

算法 & 数学碎碎念

sun123zxy

2023-06-21^{*}

摘要 现场赛公式模板库，亦可作为小而精的总结性学习材料参考。无需单独成文或暂不完善的内容会放在这里。

目录

| | |
|----------------------------|----------|
| 1 数论 | 3 |
| 1.1 ExGCD | 3 |
| 1.2 CRT | 3 |
| ExCRT | 4 |
| 1.3 BSGS | 4 |
| 1.4 Miller-Rabin 【TODO】 | 4 |
| 1.5 Pollard-Rho 【TODO】 | 4 |
| 1.6 原根、Euler 定理等 | 4 |
| 1.7 Lucas 定理 | 5 |
| 1.8 Legendre 公式和 Kummer 定理 | 5 |
| 1.9 扩展 Euler 定理 | 6 |
| 1.10 Lagrange 定理 | 6 |
| 1.11 Wilson 定理 | 6 |
| 1.12 线性预处理 | 7 |
| 线性求逆元 | 7 |
| 线性阶乘逆 | 7 |
| 2 积性函数 | 7 |
| 2.1 整除分块 | 7 |
| 上取整整除分块 | 7 |
| 2.2 杜教筛 | 7 |
| 2.3 杂式 | 9 |

^{*}最后更新于 2024-03-10.

| | |
|--|-----------|
| 3 组合 | 9 |
| 3.1 容斥 / 二项式反演 | 9 |
| 形式一 | 9 |
| 形式二 | 9 |
| 应用 | 10 |
| 3.2 球盒问题 | 11 |
| 通用性质 | 12 |
| 3.3 第二类 Stirling 数 | 12 |
| 递推 | 12 |
| 第二类 Stirling 数同行计算: $\text{OGF}_n^{1,0,1}(x)$ 或 $\text{EGF}_n^{1,1,1}(x)$ | 12 |
| 第二类 Stirling 数同列计算: $\text{EGF}_r^{1,0,1}(x)$, $\text{EGF}_r^{1,1,0}(x)$ 与 $\text{EGF}^{1,0/\Sigma,1}(x)$ | 13 |
| Bell 数 | 13 |
| 第一类 Stirling 数、Stirling 数与阶乘幂 【TODO】 | 14 |
| 3.4 分拆数 【TODO】 | 14 |
| 3.5 背包计数 | 14 |
| 3.6 各种图计数 | 15 |
| 有(无)标号有(无)根树计数 【TODO】 | 15 |
| 有标号 DAG 计数 | 15 |
| 有标号偏序图计数 | 15 |
| 有标号连通图计数 | 15 |
| 竞赛图 | 15 |
| 有标号划分为 k 个全序图 | 15 |
| 有根树拓扑序计数 | 16 |
| DAG 拓扑序计数 | 16 |
| 无向图的色多项式 (chromatic polynomial) 和无环定向 (acyclic orientations) | 16 |
| 3.7 矩阵树定理 | 17 |
| 无向图的情形 | 17 |
| 有向图的情形 | 17 |
| 3.8 Polya 计数 | 18 |
| 3.9 杂数选提 | 18 |
| Catalan 数 | 18 |
| 3.10 代数组合 【TODO】 | 19 |
| q -analog | 19 |
| 有限域上的线性空间 | 19 |
| 有限群构型计数 | 19 |
| 4 多项式 | 19 |
| 4.1 通用 | 19 |
| 4.2 FFT / FNTT / 卷积 | 19 |
| 蝶形运算与迭代算法 | 20 |
| 4.3 多项式方程求解 (Newton 迭代法) 【TODO】 | 20 |
| 4.4 多项式求逆 | 21 |
| 倍增法一 (原创) | 21 |

| | |
|------------------------------|----|
| 1 数论 | 3 |
| 倍增法二 | 21 |
| 4.5 多项式开方 【TODO】 | 22 |
| 4.6 多项式 \ln | 22 |
| 4.7 多项式 \exp 【TODO】 | 22 |
| 4.8 多项式快速幂 | 22 |
| 5 集合幂级数 【TODO】 | 23 |
| 6 矩阵 | 23 |
| 6.1 矩阵乘法 | 23 |
| 6.2 矩阵快速幂 | 23 |
| 6.3 行列式 | 23 |
| 7 字符串 / 自动机 | 23 |
| 8 图论 【TODO】 | 24 |
| 8.1 最短路 | 24 |
| 8.2 强连通分量 | 24 |
| 8.3 网络流 | 24 |
| 9 杂项 | 24 |
| 9.1 模板 | 24 |
| 9.2 对拍 | 24 |
| Windows Batch | 24 |
| Linux Shell | 24 |
| 9.3 表 | 24 |
| 质数表 | 24 |
| 典列 | 25 |

前半部分主要为公式、推导、证明等速成提纲，大部分实现、模板、表格放在文末。

1 数论

1.1 ExGCD

定理 1.1 (ExGCD) 给定线性方程组 $ax + by = \gcd(a, b)$ ，其解可递归地由下式求得

$$ay_1 + b\left(x_1 - \left\lfloor \frac{a}{b} \right\rfloor y_1\right) = \gcd(b, a \bmod b)$$

其中 x_1, y_1 是 $bx + (a \bmod b)y = \gcd(b, a \bmod b)$ 的一组解。

1.2 CRT

定理 1.2 (CRT) 给定 n 个同余方程

$$x \equiv a_i \pmod{m_i} \quad (i = 1, \dots, n)$$

其中各 m_i 两两互质, 则上式等价于

$$x \equiv \sum_{i=1}^n a_i M_i \operatorname{inv}_{m_i}(M_i) \pmod{M}$$

其中 $M = \prod_{i=1}^n m_i$, $M_i = \frac{M}{m_i}$.

注记

$$M_i \operatorname{inv}_{m_i}(M_i) \pmod{m_j} = [i = j]$$

ExCRT

对一般的情况, 考虑合并两个同余方程. 给定 2 个同余方程

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \end{cases}$$

考虑化为不定方程形式

$$x = k_1 m_1 + a_1$$

$$x = k_2 m_2 + a_2$$

合并得到

$$k_1 m_1 + a_1 = k_2 m_2 + a_2$$

即

$$k_1 m_1 - k_2 m_2 = a_2 - a_1$$

此即关于 k_1, k_2 的不定方程. 若 $\gcd(m_1, m_2) \mid a_2 - a_1$, 则可应用 ExGCD 求得方程的一组解, 带回即得

$$x \equiv k_1 m_1 + a_1 \pmod{\operatorname{lcm}(m_1, m_2)}$$

否则同余方程组无解.

注记 若一组同余方程两两可合并, 则全部均可合并. 当判断大规模同余方程组是否有解时可能用到.

1.3 BSGS

求 $a^x \equiv b \pmod{m}$ 的一个特解, 其中 $\gcd(a, m) = 1$.

实质是非常暴力的根号分治. 根据 Euler 定理, 只需检测连续 $\varphi(m)$ 个 x 就可判定是否有解. 令 $x = q \lceil \sqrt{m} \rceil - r$, 其中 $q, r \in [1, \lceil \sqrt{m} \rceil]$, 于是 $x \in [0, \lceil \sqrt{m} \rceil^2]$. 代入原方程移项即得 $a^{q \lceil \sqrt{m} \rceil} \equiv ba^r \pmod{m}$, 右边使用 `map` 提前存下即可. 时间复杂度 $O(\sqrt{m})$.

1.4 Miller-Rabin 【TODO】

1.5 Pollard-Rho 【TODO】

1.6 原根、Euler 定理等

见 [FFT/NTT 讲稿](#).

1.7 Lucas 定理

定理 1.3 (Lucas 定理)

$$\binom{n}{m} \equiv \prod_i \binom{n_i}{m_i} \equiv \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \binom{n \bmod p}{m \bmod p} \pmod{p}$$

其中 p 是质数, n_i 和 m_i 是 n 和 m 的 p 进制表示下的各数位.

