IOI2019国家集训队第一阶段作业

《画家小P》解题报告

南京外国语学校 吴思扬

2018 年 10 月

1 题目大意

你有一张 n 个点 m 条无向边的简单无向图（无重边和自环），点标号为 1, 2, ..., n。你要给图中每一个点染上一种颜色，设图中第 i 个点被染上的颜色为 col_i，则图的美丽值为 $col_1 \oplus col_2 \oplus col_3 \oplus ... \oplus col_n$（$\oplus$ 表示异或和）。每一个点染上的颜色 col_i，必须是 0, 1, 2, ..., $limit_i$ 中的一个整数（$limit_i$ 为输入给定的数）。

一个图是**不单调**的，当且仅当对于**所有**边 (u, v)，满足 $col_u \neq col_v$。

问有多少种不同的染色方案使得染色后的图不单调，并且美丽值为 C（C 为输入给定的数）。两种染色方案是不同的当且仅当存在至少一个点 u，使得 u 在两张图中的颜色 col_u 不同。

由于答案可能较大，请输出答案对 998244353 取模的结果。

2 限制与约定

本题采用捆绑测试。

对于所有测试包均满足 $1 \leq n \leq 15$, $0 \leq m \leq \frac{n(n-1)}{2}$, $0 \leq C, limit_i \leq 10^{18}$, $1 \leq u, v \leq n$。

对于所有测试包，时间限制均为 **1s**，空间限制均为 **512MB**。

测试包编号	n 的范围	m 的范围	C, $limit_i$ 的范围	测试包分值
1	≤ 3	$\leq \frac{n(n-1)}{2}$	[0, 500]	3
2	≤ 10	0	[0, 1000]	7
3	≤ 10	0	$[0, 10^{18}]$	15
4	≤ 6	$\leq \frac{n(n-1)}{2}$	[0, 100]	10
5	≤ 6	$\leq \frac{n(n-1)}{2}$	$[0, 10^{18}]$	10
6	≤ 8	$\leq \frac{n(n-1)}{2}$	$[0, 10^{18}]$	5
7	≤ 12	$\leq \frac{n(n-1)}{2}$	$[0, 10^{18}]$	20
8	≤ 13	$\leq \frac{n(n-1)}{2}$	$[0, 10^{18}]$	5
9	≤ 14	$\leq \frac{n(n-1)}{2}$	$[0, 10^{18}]$	5
10	≤ 15	$\leq \frac{n(n-1)}{2}$	$[0, 10^{18}]$	20

3 算法讨论

3.1 算法1

按照题意对每个点枚举颜色 col_u，枚举完后判断是否满足异或和为 C，以及是否对于每条边 (u, v) 都满足 $col_u \neq col_v$，如果这两个条件都成立，则将此方案统计入答案。

时间复杂度为 $O((n + m) \cdot \Pi_{i=1}^{n} limit_i)$。

空间复杂度为 $O(n + m)$。

该算法可以通过第一个测试包，期望得分 3 分。

3.2 算法2：$m = 0$ 时的做法

观察题目，发现图不单调这一限制要求我们在求解时从整体来考虑整个图的染色情况，对我们的求解造成了不少困难。我们不妨先从特殊情况考虑，若没有相邻两个点颜色不同这一限制，即 $m = 0$ 时，应该如何求解。

3.2.1 算法2.1

按照从 1 到 n 的顺序依次枚举每个点的颜色，由于没有颜色必须不同的限制，因此只有当前已经确定颜色点的异或和会影响未确定点的颜色选择。

考虑通过动态规划来解决。令 $dp(i, x)$ 表示已经确定了前 i 个点的颜色，前 i 个点的颜色异或和为 x 的方案数。

则有转移方程：

$$dp(i, x) = \sum_{y=0}^{limit_i} dp(i - 1, x \oplus y),\ 1 \leq i \leq n$$

边界条件为：$dp(0, x) = \begin{cases} 1, & x = 0 \\ 0, & x > 0 \end{cases}$。

可以发现，$dp(i, x)$ 中有用的值（非 0 值）需满足 $\lfloor log_2 x \rfloor \le \left\lfloor log_2(\max_{j \le i}\{limit_j\}) \right\rfloor$，故 dp 总状态数为 $O(n \cdot C)$[1]，转移复杂度为 $O(C)$。

