitset

二进制方式存储集合，支持任意位数。

支持上述所有集合操作。

对整个集合的操作时间复杂度均为$O(n/32)$。

std::bitset用例

```cpp
// ...
#include <bitset>

int main() {
    const int N = 1000;
    std::bitset<N> b;
    for (int i = 0; i < 5; ++i)
        b.set(i);
    b.reset(2);
    b |= (b << 3) & (b << 6);
    if (b[4]) b.flip(4);
    int cnt = b.count();
}
```

自己实现bitset

内部为unsigned int数组。

与或非异或：对所有数字进行位运算

左移右移：

```
void operator <<= (int pos) {
    int shift = pos >> 5, delta = pos & 31;
    for (int i = len - 1; i >= shift; --i) {
        unsigned int x = a[i - shift] << delta;
        if (delta > 0 && i > shift)
            x |= a[i - shift - 1] >> (32 - delta);
        a[i] = x;
    }
    for (int i = shift - 1; i >= 0; --i)
        a[i] = 0;
}
```

自己实现*bitset*

统计1的个数:

- 朴素$O(\log W)$方法
- 预处理+分段查表
- 分治$O(\log\log W)$

```
int count(unsigned int x) {
    x = ((x & 0xAAAAAAAAu) >> 1) + (x & 0x55555555u);
    x = ((x & 0xCCCCCCCCu) >> 2) + (x & 0x33333333u);
    x = ((x & 0xF0F0F0F0u) >> 4) + (x & 0x0F0F0F0Fu);
    x = ((x & 0xFF00FF00u) >> 8) + (x & 0x00FF00FFu);
    x = ((x & 0xFFFF0000u) >> 16) + (x & 0x0000FFFFu);
    return x;
}
```

bitset的简单应用

状态压缩动态规划（通常用单个int表示状态）。

存储值为bool类型的动态规划，如判断背包是否可行。

以0-1背包为例：

```
// traditional DP
bool f[N + 1];
f[0] = true;
for (int i = 1; i <= n; ++i)
    for (int j = size; j >= w[i]; --j)
        if (f[j - w[i]]) f[j] = true;

// using bitset
std::bitset<N + 1> g;
g.set(0);
for (int i = 1; i <= n; ++i)
    g |= g << w[i];
```

终

THE END