证明的要点有二: 一是 $\binom{p^i}{m} \bmod p = [m = 0 \wedge m = p^i]$, 即 $(1+x)^{p^i} \equiv 1 + x^{p^i} \pmod{p}$; 二是 Vandermonde 卷积按 p 进制拆分. 生成函数食用风味更佳.

参考:

- [Lucas's theorem - Wikipedia](#)
- [卢卡斯定理 - OI Wiki](#)

1.8 Legendre 公式和 Kummer 定理

定理 1.4 (Legendre 公式)

$$\nu_p(n!) = \sum_{i=1}^{+\infty} \left\lfloor \frac{n}{p^i} \right\rfloor = \frac{n - S_p(n)}{p-1}$$

其中 p 是质数, $S_p(n)$ 是 n 在 p 进制下各数位数字之和. $\nu_p(n!)$ 为 $n!$ 中质因子 p 的含量, 即所谓的 p -adic valuation of $n!$.

左侧显然, 按和式统计贡献即证. 对于右侧,

$$\begin{aligned} \sum_{i=1}^{+\infty} \left\lfloor \frac{n}{p^i} \right\rfloor &= \sum_{i=1}^{+\infty} \sum_{k=0}^{+\infty} n_{k+i} p^k \\ &= \sum_{j=1}^{+\infty} n_j \sum_{k=0}^{j-1} p^k \\ &= \sum_{j=1}^{+\infty} n_j \frac{p^j - 1}{p-1} \\ &= \frac{1}{p-1} \left(\sum_{j=1}^{+\infty} n_j p^j - \sum_{j=1}^{+\infty} n_j \right) \\ &= \frac{1}{p-1} \left(\sum_{j=0}^{+\infty} n_j p^j - \sum_{j=0}^{+\infty} n_j \right) \\ &= \frac{n - S_p(n)}{p-1} \end{aligned}$$

定理 1.5 (Kummer 定理) $\binom{a+b}{a}$ 中质因子 p 的含量等于 p 进制下加法 $a+b$ 发生的进位次数, 也可表示为

$$\nu_p \left(\binom{a+b}{a} \right) = \frac{S_p(a) + S_p(b) - S_p(a+b)}{p-1}$$

大力使用定理 1.4 考察 $\binom{a+b}{a}$ 的含 p 量:

$$\begin{aligned} \nu_p \left(\binom{a+b}{a} \right) &= \nu_p((a+b)!) - \nu_p(a!) - \nu_p(b!) \\ &= \sum_{k=1}^{\infty} \left(\left\lfloor \frac{a+b}{p^k} \right\rfloor - \left\lfloor \frac{a}{p^k} \right\rfloor - \left\lfloor \frac{b}{p^k} \right\rfloor \right) \end{aligned}$$

该公式恰好对 $a + b$ 在 p 进制计算过程中每一位发生的进位进行了求和. 定理的 S_p 形式容易通过定理 1.4 的另一形式得到.

研究该定理 $a + b = p^m$ 的特例. 用下降阶乘形式考虑二项式系数中 k 递增的过程, 注意到 $\nu_p(p^m - k) = \nu_p(k)$, 可以发现分子新增的 $n - k + 1$ 和分母新增的 k 恰好错一位, 于是

$$\nu_p \left(\binom{p^m}{k} \right) = m - \nu_p(k), \quad k = 1, 2, \dots, p^m$$

作为 Kummer 定理和上述特殊情况的推论, 我们有

$$\gcd_{k=1}^{n-1} \binom{n}{k} = \begin{cases} p & n = p^m \\ 1 & \text{otherwise} \end{cases}$$

其中 p 是某一质数. 这符合 Kummer 定理的直观描述. 得到 p^m 的加法必然会出现进位, 在加法 $p^{m-1} + (p-1)p^{m-1}$ 时达到最小进位次数 1; 否则在任一质数进制 p 下, 都可以取加法 $\text{lowbit}_p(n) + (n - \text{lowbit}_p(n))$ 使得不发生进位.

参考:

- [Legendre's formula - Wikipedia](#)
- [Kummer's theorem - Wikipedia](#)

1.9 扩展 Euler 定理

$$a^b \equiv a^{b \bmod \varphi(m) + \varphi(m)} \pmod{m}$$

进入循环所需步骤其实很少, 一定小于 $\varphi(m)$. (疑似量级在 $\log m$ 以下, 存疑, 见 FFT 讲稿)

1.10 Lagrange 定理

定理 1.6 (Lagrange 定理) 设 p 是质数, $A(x) \in \mathbb{Z}_p[x]$. 同余方程 $A(x) \equiv 0 \pmod{p}$ 只有至多 $\deg A(x)$ 个模 p 意义下不同的整数解, 除非这多项式的系数在模 p 意义下全为零.

这是域上的多项式理论在模 p 整数域上的应用.

1.11 Wilson 定理

定理 1.7 (Wilson 定理) 对质数 p ,

$$(p-1)! \equiv -1 \pmod{p}$$

$p = 2$ 容易特判证明. 现在只考虑 p 是奇质数的情况.

$(p-1)!$ 中, 互为逆元的数相互抵消, 只剩下逆元为自身的数 ± 1 , 立得上述定理.

另一种证法注意到 $x^{p-1} - 1 = \prod_{k=1}^{p-1} (x - k)$. 这是因为 Fermat 小定理指出 1 至 $p-1$ 的所有数的 $p-1$ 次幂均为 1, 而 Lagrange 定理又保证了多项式点值到系数映射的唯一性. 随后代入 $x = 0$ 立得结论.

参见 [Wilson's theorem - Wikipedia](#).

1.12 线性预处理

线性求逆元

现欲求出 a 模质数 p 意义下的逆元 a^{-1} . 用 a 对 p 做带余除法, $p = qa + r$, 于是 $-qa \equiv r \pmod{p}$. 两侧同时乘 r 的逆元 r^{-1} 得 $-qr^{-1}a \equiv 1 \pmod{p}$, 故

$$a^{-1} = -qr^{-1} = -\left\lfloor \frac{p}{a} \right\rfloor \cdot (p \bmod a)^{-1}$$

线性阶乘逆

$$\frac{1}{n!} = (n+1) \cdot \frac{1}{(n+1)!}$$

2 积性函数

2.1 整除分块

```
ll ans=0;
for(ll l=1,r,d;l<=N;l=r+1){
    d=N/l, r=N/d;
    ans+=(S_mu(r)-S_mu(l-1))*d;
}
```

此 $O(\sqrt{n})$ 较满, 极大劣于因子个数的 $O(\sqrt{n})$.

另有变种枚举 $\lfloor \frac{n}{d^2} \rfloor$ 的整除分块如下, 复杂度为 $O(n^{1/3})$.

```
ll ans=0;
for(ll l=1,r,d;l*l<=N;l=r+1){
    d=N/(l*l), r=sqrt(N/d);
    ans+=(S_mu(r)-S_mu(l-1))*d;
}
```

上取整整除分块

```
ll cdiv(ll a,ll b){ //ceil(a/b)
    return (a<0||a%b==0)?a/b:a/b+1;
}
ll ans=0;
for(ll l,r=N,d;r>=1;r=l-1){
    d=cdiv(N,r), l=cdiv(N,d);
    ans+=(S_mu(r)-S_mu(l-1))*d;
}
```

2.2 杜教筛

设 f 为一数论函数, 我们希望快速求得其前缀和 $\hat{f}(n) = \sum_{i=1}^n f(i)$. 考虑数论函数 g 和 $h = g * f$,

$$h(n) = \sum_{d|n} g(d)f\left(\frac{n}{d}\right)$$

两端做前缀和得

$$\begin{aligned}
 \hat{h}(n) &= \sum_{i=1}^n h(i) \\
 &= \sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right) \\
 &= \sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\
 &= \sum_{d=1}^n g(d) \hat{f}\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \\
 &= g(1) \hat{f}(n) + \sum_{d=2}^n g(d) \hat{f}\left(\left\lfloor \frac{n}{d} \right\rfloor\right)
 \end{aligned}$$