时间复杂度为 $O(n \cdot C^2)$。

空间复杂度为 $O(n \cdot C)$，可以使用滚动数组优化到 $O(C)$。

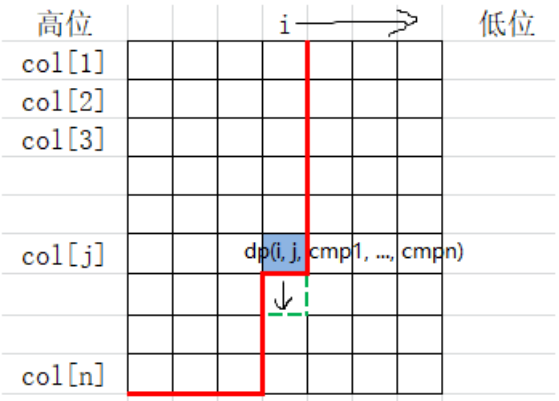
该算法可以通过第二个测试包，期望得分 7 分。

3.2.2　算法2.2

在 C 较大时，**算法2.1**中的动态规划解法复杂度无法接受，不能对每个点确定出它的颜色。

由于异或运算中每一个二进制位相互独立，故考虑采用数位动态规划的方法来解决。

我们将所有数表示成二进制，按从高位到低位的顺序，对于每一位按照从 1 到 n 的顺序依次确定 col_u 在这一位上的取值。令 $dp(i, j, cmp_1, cmp_2, ..., cmp_n)$ 表示已经确定完比 i 高的位上所有 col 的值和第 i 位上 $col_1, col_2, ..., col_j$ 的值，col_k 与 $limit_k$ 的大小关系情况为 cmp_k（$1 \le k \le n$），如下图所示。



转移时枚举 $\begin{cases} col_{j+1} \otimes 2^i, & j < n \\ col_1 \otimes 2^{i-1}, & j = n \end{cases}$ 的值（$\otimes$ 表示按位与运算），并更新 cmp 即可。总状态数为 $O(log_2 C \cdot n \cdot 2^n)$，转移复杂度为 $O(1)$。

[1]由于本题中 C 与 $limit_i$ 同阶，故在本文计算复杂度时若无特别标注均以 C 代替 $\max(C, \max_{1 \le i \le n}\{limit_i\})$

时间复杂度为 $O(log_2C \cdot n \cdot 2^n)$。

空间复杂度为 $O(log_2C \cdot n \cdot 2^n)$，可以用滚动数组优化到 $O(2^n)$。

该算法可以通过第二、三个测试包，期望得分 22 分。

3.2.3 算法2.3

仍然沿用**算法2.2**的思想，按位考虑计算答案。我们先确定 *limit* 最大的点在最高位上 *col* 的值。设 *limit* 最大的点编号为 u，u 的最高位为第 i 位，col_u 在第 i 位上取值为 x，除 u 以外其它节点的 *col* 最终取值异或和为 D。

定理 3.1. 当 $\lfloor log_2(limit_u) \rfloor \geq \lfloor log_2 C \rfloor$ 时，若 C 与 D 在第 i 位上值相同，即 $C \otimes 2^i = D \otimes 2^i$，则存在且只存在一个 col_u 满足 $0 \leq col_u < 2^i$, $D \oplus col_u = C$（$\otimes$ 表示按位与运算，$\oplus$ 表示异或运算）。

证明. $\because limit_v \leq limit_u$, $v \neq u$，

$\therefore col_v < 2^{i+1}$, $v \neq u$，

$\therefore D < 2^{i+1}$。

又 C 的最高位不超过 i，

$\therefore C \oplus D < 2^{i+1}$。

又 $C \otimes 2^i = D \otimes 2^i$，

$\therefore C \oplus D < 2^i$，即 $col_u < 2^i$。 □

观察出这些性质后，我们便可以通过递归的方式高效处理问题。具体地，我们令 $F(\{limit_1, limit_2, ..., limit_n\}, C)$ 表示一组 *limit* 和 C 的答案。

1. 如果 $limit_u = 0$，那么答案为 $\begin{cases} 1, & C = 0 \\ 0, & C > 0 \end{cases}$。
2. 如果 $\lfloor log_2(limit_u) \rfloor < \lfloor log_2 C \rfloor$，则 C 的最高位 *col* 异或和无论如何都为 0，故答案为 0。
3. 枚举 x 的值。
   (a) 如果 $x = 0$，可以通过动态规划算法求出 D 与 C 第 i 位相同的解的数量，具体实现与**算法2.1**类似，不再赘述。

(b) 如果 $x = 1$，则答案为 $F(\{limit_1, limit_2, ..., limit_{u-1}, limit_u \oplus 2^i, limit_{u+1}, ..., limit_n\}, C \oplus 2^i)$。

