

浅谈集合幂级数和 FWT

数学基础

相关符号

- 多项式系数提取符号
 $[x^i]f(x)$ 表示多项式 $f(x)$ 中 x^i 前的系数。
- 艾弗森括号
 $[P] = \begin{cases} 1, P \text{ 为真} \\ 0, P \text{ 为假} \end{cases}$
- 求积符号
 $\prod_{i=1}^n f(i) = f(1) \times f(2) \times \cdots \times f(n)$
- 异或符号: \oplus
- 集合对称差: \triangle
 $A \triangle B = \{x | x \in A \text{ 且 } x \notin B\} \cup \{x | x \in B \text{ 且 } x \notin A\}$
实际上集合对称差类似于二进制里的异或。

习题

- 求 $\sum_{i=0}^4 [x^i]F(x)$, 其中 $F(x) = 3 - 2x + x^4 - 7x^6$ 。
- 证明 $[S \subseteq T] = \prod_{x \in S} [x \in T]$ 。
- 定义 $F(S) = \sum_{x \in S} 2^x$, 证明 $F(S \triangle T) = F(S) \oplus F(T)$ 。

形式幂级数

形式幂级数是一个形如 $F(x) = \sum_{i=0}^{+\infty} a_i x^i$ 的无限级数（当然实际上在 OI 中，它往往并不是一个无限级数），它将数列 $\{a_n\}$ 映射到了一个幂级数 $F(x)$ 。

值得注意的是 $F(x)$ 往往并不收敛，但是我们不关心它是否收敛，也不关心代入一个 x 时它的值具体是多少。实际上 x^i 只是一个形式记号，并不代表具体的数值。

引入形式幂级数的好处在与它使我们能够便捷地表述数列的运算。

想象一下，你有两个数列 $\{a_n\}$ 和 $\{b_n\}$ ，你希望让这两个数列对应项相加。

定义数列 $\{c_n\}$ ，满足 $\forall i \in \mathbb{N}^+, i \leq n$ 都有 $c_i = a_i + b_i$

在你需要进行大量复杂的数列运算的时候，这种写法简直是毁灭性的。

使用形式幂级数就能简便的描述这件事（设 $A(x), B(x)$ 分别是 $\{a_n\}, \{b_n\}$ 的形式幂级数）

$C(x) = A(x) + B(x)$

这就是形式幂级数的魅力！

和卷积

幂级数 $A(x) = \sum_{i=0}^{+\infty} a_i x^i, B(x) = \sum_{i=0}^{+\infty} b_i x^i$ 的和卷积是

$C(x) = A(x)B(x) = \sum_{i=0}^{+\infty} \sum_{j+k=i} a_j b_k x^i$ （实际上这就是一般意义上的多项式乘法）

异或卷积

$$C(x) = A(x)B(x) = \sum_{i=0}^{+\infty} \sum_{j \oplus k = i} a_j b_k x^i$$

因此在使用幂级数乘法时，必须申明它代表什么卷积。

习题

计算 $(1 - 2x^3)(x^2 + 4x^3)$ 。（乘法是异或卷积）

集合幂级数

类似形式幂级数，定义集合幂级数：

$$F(x) = \sum_{S \subseteq U} f_S x^S$$

其中 f 是一个集合到实数的映射。

这里出现了一个集合做为指数的情况，但我们并不关心 $5^{\{3,5,10\}}$ 或者 $x^{\mathbb{R}}$ 究竟是多少。它们只是形式记号，我们并不关系它的数值。

集合幂级数的作用是提供了一个集合到实数的映射，并且能够便利的对这个映射进行操作。

集合卷积

$$\text{设集合幂级数 } A(x) = \sum_{S \subseteq U} a_S x^S, B(x) = \sum_{S \subseteq U} b_S x^S.$$

- 集合并卷积
$$C(x) = A(x)B(x) = \sum_{S \subseteq U} \sum_{S_1 \cup S_2 = S} a_{S_1} b_{S_2} x^S$$
- 集合交卷积
$$C(x) = A(x)B(x) = \sum_{S \subseteq U} \sum_{S_1 \cap S_2 = S} a_{S_1} b_{S_2} x^S$$
- 集合对称差卷积
$$C(x) = A(x)B(x) = \sum_{S \subseteq U} \sum_{S_1 \triangle S_2 = S} a_{S_1} b_{S_2} x^S$$
- 逐点乘积
$$C(x) = A(x) \cdot B(x) = \sum_{S \subseteq U} a_S b_S x^S$$