因此

$$\hat{f}(n) = \frac{1}{g(1)} \left(\hat{h}(n) - \sum_{d=2}^n g(d) \hat{f}\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \right)$$

故若 g 、 h 的前缀和可 $O(1)$ 算得，根据上式整除分块即可递归地计算出 f 的前缀和。预处理前 $O(n^{2/3})$ 项并记忆化得到的时间复杂度为 $O(n^{2/3})$ 。外层整除分块不会增加时间复杂度。

关于时间复杂度证明可参考 [sun123zxy's blog - OI 数论中的上界估计与时间复杂度证明](#) # 杜教筛。

$$f = \mu, g = I, h = \varepsilon$$

$$f = \varphi, g = I, h = \text{id}$$

$$f = \text{id}^k \mu, g = \text{id}^k, h = \varepsilon$$

$$f = \text{id}^k \varphi, g = \text{id}^k, h = \text{id}^{k+1}$$

注记 杜教筛的这种化法事实上也是 Eratosthenes 筛法的应用。一般的我们有

$$\sum_{d=1}^n f(d) \hat{g}\left(\left\lfloor \frac{n}{d} \right\rfloor\right) = \sum_{i=1}^n \sum_{d|i} f(d) g\left(\frac{i}{d}\right) = \sum_{i=1}^n \sum_{d|i} f\left(\frac{i}{d}\right) g(d) = \sum_{d=1}^n g(d) \hat{f}\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

```

unordered_map<ll,ll> s_mu;
ll S_mu(ll n){
    if(n<=MXG){
        return mu[n]; // already accumulated
    }else if(s_mu.count(n)){
        return s_mu[n];
    }
    ll ans=0;
    for(ll l=2,r,d;l<=n;l=r+1){
        d=n/l,r=n/d;
        ans+=S_mu(d)*(r-l+1);
    }
    return s_mu[n]=1-ans;
}

```


2.3 杂式

无平方因子数计数:

$$\sum_{i=1}^n \mu^2(i) = \sum_{i=1}^{\lfloor \sqrt{n} \rfloor} \mu(i) \left\lfloor \frac{n}{i^2} \right\rfloor$$

约数个数函数的一个性质:

$$d(ab) = \sum_{x|a} \sum_{y|b} [\gcd(x, y) = 1]$$

$$d(abc) = \sum_{x|a} \sum_{y|b} \sum_{z|c} [\gcd(x, y) = 1][\gcd(y, z) = 1][\gcd(x, z) = 1]$$

广义约数个数函数性质扩展:

$$\sigma_k(ab) = \sum_{x|a} \sum_{y|b} [\gcd(x, y) = 1] \left(x \frac{b}{y}\right)^k = \sum_{x|a} \sum_{y|b} [\gcd(x, \frac{b}{y}) = 1] (xy)^k$$

$$\sigma_k(abc) = \sum_{x|a} \sum_{y|b} \sum_{z|c} [\gcd(x, \frac{b}{y}) = 1][\gcd(y, \frac{c}{z}) = 1][\gcd(x, \frac{c}{z} = 1)] (xyz)^k$$

3 组合

3.1 容斥 / 二项式反演

形式一

容斥原理的第一种形式给出了“子集和变换”的逆变换.

定理 3.1 (容斥原理, 形式一, 集合)

$$g(S) = \sum_{T \subset S} f(T) \iff f(S) = \sum_{T \subset S} (-1)^{|S|-|T|} g(T)$$

证明的关键是 $\sum_{k=0}^n \binom{n}{k} (-1)^k = (1-1)^n = [n=0]$.

定理 3.2 (容斥原理, 形式一, 二项式反演)

$$g(n) = \sum_{k=0}^n \binom{n}{k} f(k) \iff f(n) = \sum_{k=0}^n (-1)^{n-k} \binom{n}{k} g(k)$$

若设 $F(x)$ 和 $G(x)$ 分别为 $f(n)$ 和 $g(n)$ 的指数生成函数 (EGF), 则结论可等价地表示为

$$G(x) = e^x F(x) \iff F(x) = e^{-x} G(x)$$

生成函数的形式使我们可在 $O(n \log n)$ 的优秀时间复杂度之内在 $f(n)$ 和 $g(n)$ 间做出变换.

形式二

形式一的补集形式, 给出了全集 U 下“超集和变换”的逆变换.

定理 3.3 (容斥原理, 形式二, 集合)

$$g(S) = \sum_{S \subset T \subset U} f(T) \iff f(S) = \sum_{S \subset T \subset U} (-1)^{|T|-|S|} g(T)$$

定理 3.4 (容斥原理, 形式二, 二项式反演)

$$g(n) = \sum_{k=0}^{N-n} \binom{N-n}{k} f(n+k) \iff f(n) = \sum_{k=0}^{N-n} (-1)^k \binom{N-n}{k} g(n+k)$$

应用

例 3.1 (不太常见的“容斥原理”) 满足全部性质的元素数量可容斥地通过下式计算

$$\begin{aligned} \text{card} \left(\bigcap_{i \in U} A_i \right) &= \sum_{k=0}^{|U|} (-1)^{|U|-k} \sum_{|S|=k} \text{card} \left(A - \bigcup_{i \in U-S} A_i \right) \\ &= \sum_{k=0}^{|U|} (-1)^{|U|-k} \sum_{|S|=k} \text{card} \left(\bigcup_{i \in S} A_i \right) \\ &= \sum_{k=0}^{|U|} (-1)^{|U|-k} \sum_{i_1 < \dots < i_k} \text{card} (A_{i_1} \cup \dots \cup A_{i_k}) \end{aligned}$$

其中 A 代表全集, A_i 代表满足第 i 个性质的元素构成的集合, U 是非空有限的性质指标集.

证明 令 $f(S)$ 为恰好只满足 S 中各性质的元素数量, $g(S)$ 为至多只满足 S 中各性质的元素数量, 即

$$\begin{aligned} f(S) &:= \text{card} \left(\left(A \cap \bigcap_{i \in S} A_i \right) - \bigcup_{i \in U-S} A_i \right) \\ g(S) &:= \text{card} \left(A - \bigcup_{i \in U-S} A_i \right) = \text{card} \left(A - \bigcup_{i \in U} A_i \right) + \text{card} \left(\bigcup_{i \in S} A_i \right) \end{aligned}$$

取 $S = U$ 代入定理 3.1 右侧就得到结论第一行的等式. 继续化简

$$\begin{aligned} \text{card} \left(\bigcap_{i \in U} A_i \right) &= \sum_{k=0}^{|U|} (-1)^{|U|-k} \sum_{|S|=k} \text{card} \left(A - \bigcup_{i \in U-S} A_i \right) \\ &= \sum_{k=0}^{|U|} (-1)^{|U|-k} \sum_{|S|=k} \left(\text{card} \left(A - \bigcup_{i \in U} A_i \right) + \text{card} \left(\bigcup_{i \in S} A_i \right) \right) \\ &= \text{card} \left(A - \bigcup_{i \in U} A_i \right) \sum_{k=0}^{|U|} (-1)^{|U|-k} \binom{|U|}{k} + \sum_{k=0}^{|U|} (-1)^{|U|-k} \sum_{|S|=k} \text{card} \left(\bigcup_{i \in S} A_i \right) \end{aligned}$$

注意到

$$\sum_{k=0}^{|U|} (-1)^{|U|-k} \binom{|U|}{k} = [U] = 0$$

而 $U \neq \emptyset$, 故上式左项为 0, 即得结论式第二行. \square

例 3.2 (有点常见的“容斥原理”) 不满足任何性质的元素数量可容斥地通过下式计算

$$\begin{aligned} \text{card} \left(A - \bigcup_{i \in U} A_i \right) &= \sum_{k=0}^{|U|} (-1)^k \sum_{|S|=k} \text{card} \left(A \cap \bigcap_{i \in S} A_i \right) \\ &= \text{card} A + \sum_{k=1}^{|U|} (-1)^k \sum_{|S|=k} \text{card} \left(\bigcap_{i \in S} A_i \right) \\ &= \text{card} A + \sum_{k=1}^{|U|} (-1)^k \sum_{i_1 < \dots < i_k} \text{card} (A_{i_1} \cap \dots \cap A_{i_k}) \end{aligned}$$

其中 A 代表全集, A_i 代表满足第 i 个性质的元素构成的集合, U 是非空有限的性质指标集.