由于每递归一次，都会减少 *limit* 最高位上一个数，故最多递归 $O(n\ log_2 C)$ 次。每次递归需要 $O(n)$ 时间处理**3.(a)**，故时间复杂度为 $O(n^2\ log_2 C)$。

空间复杂度为 $O(n)$。

该算法可以通过第二、三个测试包，期望得分 22 分。

3.3 算法3

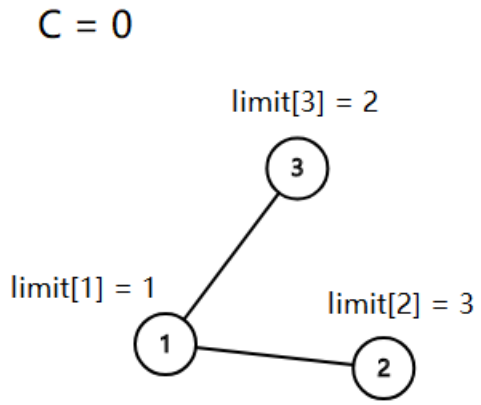
现在来解决 $m > 0$ 的情况，$m > 0$ 的主要难点在于相邻点之间 *col* 存在不相等的限制，这让我们在算一个点的 *col* 值时还要考虑与它相邻点的 *col* 值。如果我们反其道而行之，强制一些限制不满足，即对于一些边 (u, v) 强制要求 $col_u = col_v$，并且不考虑其它边的限制，就可以将问题转化成 $m = 0$ 了，因此考虑容斥。

3.3.1 算法3.1

我们记图的边集为 E，我们每次枚举 E 的一个子集 S，要求 $\forall (u, v) \in S, col_u = col_v$，记所有满足这个条件并且 $col_1 \oplus col_2 \oplus ... col_n = C, 0 \le col_i \le limit_i$ 的染色方案数为 $ans(S)$。则答案为 $\sum_S (-1)^{|S|} ans(S)$。

我们会在 $S = \emptyset$ 时将所有符合题意的答案都算入，但会多算不满足 $col_u \neq col_v$ 限制的方案。在 $|S| = 1$ 时将至少有一条边 $col_u = col_v$ 的不合法方案给减去，但会多扣有至少两条边不合法的方案。在 $|S| = 2$ 时将多扣的至少两条边不合法的方案再加回来，以此类推，最终得到的就是所有合法方案的数量。下为一组例子：

例1



当 $S = \emptyset$ 时，会将 $\{0,0,0\}$, $\{0,1,1\}$, $\{0,2,2\}$, $\{1,0,1\}$, $\{1,1,0\}$, $\{1,3,2\}$ 计算一遍。

当 $S = \{(1,3)\}$ 时，会将 $\{0,0,0\}$, $\{1,0,1\}$ 减去一次。

当 $S = \{(1,2)\}$ 时，会将 $\{0,0,0\}$, $\{1,1,0\}$ 减去一次。

此时 $\{0,0,0\}$ 被减去了两次。

当 $S = \{(1,2), (1,3)\}$ 时，会将 $\{0,0,0\}$ 计算一遍。

最终被恰好计算一次的有 $\{0,1,1\}$, $\{0,2,2\}$, $\{1,3,2\}$。

现在考虑如何计算 $ans(S)$。对 $(u, v) \in S$，将 u 与 v 连边，最后每一个联通块中的点 col 值应相同。将联通块按大小奇偶性分成两组，设第 i 个联通块中 $limit$ 最小值为 a_i，大小为奇数的联通块编号为 $p_1, p_2, ..., p_k$，大小为偶数的联通块编号为 $q_1, q_2, ..., q_l$。

- 对于大小为奇数联通块，计算 $ans(S)_{odd} = F(\{a_{p_1}, a_{p_2}, ..., a_{p_k}\}, C)$。
- 对于大小为偶数联通块，计算 $ans(S)_{even} = \Pi_{i=1}^{l}(a_{q_i} + 1)$。

则 $ans(S) = ans(S)_{odd} \cdot ans(S)_{even}$。

需要枚举每一个 E 的子集，容斥的时间复杂度为 $O(2^m)$。如果每一次容斥后都计算一遍 F 复杂度太高，尝试优化。可以发现 F 值只跟 a 有关，设图的点集为V，可以对 V 的每个子集 T，预处理出 $F(\{limit_u \mid u \in T\}, C)$，这样的子集共有 2^n 个，只用计算 2^n 次 F 函数。