习题

- 计算 $(x^{\{e\}} - 2x^{\{e, \sqrt{2}\}})(-x^{\mathbb{R}} + 3x^{\{\sqrt{2}\}})$ 在乘法是并卷积/交卷积时的值。
- 定义 $\text{INV}(F(x)) = \sum_{S \subseteq U} ([x^S]F(x))x^{U/S}$ 。
设 $A(x)$ 是 $F(x)$ 和 $G(x)$ 的交卷积。
 $B(x)$ 是 $\text{INV}(F(x))$ 和 $\text{INV}(G(x))$ 的并卷积。
证明 $A(x) = \text{INV}(B(x))$ 。

FWT

FWT 可以在 $O(n2^n)$ 计算两个集合幂级数的并/交/对称差卷积， n 是集合中元素个数。

基本思想

定义集合幂级数到集合幂级数的映射 FWT 和逆映射 FWT^{-1} ，满足：

- $\text{FWT}^{-1}(\text{FWT}(F(x))) = F(x)$
- $A(x)B(x) = \text{FWT}^{-1}(\text{FWT}(A(x)) \cdot \text{FWT}(B(x)))$

只要能够在 $O(n2^n)$ 时间内计算 FWT 和 FWT^{-1} ，就能做到快速卷积。

定义**特征函数** $\phi(X, Y)$, 使得:

$$[x^Y] \text{FWT}(\sum_{S \subseteq U} f_S x^S) = \sum_{X \subseteq U} f_X \phi(X, Y),$$

特征函数的性质

为了使变换后卷积运算转化为逐点乘积:

$$\text{FWT}(F(x)G(x)) = \text{FWT}(F(x)) \cdot \text{FWT}(G(x))$$

特征函数 $\phi(X, Y)$ 必须满足**乘法同态性**, 即对特定集合运算 \star , 有:

$$\phi(X \star X', Y) = \phi(X, Y) \cdot \phi(X', Y)$$

至于为什么它要满足乘法同态性, 本文不作展开, 感兴趣的可以阅读 [这篇文章](#)

三类卷积的特征函数

1. 对称差卷积

$$\phi_{\Delta}(X, Y) = (-1)^{|X \cap Y|}$$

同态性证明:

$$\begin{aligned} (-1)^{|(X \Delta X') \cap Y|} &= (-1)^{|(X \cap Y) \Delta (X' \cap Y)|} \\ &= (-1)^{|X \cap Y| + |X' \cap Y| - 2|X \cap X' \cap Y|} \\ &= (-1)^{|X \cap Y|} \cdot (-1)^{|X' \cap Y|} \end{aligned}$$

2. 并卷积

$$\phi_{\cup}(X, Y) = [X \subseteq Y]$$

同态性证明:

$$[X \cup X' \subseteq Y] = [X \subseteq Y] \times [X' \subseteq Y].$$

3. 交卷积

补集转化后变为并卷积。

建议把对称差卷积和并卷积的公式背下来, 注意背的是公式而不是背代码, 因为公式在难题的推导过程中很有用。

具体实现

对称差卷积

对称差卷积的代码建议全文背诵。

```
void FWT_xor(int *a)
{
    for(int mid = 1; mid < (1 << n); mid <= 1)
    {
        for(int i = 0; i < (1 << n); i += (mid << 1))
        {
            for(int j = 0; j < mid; j++)
            {
```

```

        int x = a[i + j], y = a[i + j + mid];
        a[i + j] = (x + y) % mod;
        a[i + j + mid] = (x - y + mod) % mod;
    }
}
}
}

```

FWT⁻¹的方法是对一个幂级数做 FWT 之后让所有系数除以 2^n (n 是集合中元素个数)

```

#define MOD 998244353
void IFFT_xor(int n, int *a){
    FWT_xor(n, a);
    int n1=power((1<<n), MOD-2);
    for(int i=0;i<(1<<n);i++){
        a[i]*=n1;a[i]%=MOD;
    }
}