证明 令 $f(S)$ 为恰好只满足 S 中各性质的元素数量, $g(S)$ 为至少满足 S 中各性质的元素数量, 即

$$f(S) := \text{card} \left(\left(A \cap \bigcap_{i \in S} A_i \right) - \bigcup_{i \in U-S} A_i \right)$$

$$g(S) := \text{card} \left(A \cap \bigcap_{i \in S} A_i \right) = \begin{cases} \text{card } A & S = \emptyset \\ \text{card} \left(\bigcap_{i \in S} A_i \right) & \text{otherwise} \end{cases}$$

取 $S = \emptyset$ 代入定理 3.3 右侧就得到结论. \square

习题 3.1 (错排) 计算 n 元错排的数量.

解 设 A_i 表示第 i 个位置配对正确的置换构成的集合. 直接应用例 3.2 立得

$$\begin{aligned} \text{card} \left(A - \bigcup_{i \in U} A_i \right) &= \sum_{k=0}^{|U|} (-1)^k \sum_{|S|=k} \text{card} \left(A \cap \bigcap_{i \in S} A_i \right) \\ &= \sum_{k=0}^{|U|} (-1)^k \binom{n}{k} (n-k)! \\ &= n! \sum_{k=0}^{|U|} (-1)^k \frac{1}{k!} \sim \frac{n!}{e} \end{aligned}$$

这说明随机取一排列, 其错排的概率趋近于 $\frac{1}{e}$.

例 3.3 (常见的“容斥原理”) 满足至少 1 个性质的元素数量可容斥地通过下式计算

$$\begin{aligned} \text{card} \left(\bigcup_{i \in U} A_i \right) &= \sum_{k=1}^{|U|} (-1)^{k-1} \sum_{|S|=k} \text{card} \left(\bigcap_{i \in S} A_i \right) \\ &= \sum_{k=1}^{|U|} (-1)^{k-1} \sum_{i_1 < \dots < i_k} \text{card} (A_{i_1} \cap \dots \cap A_{i_k}) \end{aligned}$$

其中 A_i 代表满足第 i 个性质的元素构成的集合, $i \in U$.

证明 对例 3.2 做简单移项即得. \square

3.2 球盒问题

组合数学的万恶之源.

以后我们约定:

- 形如 $\text{BB}_{1,1,0}$ 的记号表示代号 1, 1, 0 对应球盒问题的方案数;
- 形如 $\text{EGF}_n^{1,1,0}(x)$ 代表代号 1, 1, 0 对应 n 球球盒问题的指数生成函数;
- 形如 $\text{EGF}_r^{1,1,0}(x)$ 代表代号 1, 1, 0 对应 r 球球盒问题的指数生成函数;
- 普通生成函数 $\text{OGF}_r^{1,1,0}(x)$ 同理.

在不至混淆的情况下, 也可省略下标或上标上的代号.

通用性质

命题 3.1 (非空盒数量不限制)

$$\begin{aligned} \text{BB}_{*,*,\Sigma,1}(n) &= \sum_{k=0}^n \text{BB}_{*,*,1}(n, k) \\ \text{EGF}^{*,*,\Sigma,1}(x) &= \sum_{k=0}^{+\infty} \text{EGF}_{r=k}^{*,*,1}(x) \end{aligned}$$

命题 3.2 (有标号盒空置)

$$\begin{aligned} \text{BB}_{*,1,0}(n, r) &= \sum_{k=0}^r \binom{r}{k} \text{BB}_{*,1,1}(n, k) \\ \text{EGF}_n^{*,1,0}(x) &= e^x \text{EGF}_n^{*,1,1}(x) \end{aligned}$$

对有标号盒的二项式反演. 在 $1, 1, 0$ -球盒问题中, 上式体现为第二类 Stirling 数的通项公式; 在 $0, 1, 0$ -球盒问题中, 上式体现为 Vandermonde 卷积.

命题 3.3 (无标号盒空置)

$$\begin{aligned} \text{BB}_{*,0,0}(n, r) &= \sum_{k=0}^r \text{BB}_{*,0,1}(n, k) \\ \text{OGF}_n^{*,0,0}(x) &= \frac{1}{1-x} \text{OGF}_n^{*,0,1}(x) \end{aligned}$$

对无标号盒, 直接求和即可.

3.3 第二类 Stirling 数

第二类 Stirling 数的组合定义即 $1, 0, 1$ -球盒问题的方案数 $\text{BB}_{1,0,1}(n, r) = \left\{ \begin{smallmatrix} n \\ r \end{smallmatrix} \right\}$, 亦作将 n 个元素划分入 r 个集合的方案数.

递推

由组合意义, 考虑在已有 n 个球时加入新球, 此时面临将其放入原有的 r 个集合或新开辟一个集合的两种选择, 由此得递推式

$$\left\{ \begin{smallmatrix} n+1 \\ r \end{smallmatrix} \right\} = r \left\{ \begin{smallmatrix} n \\ r \end{smallmatrix} \right\} + \left\{ \begin{smallmatrix} n \\ r-1 \end{smallmatrix} \right\}$$

边界在

$$\left\{ \begin{smallmatrix} n \\ 0 \end{smallmatrix} \right\} = [n=0], \quad \left\{ \begin{smallmatrix} n \\ n \end{smallmatrix} \right\} = 1$$

第二类 Stirling 数同行计算: $\text{OGF}_n^{1,0,1}(x)$ 或 $\text{EGF}_n^{1,1,1}(x)$

来考虑用两种方法得到 $1, 1, 1$ -球盒问题的方案数.

我们的第一种方法注意到, 上述问题的方案数恰好是 $1, 0, 1$ -球盒问题方案数 $r!$ 倍——这是对后者的盒进行标号的结果. 我们得到

$$\text{BB}_{1,1,1}(n, r) = r! \text{BB}_{1,0,1}(n, r) = r! \left\{ \begin{smallmatrix} n \\ r \end{smallmatrix} \right\}$$

第二种方法考虑用容斥非空盒的方法与 $1, 1, 0$ -球盒问题 $BB_{1,1,0}(n, r) = r^n$ 建立联系. 我们有

$$BB_{1,1,0}(n, r) = \sum_{k=0}^r \binom{r}{k} BB_{1,1,1}(n, k)$$

即

$$r^n = \sum_{k=0}^r \binom{r}{k} k! \left\{ \begin{matrix} n \\ k \end{matrix} \right\}$$

二项式反演即得

$$r! \left\{ \begin{matrix} n \\ r \end{matrix} \right\} = \sum_{k=0}^r (-1)^{r-k} \binom{r}{k} k^n$$

即

$$\left\{ \begin{matrix} n \\ r \end{matrix} \right\} = \sum_{k=0}^r \frac{(-1)^{r-k} k^n}{k! (r-k)!} = \sum_{k=0}^r \frac{k^n}{k!} \cdot \frac{(-1)^{r-k}}{(r-k)!}$$

这正是第二类 Stirling 数的通项公式. 注意到其具有卷积的形式, 由此可快速计算出同一行的第二类 Stirling 数. 事实上, 二项式反演的生成函数形式已向我们道尽一切

$$OGF_n^{1,0,1}(x) = EGF_n^{1,1,1}(x) = e^{-x} EGF_n^{1,1,0}(x) = e^{-x} \sum_{k=0}^{+\infty} k^n \frac{x^k}{k!}$$

第二类 Stirling 数同列计算: $EGF_r^{1,0,1}(x)$, $EGF_r^{1,1,0}(x)$ 与 $EGF^{1,0/\Sigma,1}(x)$

为快速计算同一列的第二类 Stirling 数, 考虑用生成函数的思路构造 $1, 1, 0$ -球盒问题和 $1, 0, 1$ -球盒问题的方案数. 前者将大小为 n 的有标号集合划分为 r 个有标号等价类, 这相当于有序拼接 r 个非空有标号集合; 后者则是前者除掉 $r!$ 消序的版本. 写成生成函数即

$$\begin{aligned} EGF_r^{1,1,0}(x) &= (e^x - 1)^r \\ EGF_r^{1,0,1}(x) &= \frac{(e^x - 1)^r}{r!} \end{aligned} \tag{1}$$

Bell 数

Bell 数 $B(n)$ 的组合定义即 $1, 0/\Sigma, 1$ -球盒问题的方案数 $BB_{1,0/\Sigma,1}$, 亦可描述为 n 元集合上等价关系 (划分) 的数量.