使用**算法2.1**的求 F 方法，可以通过第一二四个测试包，期望得分 20 分。

使用**算法2.2**或**算法2.3**的求 F 方法，可以通过前五个测试包，期望得分 45 分。

3.3.2 算法3.2

算法3.1需要枚举 E 的每一个子集 S 来计算答案，当 m 较大时复杂度不能接受，尝试优化。

观察容斥之后 $ans(S)$ 的求法，可以发现 $ans(S)$ 的计算只和联通块有关，与具体连边方式无关。也就是说，有一些边集得到的联通块划分是相同的，如果可以知道这些边集容斥系数之和，那么就可以大大优化复杂度。

我们尝试枚举最后联通块划分的情况，设划分为 $P, P = \{S_1, S_2, ..., S_k\}$, S_i 为一个联通块。P 的系数应为 $\sum_{E' \subset E}$ [连接 E' 中所有边得到的联通块情况为 P] $(-1)^{|E'|}$。我们对 V 的子集 S 定义 $coef(S) = \sum_{E' \subset E}$[$E'$ 中所有边两端点都在 S 内，且连接 E' 中所有边使得 S 联通] $(-1)^{|E'|}$，则有 P 的系数为 $\Pi_{i=1}^{k} coef(S_i)$。故我们只用对每个 S 算出 $coef(S)$，然后再枚举点集的划分情况 P，就可以计算答案了。

考虑如何计算 $coef(S)$。求 $coef(S)$ 所枚举的 E' 必须满足每条边 e 两端在 S 中，且能使 S 联通，两个条件同时考虑较为麻烦。先算每条边两端都在 S 中的边集对 $coef(S)$ 的贡献，然后减去不连通的方案。

具体地，令 u 为 S 中编号最小的点，设 $cntE(S)$ 为两端都在 S 中的边的数量。令 $fE(S) = \sum_{i=0}^{cntE(S)} \binom{cntE(S)}{i} (-1)^i$，显然 $fE(S) = \begin{cases} 1, & cntE(S) = 0 \\ 0, & cntE(S) > 0 \end{cases}$。$coef(S) = fE(S) - \sum_{T \subset (S \setminus \{u\})} coef(T \cup \{u\}) \cdot fE(S \setminus (\{u\} \cup T))$。

有了 $coef(S)$ 后便可以枚举划分 P，之后与**算法3.1**一样。

由于 P 的个数为 $Bell(n)$，故容斥部分复杂度变为 $O(3^n + Bell(n))$。如果使用**算法2.3**算 F 值方法，时间复杂度为 $O(3^n + 2^n \cdot n^2 log_2 C + Bell(n))$。

空间复杂度为 $O(2^n + n)$。

根据实现的好坏，可以通过前七至九个测试点，期望得分 70 – 80 分。

3.4 算法4

与**算法3**一样，仍然尝试通过容斥的方法来计算答案。此处将分享另一种容斥方法。

通过枚举 V 的一种划分 $P = \{S_1, S_2, ..., S_k\}$，强制要求划分到同一集合的点 col 相同，对于不同集合中的点不做要求，计算出此条件下满足异或和以及

limit 限制的染色方案数，再乘上 P 的容斥系数 $coef(P)$ 即可得到答案。现在问题变成如何计算容斥系数 $coef(P)$。

3.4.1 算法4.1

我们考虑一组染色方案 $col_1, col_2, col_3, ..., col_n$，对这种方案，将 n 个点划分成 l 个集合 $Q = \{T_1, T_2, ..., T_l\}$，两个点被划分到同一集合中当且仅当 col 相同。

对于两种划分 $P = \{S_1, S_2, ..., S_k\}$ 和 $Q = \{T_1, T_2, ..., T_l\}$，称 P 为 Q 的一种划分当且仅当存在一种对 Q 每个集合 T 继续划分的方案，使得 Q 能变成 P。譬如 $Q = \{\{1,2\}, \{3\}\}$, $P = \{\{1\}, \{2\}, \{3\}\}$，则 P 为 Q 的一种划分。

定理 3.2. *对于容斥时枚举的一种划分 $P = \{S_1, S_2, ..., S_k\}$，染色方案 $Q = \{T_1, T_2, ..., T_l\}$ 能在算 P 的答案时被算到，当且仅当 P 为 Q 的一种划分。*

证明. $\because \forall S_i$, 存在 T_j 使得 $S_i \subset T_j$，$\forall u, v \in T_j, col_u = col_v$，

$\therefore \forall u, v \in S_i, col_u = col_v$，满足划分 P 的要求。 □

我们希望对于一个 Q，若它被算入最终答案中，则其所有划分 P 的 $coef(P)$ 之和应为 1，否则为 0。那么我们可以列方程来求解 $coef(P)$。令 $status(Q)$ 表示 Q 是否会被算入最终答案中，1 表示算入，0 表示不算入。令 $P_1, P_2, ..., P_t$ 为点集 $V = \{1, 2, ..., n\}$ 的所有划分，$t = Bell(n)$。则有方程