```

你会发现 FWT 的代码和 FFT 的代码几乎一模一样，FWT⁻¹的方式也和 FFT⁻¹的方式一样，这是因为它们的本质都是让数列乘以范德蒙矩阵。

并卷积

根据并卷积的特征函数，求 FWT($F(x)$) 的过程就是高维前缀和，FWT⁻¹($F(x)$) 的过程就是高维差分。

```

void FWT_or(int *a)
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < (1 << n); j++)
        {
            if(j >> i & 1)
                a[j] = (a[j] + a[j - (1 << i)]) % mod;
        }
    }
}

void IFFT_or(int *a)
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < (1 << n); j++)
        {
            if(j >> i & 1)
                a[j] = (a[j] - a[j - (1 << i)] + mod) % mod;
        }
    }
}

```

交卷积

补集转化后套用并卷积的代码。

例题

让我们先看三道模板题。

模板1（自编题）

对于每一个 $T \subseteq \{1, 2, \dots, m\}$, 分别回答如下问题：

问有多少种集合列 S_1, \dots, S_n , 满足：

- $S_i \subseteq \{1, 2, \dots, m\}$
- $\text{GCD}_{x \in S_i} x = 1$
- $\bigcup_{i=1}^n S_i = T$

$$m \leq 20, n \leq 10^9$$

解析

这道题体现了集合幂级数最主要的作用：计数。

考虑幂级数 $F(x) = \sum_{S \subseteq \{1, 2, \dots, m\}} [\text{GCD}_{a \in S} = 1] x^S$ 。

那么我们有如下结论：

■ $[x^T]F(x)$ 等于 $n = 1$ 时 T 的答案。

考虑 $n = 2$ 时如何计算 T 的答案。我们有如下结论：

■ $[x^T](F(x)^2)$ 等于 $n = 2$ 时 T 的答案。（这里的平方是集合并卷积）

证明也比较显然： $[x^T](F(x)^2) = \sum_{S_1 \cup S_2 = T} [S_1]F(x) \times [S_2]F(x)$, 这里 S_1 就是第一个集合的选择, S_2 就是第二个集合的选择, 因此这个求和符号一定取遍所有合法的选法。

类似地, 我们有：

■ $[x^T](F(x)^k)$ 等于 $n = k$ 时 T 的答案。

证明显然。

所以本题的答案是 $[x^T](F(x)^n)$ 。只需要计算 $F(x)^n$, 提取各项系数就是答案。

可以用类似快速幂的方法计算 $F(x)^n$, 这样复杂度是 $O(m2^m \log n)$, 更好的办法是先算出 $\text{FWT}(F(x))$, 所有系数都取 n 次方后, 再跑一遍 FWT^{-1} 即可得出答案。最终复杂度 $O(m2^m + 2^m \log n)$ 。

评析

这道题唯一考察的知识点是幂级数的运用。幂级数最常见的运用便是用于计数题, 使用系数代表方案数, 这样幂级数的求和/卷积都具有了组合意义。

补充练习

为了让大家能够充分理解幂级数的组合意义，我们再补充几道题：

- 证明选出八个质数，使它们的和是 114514 的方案数是 $[x^{114514}](\sum_{p \in \mathbb{P}} x^p)^8$ (和卷积)
- 假设肯德基有三种套餐，分别是汉堡加薯条套餐，汉堡加可乐套餐，薯条加鸡块套餐。麦当劳除了上述三种套餐之外，还有土豆泥加薯条加可乐套餐。使用集合幂级数的知识，求出在肯德基吃两份套餐，在麦当劳吃三份套餐后，你把所有食物都尝了至少一遍的方案数。（只需要列式）
- 定义一种选数方案的权值是选出的所有数的阶乘的乘积，求出八个质数，使它们的和是 114514 的所有方案的权值和。（只需要列式）

模板2 ([ABC396G] Flip Row or Col)