注意到 $1, 0, 1$ -球盒问题和 $1, 0/\Sigma, 1$ -球盒问题的关系

$$BB_{1,0/\Sigma,1}(n) = \sum_{r=0}^n BB_{1,0,1}(n, r) = \sum_{r=0}^{+\infty} BB_{1,0,1}(n, r)$$

我们首先有

$$B(n) = \sum_{r=0}^n \left\{ \begin{matrix} n \\ r \end{matrix} \right\}$$

其次, 根据式 1 写出此关系的 EGF 形式就有

$$EGF^{1,0/\Sigma,1}(x) = \sum_{r=0}^{+\infty} EGF_r^{1,0,1}(x) = \sum_{r=0}^{+\infty} \frac{(e^x - 1)^r}{r!} = e^{e^x - 1}$$

我们得到 Bell 数的可用于快速计算的 EGF.

注记 有标号集合的无序划分是指数型生成函数的一类重要应用。例如，有标号连通图计数可视为对有标号一般图的无序划分，故有标号连通图的 EGF $G(x)$ 和有标号一般图的 EGF $F(x)$ 间有着 $e^{G(x)} = F(x)$ 的关系。

Bell 数还有递推形式

$$B(n+1) = \sum_{k=0}^n \binom{n}{k} B(k)$$

边界 $B(0) = 1$ 。组合的解释是，枚举第 $n+1$ 个元素被放入的集合的大小，再对除该集合之外的元素进行划分。从指数生成函数的角度来看，设 $B(n)$ 的 EGF 为 $F(x)$ ，由 EGF 的移位性质和二项式反演的 EGF 形式，上式等价于 $F'(x) = e^x F(x)$ ，解此微分方程也能得到 Bell 数的生成函数。

第一类 Stirling 数、Stirling 数与阶乘幂 【TODO】

更多参考：

- [Stirling number - Wikipedia](#)
- [斯特林数 - OI Wiki](#)

3.4 分拆数 【TODO】

关于 k 部分拆数，

By taking conjugates, the number $p_k(n)$ of partitions of n into exactly k parts is equal to the number of partitions of n in which the largest part has size k .

- [Partition \(number theory\) - Wikipedia](#)
- [分拆数 - OI Wiki](#)
- [组合数学（2）分拆数 - 知乎](#)

3.5 背包计数

通式：

$$\prod_{i=1}^n (1 + s_i x^{v_i})^{m_i} = \exp \sum_{i=1}^n m_i \ln(1 + s_i x^{v_i}) = \exp \sum_{i=1}^n m_i \sum_{k=1}^{+\infty} (-1)^{k-1} \frac{s_i^k}{k} x^{kv_i}$$

其中 v_i 互不相同（相同者体现在 m_i 上）。后者可以埃筛调和级数计算贡献 + 多项式 \exp 地在 $O(t \log t)$ （这里的 t 指我们所关心的体积上限）内快速计算。

下面问题的 OGF 都可化归至通式，从而 $O(t \log t)$ 地得到计算。

设有 n 种可区分的物品，体积分别为 v_i 。当每种物品只有一件时，方案数 OGF 为

$$\prod_{i=1}^n (1 + x^{v_i})$$

当每种物品有无限件时，方案数 OGF 为

$$\prod_{i=1}^n (1 + x^{v_i} + x^{2v_i} + \dots) = \prod_{i=1}^n (1 - x^{v_i})^{-1}$$

当每种物品分别有 c_i 件时，方案数 OGF 为

$$\prod_{i=1}^n (1 + x^{v_i} + x^{2v_i} + \dots + x^{c_i v_i}) = \prod_{i=1}^n \frac{1 - x^{(c_i+1)v_i}}{1 - x^{v_i}} = \prod_{i=1}^n (1 - x^{(c_i+1)v_i}) \prod_{i=1}^n (1 - x^{v_i})^{-1}$$

ex: 普通的最优化背包也有卷积视角的理解，见 [Knapsack, Subset Sum and the \(max,+\) Convolution - Codeforces](#).

3.6 各种图计数

有（无）标号有（无）根树计数 **【TODO】**

有标号 DAG 计数

$$f_n = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f_{n-i}$$

思路是对 DAG 的入度为零的点做容斥。进一步推导可拆出卷积形式，再用类似分治 FFT 的生成函数方法可得封闭形式。

- [Wikipedia](#)
- [OEIS](#)
- [OI-Wiki](#)
- [cjyyb 题解](#)

有标号偏序图计数

问得好，但这是个著名的 open problem. 各种类型的偏序图计数参考 [Partially ordered set - Wikipedia # Number of partial orders](#).

- [Stack Exchange](#)
- [OEIS](#)
- [Erné, M., Stege, K. Counting finite posets and topologies. Order 8, 247–265 \(1991\)](#) (内有研究历史综述)

有标号连通图计数

标准的有标号无序划分。EGF 是有标号一般图计数 EGF 的 \ln 。

竞赛图

强连通的竞赛图一定存在 Hamilton 回路（归纳证明）；无环的竞赛图是全序图。两者结合可推出竞赛图一定存在 Hamilton 路径。同时，强连通竞赛图中存在所有大小的环路。

有标号划分为 k 个全序图

Lah 数

$$L(n, k) = \frac{n!}{k!} \binom{n-1}{k-1}$$

思路是先 $n!$ 排个大序，再无标号球入非空有标号桶，再除 $k!$ 给桶消序。

生成函数的思路是，考虑非空全序计数的 EGF 为 $\frac{1}{1-x} - 1 = \frac{x}{1-x}$ ， k 次有序拼接再消序即得

$$\sum_{n \geq k} L(n, k) \frac{x^n}{n!} = \frac{1}{k!} \left(\frac{x}{1-x} \right)^k$$

更多信息参考 Wikipedia.

有根树拓扑序计数

对于外向树， $\frac{n!}{\prod_{u \in V} \text{size}(u)}$ ，其中 V 是所有节点的集合， n 是树大小， $\text{size}(u)$ 是以 u 为根的子树大小。

树形 DP 风格的归纳证明是可以的。下面的链接中提供了一个有趣的组合理解，其思路是从全序出发，逐步按照拓扑序的要求对每个节点下的子树消序。

题目常见要求对每个节点作为根节点求出方案数，换根 DP 即可。

注意到每个合法的外向树拓扑序 reverse 后立刻与内向树拓扑序形成一一对应，故内向树拓扑序计数与外向树相同。

- [\[Insight\] Number of Topological Orderings of a Directed Tree - Codeforces](#)

DAG 拓扑序计数

对一般的 DAG 上的拓扑序计数，其本质上是偏序上线性扩张的计数问题，这被证明是 #P-complete 的问题（总之就是很困难）。

- [How many topological orderings exist for a graph? - Mathematics Stack Exchange](#)
- [Topological sorting - Wikipedia # Relation to partial orders](#)
- [Linear extension - Wikipedia](#)

无向图的色多项式 (chromatic polynomial) 和无环定向 (acyclic orientations)

色多项式是对图的 k -colorings 的数量在 $k = 0, 1, \dots, n$ 进行 Lagrange 插值后得到的多项式。 $k > n$ 的 k -colorings 的数量也可通过在色多项式的 $x = k$ 处求值得到。这一证明主要依赖所谓的 deletion-contraction 递推关系式。

对一般的图而言，色多项式的大部分系数和求值问题都是“NP”相关的，但在一些特殊的图上有好的形式：

- 完全图 K_n : x^n (下降阶乘幂)
- 无边图 \overline{K}_n : x^n
- 链 / 树: P_n : $x(x-1)^{n-1}$
- 环: $(x-1)^n + (-1)^n(x-1)$

多个连通分量拼接时，色多项式满足乘法性。

关于图的无环定向的方案数，Richard Stanley 在一篇 1973 年的论文中证明其恰为图的色多项式在 -1 处的取值。

- [Chromatic Polynomial - from Wolfram MathWorld](#)
- [Chromatic polynomial - Wikipedia](#)
- [Orientation \(graph theory\) - Wikipedia](#)
- Stanley, R. P. “Acyclic Orientations of Graphs.” Disc. Math. 5, 171-178, 1973.

3.7 矩阵树定理

无向图的情形

对无向图，度数矩阵 $D = \text{diag}\{\deg(i)\}$ ，邻接矩阵 A 定义为

$$A_{i,j} = \begin{cases} 0 & i = j \\ e(i,j) & i \neq j \end{cases}$$

其中 $e(i,j)$ 表示点 i 到点 j 的边的数量（对无向图， $e(i,j) = e(j,i)$ ）。

定义 Laplace 矩阵（Kirchhoff 矩阵） $L = D - A$ 。

Laplace 矩阵有性质 $L = BB^T$ ，其中关联矩阵 B 按如下方式定义

$$B_{i,j} = \begin{cases} 1 & \text{node } i \text{ is the ID-smaller endpoint of edge } j \\ -1 & \text{node } i \text{ is the ID-larger endpoint of edge } j \\ 0 & \text{otherwise} \end{cases}$$

这里 1 与 -1 的引入完成了一种对边的“手动定向”，其用途将在后文介绍。

定理 3.5（矩阵树定理，无向图） n 点无向图的生成树的个数与该图的 Laplace 矩阵 L 的任意主子式的值相等。其也与 L 所有 $n-1$ 个非零特征值乘积的 $\frac{1}{n}$ 倍相等。

Laplace 矩阵同行的代数余子式均相等（这性质由行和为 0 得到），因此去掉任意一行一列均可得到正确的无向图生成树计数。此外，由于 $L = BB^T$ 至少半正定， L 的所有特征值非负。

证明的要点在于对 $L = BB^T$ 的某个主子式（一般选择去掉第一行第一列）应用 Cauchy-Binet 公式，随后说明行列式的组合意义中，环的情况一定相互抵消。关于特征值的结论可从特征多项式、各 $n-1$ 阶主子式与韦达定理的关系中得到（依此方法能进一步得到有关 k -生成森林的一些结论）。

事实上，去掉第 i 行第 i 列，即是统计以 i 为根的根向生成树的数量。先前定义关联矩阵 B 时“手动定向”，是为了使换向过程中环的情况相互抵消，只留下树的唯一一种情况。当然，因为是无向图，这里树的朝向和根的具体位置并不重要。

有向图的情形

对有向图，我们明确统计的对象为根向（或叶向）生成树的数量。根向树形图与出度 Laplace 矩阵相关， $L^{out} = D^{out} - A$ ，其中 D^{out} 是出度矩阵。

为体现有向图的要求，出度 Laplace 矩阵对应的关联矩阵需要一些修改。令矩阵 B 满足

$$B_{i,j} = \begin{cases} 1 & \text{node } i \text{ is the head of edge } j \\ 0 & \text{otherwise} \end{cases}$$

矩阵 C 满足

$$C_{i,j} = \begin{cases} 1 & \text{node } i \text{ is the head of edge } j \\ -1 & \text{node } i \text{ is the tail of edge } j \\ 0 & \text{otherwise} \end{cases}$$

则出度 Laplace 矩阵满足性质 $L^{out} = BC^T$ 。我们构造的矩阵 B 体现了对出边方向的要求，在此基础上矩阵 C 进一步完成了“手动定向”的工作。

定理 3.6 (矩阵树定理, 根向树形图) n 点有向图以 i 为根的生成根向树形图的数量与该图出度 Laplace 矩阵 L^{out} 去掉第 i 行第 i 列的 $n-1$ 阶主子式的值相等. 该有向图的所有生成根向树形图的数量也与 L^{out} 的所有 $n-1$ 个非零特征值的乘积相等.

由于 L^{out} 的行和仍为 0, 其同行代数余子式仍然相等.

关于叶向树形图, 我们有类似的结论:

定理 3.7 (矩阵树定理, 叶向树形图) n 点有向图以 i 为根的生成叶向树形图的数量与该图入度 Laplace 矩阵 L^{in} 去掉第 i 行第 i 列的 $n-1$ 阶主子式的值相等. 该有向图的所有生成根向树形图的数量也与 L^{in} 的所有 $n-1$ 个非零特征值的乘积相等.

由于 L^{in} 的列和 (而非行和) 为 0, 其同列 (而非同行) 代数余子式均相等.

更多内容, 参考

- [Laplacian matrix - Wikipedia](#)
- [矩阵树定理 - OI Wiki](#)
- [Kirchhoff's theorem - Wikipedia](#)

3.8 Polya 计数

定理 3.8 (Burnside)

$$= \frac{1}{|G|} \sum_{f \in G} |C(f)|$$

其中 G 是等价操作群, $C(f)$ 是操作 f 下的不动点集合.

3.9 杂数选提

Catalan 数

$$C_n = \binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1} \binom{2n}{n} = \prod_{k=2}^n \frac{n+k}{k}$$

Segner's recurrence relation:

$$C_0 = 1; \quad C_{n+1} = \sum_{i=0}^n C_i C_{n-i}$$

OGF:

$$A(z) = 1 + zA^2(z) \implies A(z) = \frac{1 - \sqrt{1 - 4z}}{2z}$$

注意下面组合意义间的前后逻辑顺序.

- number of full binary tree with n interior nodes / $n+1$ leaves / $2n+1$ nodes
在数内点意义下可以导出 Catalan 数的生成函数.
- number of ways of associating n binary operators / parenthesizing $n+1$ factors completely
- number of proper parenthesis sequences with n pairs of parentheses

【TODO】 暂时不知道怎么从上面推过来的. 另一种理解是长度为 n 的出栈序列的数量.

- number of ordered trees with $n + 1$ vertices

最外层要补一对括号.

- number of binary trees with n vertices

by left-child right-sibling encoding of ordered trees. 最后删去只有左儿子的根.

该组合意义可以导出 Catalan 数的生成函数.

Catalan 数的组合意义并不止如此几种. cf. [Wikipedia](#)

3.10 代数组合【TODO】

q -analog

有限域上的线性空间

有限群构型计数

4 多项式

此部分详细介绍请移步 [FFT/NTT 讲稿](#).