有一个 $n \times m$ 的 01 矩阵，可以进行任意多次操作，每次将某一行或某一列 01 翻转，问操作后矩阵中所有数的和最小是多少。

$$n \leq 18$$

$$m \leq 2 \times 10^5$$

解析

一个思路是枚举哪些行进行翻转，假设进行翻转的行的集合是 S ，那么答案显然是

$$\sum_{i=1}^m \min(|S \triangle B_i|, n - |S \triangle B_i|).$$

$$\text{设 } cnt_A = \sum_{i=1}^m [B_i = A]$$

让我们推一波式子：

$$\begin{aligned} & \sum_{i=1}^m \min(|S \triangle B_i|, n - |S \triangle B_i|) \\ &= \sum_{T \subseteq U} \min(|T|, n - |T|) \sum_{i=1}^m [S \triangle B_i = T] \\ &= \sum_{T \subseteq U} \min(|T|, n - |T|) \sum_{i=1}^m [T \triangle B_i = S] \\ &= \sum_{T \subseteq U} \min(|T|, n - |T|) \sum_{A \subseteq U} cnt_A [T \triangle A = S] \\ &= [x^S] ((\sum_{T \subseteq U} \min(|T|, n - |T|) x^T) \times (\sum_{A \subseteq U} cnt_A x^A)) \quad (\text{这里的乘法是异或卷积}) \end{aligned}$$

然后就做完了，复杂度 $O(n2^n)$ 。

简单讲一下推式子的思路：

这道题显然使用异或卷积做，但异或卷积维护的是形如 $[A \triangle B = S] f(A) g(B)$ 的形式而非 $\min(|S \triangle B_i|, n - |S \triangle B_i|)$ 的形式。

所以我们要做的是把后者向前者的形式靠拢，自然的想法是枚举 $T = S \triangle B_i$ ，这样 S 就能从 T 和 B_i 转移而来。

模板3 ([ABC212H] Nim Counting)

给定 n 和正整数集合 $S \subseteq [1, V]$,

求满足如下条件的正整数数列 $\{a_1, a_2, \dots, a_m\}$ 的个数。

- $m \leq n$
- $a_i \in S$
- 在 $\{a_1, a_2, \dots, a_m\}$ 上玩 Nim 游戏，先手必胜。

解法

为了便于统计，改为求先手必败的方案数，那么

第三个条件等价于 $\bigoplus_{i=1}^m a_i = 0$ ，所以答案是 $\sum_{m=1}^n [x^0](\sum_{a \in S} x^a)^m$ （异或卷积）

直接计算这个式子太慢了，因此考虑推一下：

设 $f(x) = \text{FWT}(\sum_{a \in S} x^a)$

$$\begin{aligned} & \sum_{m=1}^n [x^0](\sum_{a \in S} x^a)^m \\ &= \sum_{m=1}^n [x^0] \text{FWT}^{-1}(f(x)^m) \quad (\text{乘法是逐项相乘}) \\ &= \sum_{m=1}^n \frac{1}{V} [x^0] \text{FWT}(f(x)^m) \\ &= \sum_{m=1}^n \frac{1}{V} \sum_{i=0}^V [x^i](f(x)^m) \\ &= \sum_{i=0}^V \frac{1}{V} \sum_{m=1}^n [x^i](f(x)^m) \\ &= \sum_{i=0}^V \frac{1}{V} \sum_{m=1}^n ([x^i]f(x))^m \end{aligned}$$

预计算出 $f(x)$ 的各项系数后，用等比数列求和公式快速计算第二个求和符号，然后枚举第一个求和符号，这道题就做完了。时间复杂度 $O(n \log n)$ 。

评析

和第一道自编题类似，这道题同样用幂级数来计数，但是区别在与如果这道题直接进行幂级数运算，复杂度太慢。

为了优化复杂度，我们对式子进行了推导，使它变成易于维护的形式。而推导式子的技巧在与将卷积变成先 FWT，再点乘，再 FWT^{-1} 的运算，然后利用 $\text{FWT}()$ 的定义将其展开。

接下来，让我们看两道有难度的题。

困难题1（[省选联考 2022] 卡牌）

给定正整数集合 S ， q 次问讯，每次问讯给出一个素数组成的集合 P ，查询有多少满足条件的集合 T ，要求 $T \subseteq S$ 且 $\forall p \in P, \prod_{x \in T} x$ 被 p 整除。

$|S| \leq 10^6, q \leq 1500, \sum |P| \leq 18000, S$ 中的所有数都不超过 2000。

解法

有一个关于素数的经典 trick：

不超过 \sqrt{n} 的素数个数很少，而任意两个大于 \sqrt{n} 的素数的乘积大于 n 。

在这道题中，我们就可以使用这样的思路。小于等于 43 的素数只有 14 个，对于这 14 个素数，可以使用 FWT 计算。

设 $P(x)$ 表示 x 的所有小于等于 43 的素因数构成的集合。

如果 P 中的所有素数都小于等于 43，那么答案是 $\sum_{P \subseteq T} [x^T] \prod_{a \in S} (1 + x^{P(a)})$

（想一下这里应该是什么类型的卷积）

但是如果 P 中有大于 43 的素数，例如 53，那么所有被 53 整除的数都不选的方案就应该被排除掉。

设 S_p 表示 S 中被 p 整除的数的集合， S_{extra} 表示 S 中不被任何一个 P 中大于 43 的素数整除的数构成的集合。

显然地，对于所有大于 43 的 p ， S_p 之间两两无交， S_p 和 S_{extra} 也无交。这样就把 S 划分为了多个部分，可以对每个部分分别计算，再乘起来了。

定义 $F(S) = \prod_{a \in S} (1 + x^{P(a)})$

答案是 $\sum_{P' \subseteq T} [x^T] (F(S_{\text{extra}})) \times (\prod_{p \in P, p > 43} (F(S_p) - 1))$

这里 P' 是 P 中小于等于 43 的素数的集合。

$$F(S_{\text{extra}}) = \frac{F(S)}{\prod_{p \in P, p > 43} F(S_p)}$$

这样只要预处理 $F(S_p)$, 对于每个问讯分别计算即可。

在开启下一道困难题前, 让我们先看一道铺垫题:

铺垫题

对于给定的数列 $\{A_i\}$, 求出 $j = 0, 2, \dots, V$ 时 $\sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}]$ 。

解法

$$\begin{aligned} & \sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}] \\ &= \sum_{i=1}^n [(-1)^{\text{popcount}(A_i \& j)} = 1] \\ &= \frac{1}{2} (n + \sum_{i=1}^n (-1)^{\text{popcount}(A_i \& j)}) \\ &= \frac{1}{2} (n + \sum_{i=1}^n [x^j] \text{FWT}(x^{A_i})) \\ &= \frac{1}{2} (n + [x^j] \sum_{i=1}^n \text{FWT}(x^{A_i})) \\ &= \frac{1}{2} (n + [x^j] \text{FWT}(\sum_{i=1}^n x^{A_i})) \end{aligned}$$

解释一下第二个等号,

$$\sum_{i=1}^n (-1)^{\text{popcount}(A_i \& j)} = \sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}] - \sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是奇数}]$$

又因为 $\sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}] + \sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是奇数}] = n$, 联立即可证明第二个等号。

第三个等号是 FWT 的定义。

第五个等号是由 FWT 的线性性得出。

因此只需要对 $\sum_{i=1}^n x^{A_i}$ 进行一次 FWT 即可。

[ABC367G] Sum of (XOR^K or 0)

求 $\sum_{S \subseteq \{1, 2, \dots, n\}, m | (|S|)} (\bigoplus_{i \in S} A_i)^k$ 。

$$n, k \leq 2 \times 10^5$$

$$m \leq 100$$

$$A_i \leq 2^{20}$$

解法

概要

本题的核心思想: 异或和的 k 次方没有良好的性质, 因此考虑枚举异或和是 a , 设有 cnt_a 种选数方案 (选出 m 的倍数个数) 能够让选出的数的异或和为 a , 答案就是 $\sum_{a=0}^V a^k \text{cnt}_a$ 。

因此我们必须快速求出 cnt 数组。为此我们设计出生成函数 $\prod_{i=1}^n (x^{A_i} y + x^0)$ (其中 x 的乘法是异或卷积, 即 $x^a \times x^b = x^{a \oplus b}$, y 的乘法是模 m 的循环卷积, 即 $y^a \times y^b = y^{(a+b) \bmod m}$)。

容易注意到该生成函数 $x^a y^b$ 前的系数等于从 $\{A_i\}$ 种选出若干个数, 使得它们的异或和为 a , 且选出的数的个数 $\bmod m = b$ 的方案数。所以有 $\text{cnt}_a =$ 生成函数中 $x^a y^0$ 前的系数。

为了快速计算出该生成函数，我把计算过程拆分为三部分。其中两部分放入了如下的两个引理中。

两个引理

引理 1：存在解法能够在 $O(n + V \log V)$ 时间内对于给定的数列 $\{A_i\}$ ，求出 $j = 0, 2, \dots, V$ 时 $\sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}]$ 。