4.1 通用

4.2 FFT / FNTT / 卷积

- DFT: (本原) 单位根构造 $\omega_n = e^{\frac{2\pi}{n}i}$.
- NTT: $P = 998244353 = 7 \times 17 \times 2^{23} + 1$, PR = 3 是它的一个原根. (本原) 单位根构造 $\omega_n = \text{PR}^{\frac{P-1}{n}} \bmod P$.
- $P = 1004535809 = 479 \times 2^{21} + 1$, PR = 3
- $P = 469762049 = 7 \times 2^{26} + 1$, PR = 3

考虑将待变换多项式

$$A(x) = \sum_{k=0}^{2n-1} a_k x^k$$

奇偶分项为两个多项式

$$A(x) = A_0(x^2) + xA_1(x^2)$$

其中

$$A_0(x) = \sum_{k=0}^{n-1} a_{2k} x^k$$

$$A_1(x) = \sum_{k=0}^{n-1} a_{2k+1} x^k$$

代入 $x = \omega_{2n}^k$ ($k = 0, \dots, 2n-1$), 用单位根消去 / 折半性质 ($\omega_{2n}^{2k} = \omega_n^k$)

$$A(\omega_{2n}^k) = A_0(\omega_n^k) + \omega_{2n}^k A_1(\omega_n^k)$$

用 $\omega_{2n}^{n+k} = -\omega_{2n}^k$

$$\begin{aligned} A(\omega_{2n}^k) &= A_0(\omega_n^k) + \omega_{2n}^k A_1(\omega_n^k) \\ A(\omega_{2n}^{n+k}) &= A_0(\omega_n^k) - \omega_{2n}^k A_1(\omega_n^k) \end{aligned} \quad (k = 0, \dots, n-1)$$

即得 FFT/FNTT 递归算法. 用单位根求和性质

$$\frac{1}{n} \sum_{k=0}^{n-1} \omega^{ik} = [i \mid n]$$

可知 DFT/NTT 变换矩阵 $F = \left(\omega_n^{ij} \right)_{(i,j) \in n \times n}$ 满足 $FF^H = F^H F = nI_n$, 故 $F^{-1} = \frac{1}{n} F^H = \left(\frac{1}{n} \omega_n^{-ij} \right)_{(i,j) \in n \times n}$, 此即 DFT/NTT 逆变换矩阵.

蝶形运算与迭代算法

为使用迭代算法, 需要快速得到递归算法向下深入过程中 $\{a_n\}$ 置换后的最终结果. 观察知该置换是位逆序置换, 可按如下方法线性求出.

```
rev[0]=0; for(ll i=1;i<(1<<n);i++) rev[i]=(rev[i>>1]>>1)+((i&1)<<(n-1));
```

4.3 多项式方程求解 (Newton 迭代法) 【TODO】

给定一多项式 $A(x)$, 求解满足 $A(B(x)) = 0 \pmod{x^n}$ 的多项式 $B(x)$.

显然 $B(x)$ 只有前 n 项有效. 目前我们尚不清楚解的存在性、唯一性等性质, 但注意到

$$A(B(x)) = 0 \pmod{x^{2n}} \implies A(B(x)) = 0 \pmod{x^n}$$

故考虑递推求解. 首先, 边界条件 $A(b_0) = 0$ 需要单独求解. 在确定某一 b_0 的基础上, 我们开始递推. 考虑已经获得 $A(B(x)) = 0 \pmod{x^n}$ 的一个解 $B(x) = B_0(x)$, 下面尝试得到方程 $A(B(x)) = 0 \pmod{x^{2n}}$ 的解.

将待求解方程 $A(B(x)) = 0 \pmod{x^{2n}}$ 左式多项式 A 在 B_0 处 Taylor 展开

$$\begin{aligned} A(B(x)) &= A(B_0(x)) + A'(B_0(x))(B(x) - B_0(x)) \\ &\quad + \frac{A''(B_0(x))}{2!}(B(x) - B_0(x))^2 + \dots = 0 \pmod{x^{2n}} \end{aligned}$$

假若 $B(x) - B_0(x) = 0 \pmod{x^n}$, 那么模 x^{2n} 意义下二次方以上的项可以舍去, 上式等价于

$$A(B_0(x)) + A'(B_0(x))(B(x) - B_0(x)) = 0 \pmod{x^{2n}}$$

移项即可解出待求 $B(x)$

$$B(x) = B_0(x) - \frac{A(B_0(x))}{A'(B_0(x))} \pmod{x^{2n}}$$

这里要求 $A'(B_0(x))$ 需在 x^{2n} 意义下可逆, 即

$$\begin{aligned} [x^0]A'(B_0(x)) &= [x^0]A'(b_0) = \sum_{k=0}^{\infty} ([x^k]A') b_0^k \\ &= \sum_{k=0}^{\infty} (k+1)a_{k+1}b_0^k \neq 0 \pmod{x^{2n}} \end{aligned}$$

注意到 $B(x) - B_0(x) = -\frac{A(B_0(x))}{A'(B_0(x))} = 0 \pmod{x^n}$, 故满足上述“假若”的解存在且唯一. 然而, 若不要求这一“假若”成立, 则每步迭代解的唯一性无法得到保证. 考虑到边界条件 b_0 的解亦不一定唯一, 故一般的 Newton 迭代法解唯一性的讨论较为复杂. 当然, 上述推导至少为我们提供了一种寻找特解的方法.

Newton 迭代法作为通用求解框架, 可涵盖几乎所有多项式初等运算. 【TODO】

4.4 多项式求逆

给定一多项式 $A(x)$, 求解满足 $A(x)B(x) = 1 \pmod{x^n}$ 的多项式 $B(x)$.

多项式逆元存在的充分必要条件是其实常数项非零 (这是因为边界条件 $b_0 = \frac{1}{a_0}$), 若存在则在模意义下一定唯一. 这结论可直接由下述求解方法得到. 不失一般性, 我们只研究 $A(x)$ 的次数为奇数 $2n - 1$ 的情况. 设

$$A(x) = A_0(x) + x^n A_1(x)$$

$$B(x) = B_0(x) + x^n B_1(x)$$

下述两种方法均递归地在已知

$$A(x)B_0(x) = A_0(x)B_0(x) = 1 \pmod{x^n}$$

的基础上求解 $A(x)$ 的逆元. 时间复杂度均为

$$T(n) = T\left(\frac{n}{2}\right) + O(n \log n) = O(n \log n)$$

以下简记 $A(x)$ 为 A , 其它多项式同理.

倍增法一 (原创)

$$\begin{aligned} AB &= 1 \pmod{x^{2n}} \\ \iff (A_0 + x^n A_1)(B_0 + x^n B_1) &= 1 \pmod{x^{2n}} \\ \iff x^n(A_1 B_0 + A_0 B_1) + A_0 B_0 &= 1 \pmod{x^{2n}} \\ \iff A_1 B_0 + A_0 B_1 + \left\lfloor \frac{A_0 B_0}{x^n} \right\rfloor &= 0 \pmod{x^n} \\ \iff A_0 B_1 = -\left\lfloor \frac{A_0 B_0}{x^n} \right\rfloor - A_1 B_0 &\pmod{x^n} \\ \iff B_1 = -B_0 \left(\left\lfloor \frac{A_0 B_0}{x^n} \right\rfloor + A_1 B_0 \right) &\pmod{x^n} \end{aligned}$$

常数偏大, 这里就不放代码了.

倍增法二

注意到

$$\begin{cases} AB_0 = 1 \pmod{x^n} \\ AB = 1 \pmod{x^n} \end{cases} \implies A(B - B_0) = 0 \pmod{x^n}$$

由于 A 的常数项非零, 故

$$B - B_0 = 0 \pmod{x^n}$$

(这证明了逆元在不同模数下的前缀保持一致)

两边平方得

$$B^2 - 2BB_0 + B_0^2 = 0 \pmod{x^{2n}}$$

两侧同乘 A 并移项得

$$B = 2B_0 - AB_0^2 \pmod{x^{2n}}$$

4.5 多项式开方【TODO】

和多项式求逆类似的推导可得递推方程

$$B = \frac{1}{2} \left(\frac{A}{B_0} + B_0 \right) \pmod{x^{2n}}$$

有一些和 Newton 法一样麻烦的边界条件讨论，也会出现复杂的多解情况。 $a_0 = 1$ 时 $b_0 = \pm 1$ ，按 $b_0 = 1$ 的实现如下。

亦可 $\sqrt{A} = \exp(\frac{1}{2} \ln A)$ ，此法可处理多项式任意幂指数运算。

4.6 多项式 \ln

给定一多项式 $A(x)$ ，求解满足 $B(x) = \ln A(x) \pmod{x^n}$ 的多项式 $B(x)$ 。

次数为 $+\infty$ 的多项式 \ln 存在的充分必要条件为其常数项非零（这是因为边界条件 $b_0 = \ln a_0$ ），同样一旦存在则唯一。注意到仅整数 $a_0 = 1$ 时， $\ln a_0$ 可取得整数，故合理的 a_0 只能是

1. 另一种解释参见[多项式初等函数 - OI Wiki](#) # [多项式对数函数 & 指数函数](#)。

推导是容易的。方程两侧同时求导得

$$B'(x) = \frac{A'(x)}{A(x)} \pmod{x^{n-1}}$$

两侧再积分得

$$B(x) = \int \frac{A'(x)}{A(x)} dx + C \pmod{x^n}$$

其中 $C = \ln a_0$ 。多项式求逆、求导、积分即可。时间复杂度 $O(n \log n)$ 。

4.7 多项式 \exp 【TODO】

Newton 迭代法可推出

$$B = B(1 - \ln B_0 + A) \pmod{x^{2n}}$$

时间复杂度

$$T(n) = T\left(\frac{n}{2}\right) + O(n \log n) = O(n \log n)$$

存在的充要条件是 $a_0 = 0$ 。唯一性证明暂不明确。

4.8 多项式快速幂

普通的多项式快速幂实现当然是 $O(n \log n \log k)$ 的。下面介绍基于指对数性质的 $O(n \log n)$ 求法。

对常数项 $a_0 = 1$ 的 $n-1$ 次多项式 $A(x)$ ，

$$A^k(x) = e^{k \ln A(x)}$$

我们指出，在系数对质数 p 取模的意义下，当我们关心的多项式长度 $n \leq p$ 时，有

$$A^p(x) \equiv a_0 \equiv 1 \pmod{p}$$

这是因为

$$(a+b)^p \equiv a^p + b^p \pmod{p}$$

故

$$A^p(x) = (a_0 + xA_1(x))^p \equiv a_0^p + x^p A_1^p(x) \pmod{p}$$

由费马小定理， $a_0^p = a_0$ ，而 $n \leq p$ 表明 $x^p A_1^p(x)$ 一项可被忽略，故上述结论得到证明。这些讨论可用于处理幂指数 $k \geq p$ 的情况。

一般的，当常数项非 1 时，为满足多项式 \ln 的要求，设多项式 $A(x)$ 的最低次项为 $a_t x^t$ ，则

$$A^k(x) = (a_t x^t)^k \left(\frac{A(x)}{a_t x^t} \right)^k$$

右侧的多项式常数项归一，故可再应用上述方法计算。

关于多项式，更代数的内容参考 [Formal power series - Wikipedia](#)

5 集合幂级数 【TODO】

6 矩阵

6.1 矩阵乘法

普通的 $O(n^3)$ 实现。

6.2 矩阵快速幂

普通的 $O(n^3 \log k)$ 实现。

6.3 行列式

普通的实现是使用逆元进行高斯消元，可用于域上的线性空间。若求解逆元的时间复杂度为 $O(\log p)$ ，则时间复杂度为 $O(n^3 + n^2 \log p)$ 。

这里给出另一种做法。该做法在消去时使用辗转相除法，可用于任意 Euclid 整环 (Euclidean domain，有带余除法的无零因子的交换幺环) 上的模 (环上的线性空间)，且时间复杂度不会增加。一个常见的应用是模 m 整数环 \mathbb{Z}_m 上的行列式求值，其中 m 不是质数。

消去各目标行第 c 列元素时，以第 c 行的 $a_{c,c}$ 为除数与目标行的第 c 列元素辗转相除，最终使 $a_{c,c}$ 变为 0，再做一次行交换将其换至目标行，就完成了一次行消去过程。注意到当 $a_{c,c}$ 非零时，一次行消去操作结束后 $a_{c,c}$ 单调不减，且过程中 $a_{c,c}$ 不会从非零变为零，故辗转相除带来的 $\log p$ 次额外操作开销被分摊到整轮对第 c 列的消去过程中，因此时间复杂度仍为 $O(n^3 + n^2 \log p)$ 。

7 字符串 / 自动机

基于 DFA 理论，[OI-Wiki](#) 上有对 KMP / AC 自动机，SAM / GSAM 和 PAM 简明扼要的概括。[OI-Wiki](#) 的后缀自动机讲解亦值得参考。

8 图论【TODO】

8.1 最短路

8.2 强连通分量

8.3 网络流

9 杂项

9.1 模板

见 [model.cpp](#), [model_temp.cpp](#).

9.2 对拍

Windows Batch

```
:loop
    gen.exe>dat.in
    my.exe<dat.in>my.out
    std.exe<dat.in>std.out
    fc my.out std.out
    if not errorlevel 1 goto loop
pause
```

Linux Shell

```
while true; do
    ./gen>dat.in
    ./std<dat.in>std.out
    ./my<dat.in>my.out
    if diff std.out my.out; then
        printf OK
    else
        printf DIFF
        exit 0
    fi
done
```

9.3 表

质数表

| 1e2 | 1e3 | 1e4 | 1e5 | 1e6 | |
|----------|-----------|------------|-------------|--------------|--|
| 101 | 1009 | 10007 | 100003 | 1000003 | |
| 1e7 | 1e8 | 1e9 | 1e10 | 1e11 | |
| 10000019 | 100000007 | 1000000007 | 10000000019 | 100000000003 | |

| | | | |
|--------------------|---------------------|----------------------|-------------------|
| 1e12 | 1e13 | 1e14 | 1e15 |
| 10000000000039 | 100000000000037 | 1000000000000031 | 10000000000000037 |
| 1e16 | 1e17 | 1e18 | |
| 100000000000000061 | 1000000000000000003 | 10000000000000000003 | |

典列

| | | | | | | | | | | | |
|-----------|---|---|---|---|----|----|-----|-----|------|-------|--------|
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Catalan | 1 | 1 | 2 | 5 | 14 | 42 | 132 | 429 | 1430 | 4862 | 16796 |
| Bell | 1 | 1 | 2 | 5 | 15 | 52 | 203 | 877 | 4140 | 21147 | 115975 |
| partition | 1 | 1 | 2 | 3 | 5 | 7 | 11 | 15 | 22 | 30 | 42 |
| group | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 5 | 2 | 2 |

| | | | | | | |
|-------------|---|----|----|----|----|---|
| binomial | | | | | | |
| x | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 1 | | | | | |
| 1 | 1 | 1 | | | | |
| 2 | 1 | 2 | 1 | | | |
| 3 | 1 | 3 | 3 | 1 | | |
| 4 | 1 | 4 | 6 | 4 | 1 | |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 |
| Stirling I | | | | | | |
| x | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 1 | | | | | |
| 1 | | 1 | | | | |
| 2 | | 1 | 1 | | | |
| 3 | | 2 | 3 | 1 | | |
| 4 | | 6 | 11 | 6 | 1 | |
| 5 | | 24 | 50 | 35 | 10 | 1 |
| Stirling II | | | | | | |
| x | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 1 | | | | | |
| 1 | | 1 | | | | |
| 2 | | 1 | 1 | | | |
| 3 | | 1 | 3 | 1 | | |
| 4 | | 1 | 7 | 6 | 1 | |
| 5 | | 1 | 15 | 25 | 10 | 1 |
| Lah | | | | | | |
| x | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 1 | | | | | |
| 1 | | 1 | | | | |
| 2 | | 2 | 1 | | | |
| 3 | | 6 | 6 | 1 | | |
| 4 | | 24 | 36 | 12 | 1 | |

```
5 |      120 240 120 20  1
```

```
k-partition
```

```
    0   1   2   3   4   5
```

```
0 | 1
```

```
1 |    1
```

```
2 |    1   1
```

```
3 |    1   1   1
```

```
4 |    1   2   1   1
```

```
5 |    1   2   2   1   1
```