就是之前那道铺垫题。

引理 2：存在解法能在 $O(V \log V + nm)$ 的时间内，对于给定的数列 $\{A_i\}$ 和所有的 j ，求出 $[y^0] \prod_{i=1}^n (1 + (-1)^{\text{popcount}(A_i \& j)} y)$ （注意这里的多项式乘法是模 m 意义下的循环卷积，即 $y^a \times y^b = y^{(a+b) \bmod m}$ ）

证明： $\prod_{i=1}^n (1 + (-1)^{\text{popcount}(A_i \& j)} y)$
 $= (1 + y)^{\sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}]} (1 - y)^{\sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是奇数}]}$

考虑对每个 i 预处理出 $(1 + y)^i$ 和 $(1 - y)^i$ 的表达式，然后使用引理 1 求出 $\sum_{i=1}^n [\text{popcount}(A_i \& j) \text{ 是偶数}]$ 即可。

因为次数是模 m 意义下的，所以多项式的项数不超过 m ，因为 m 很小，所以完全可以暴力计算多项式乘法，复杂度是 $O(nm^2)$ ，但是注意到只有 y^0 的系数对我们有用，所以可以只计算 y^0 系数，复杂度降低至 $O(nm)$ 。加上引理 1 里的 $O(V \log V)$ ，总复杂度 $O(nm + V \log V)$ 。

引理证毕。

主要推导部分

题目求的是

$$\begin{aligned} & \sum_{S \subseteq \{1, 2, \dots, n\}, m | (|S|)} \left(\bigoplus_{i \in S} A_i \right)^k \\ &= \sum_{a=0}^V a^k \sum_{S \subseteq \{1, 2, \dots, n\}, m | (|S|)} [\bigoplus_{i \in S} A_i = a] \\ &= \sum_{a=0}^V a^k [x^a y^0] \prod_{i=1}^n (x^{A_i} y + x^0) \quad (x \text{ 的乘法是异或卷积, } y \text{ 的乘法是模 } m \text{ 意义下的循环卷积}) \end{aligned}$$

现在我们来研究 $\prod_{i=1}^n (x^{A_i} y + x^0)$

$$\begin{aligned} & \prod_{i=1}^n (x^{A_i} y + x^0) \\ &= \text{FWT}^{-1}(F_1(x) \cdot F_2(x) \cdot \dots \cdot F_n(x)) \quad (``\cdot'' \text{ 表示对应项的系数相乘}) \end{aligned}$$

其中 $F_i = \text{FWT}(x^{A_i} y + x^0)$

$$\begin{aligned} &= \text{FWT}(x^{A_i} y) + \text{FWT}(x^0) \\ &= \sum_{j=0}^V ((-1)^{\text{popcount}(A_i \& j)} y + 1) x^j \end{aligned}$$

$$\begin{aligned} & \text{所以 } \prod_{i=1}^n (x^{A_i} y + x^0) \\ &= \text{FWT}^{-1}(\sum_{j=0}^V (\prod_{i=1}^n (1 + (-1)^{\text{popcount}(A_i \& j)} y)) x^j) \end{aligned}$$

使用引理 2 的方法算出 $[y^0] (\prod_{i=1}^n (1 + (-1)^{\text{popcount}(A_i \& j)} y))$ ，然后进行 FWT^{-1} 即可得到 $\prod_{i=1}^n (x^{A_i} y + x^0)$ 的各项系数，于是这道题就做完了。

总结

如果 $m = 1$ ，这题并不困难，但是 m 的约束很难单独处理，因此巧妙地设计出了二元生成函数，从而做到同时满足 m 的约束和异或和的约束。

而在处理二元生成函数卷积时，采用了主元的思想，将 y 视作常数进行 FWT。

快速计算 $\sum_{i=0}^n [\text{popcount}(A_i \& j) \text{ 是偶数}]$ 也是一个难点。因为这个式子的形式和异或卷积的形式非常相似，所以尝试构造一个生成函数，对它进行沃尔什正变换，再恒等变形后得到 $\sum_{i=0}^n [\text{popcount}(A_i \& j) \text{ 是偶数}]$ 。