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,t} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ A_{t,1} & A_{t,2} & \cdots & A_{t,t} \end{pmatrix} \cdot \begin{pmatrix} coef(P_1) \\ coef(P_2) \\ \vdots \\ coef(P_t) \end{pmatrix} = \begin{pmatrix} status(P_1) \\ status(P_2) \\ \vdots \\ status(P_t) \end{pmatrix}$$

其中当 P_j 为 P_i 的一种划分时，$A_{i,j}$ 为 1，否则为 0。

解方程使用高斯消元算法，时间复杂度为 $O(Bell(n)^3)$。使用**算法2.3**中计算 F 的方法，总时间复杂度为 $O(Bell(n)^3 + 2^n \cdot n^2 log_2 C)$。

空间复杂度为 $O(Bell(n)^2 + n + 2^n)$。

可以通过第一四五个测试包，期望得分 23 分。

3.4.2 算法4.2

算法4.1时间复杂度瓶颈在于高斯消元，尝试优化。

易发现若 P_j 为 P_i 的一种划分，P_j 的集合数一定不会少于 P_i 的集合数，所以如果将 P 按照集合数从小到大排列，则只有当 $j \geq i$ 时 $A_{i,j}$ 有可能为 1，即 A 为上三角矩阵。

当 A 是上三角矩阵时，就无须高斯消元解方程，按照从下往上的顺序就可以计算 $coef$，时间复杂度为 $O(Bell(n)^2)$。

总时间复杂度为 $O(Bell(n)^2 + 2^n \cdot n^2 log_2 C)$。

空间复杂度为 $O(Bell(n)^2 + n + 2^n)$。

可以通过第一四五六个测试包，期望得分 28 分。

3.4.3 算法4.3

算法4.2主要时间复杂度仍然在于求 $coef$ 上，当 n 较大时 $Bell(n)^2$ 无法接受，尝试对此进行优化。

定理 3.3. *存在一组 $g(S)$, $S \subset \{1, 2, ..., n\}$，使得对于划分 $P = \{S_1, S_2, ..., S_k\}$，$coef(P) = \Pi_{i=1}^{k} g(S_i)$。*

证明. 对于点集 $S \subset \{1, 2, ..., n\}$，我们定义 $h(S) =$ [原图中没有边两端都在 S 内]，则有对于划分 $P = \{S_1, S_2, ..., S_k\}$，$status(P) = \Pi_{i=1}^{k} h(S_i)$。把 h 看成集合幂级数，令 $h' = ln(h)$，即 $h(S) = \sum_{T_1, T_2, ..., T_k} [\cup_{i=1}^{k} T_i = S][\forall\, i \neq j,\, T_i \cap T_j = \emptyset] \Pi_{i=1}^{k} h'(T_i)$。

考虑划分 Q 的所有划分 $P_1, P_2, ..., P_k$，有 $status(Q) = \sum_{i=1}^{k} coef(P_i) = \sum_{i=1}^{k} \Pi_j g(S_{i_j})$，其中 $P_i = \{S_{i_1}, S_{i_2}, ...\}$。

令 $Q = \{T_1, T_2, ..., T_l\}$，则又有 $status(Q) = \Pi_{i=1}^{l} h(T_i) = \Pi_{i=1}^{l} (\sum_{R_1, R_2, ..., R_k} [\cup_{j=1}^{k} R_j = T_i][\forall\, a \neq b,\, R_a \cap R_b = \emptyset] \Pi_{j=1}^{k} h'(R_j)) = \sum_{P = \{S_1, S_2, ..., S_k\}} [P$ 为 Q 的一种划分 $] \Pi_{j=1}^{k} h'(S_j)$。

故 $h'(S)$ 即为一组可行 $g(S)$。 □

因此我们只需要求出 h 数组，然后对 h 做集合幂级数 ln 运算，便可以对于每个 P 计算出 $coef(P)$。此部分时间复杂度为 $O(2^n \cdot n^2)$。总时间复杂度为 $O(2^n \cdot n^2 log_2 C + Bell(n))$。

空间复杂度为 $O(2^n \cdot n)$。

根据实现的好坏，可以通过前七至九个测试点，期望得分 70 – 80 分。

3.5 算法5

算法3.2和**算法4.3**对于容斥系数的计算已经足够优秀了，可这两种算法在计算答案时都需要枚举点的划分，需要 $Bell(n)$ 的时间复杂度，在 n 大时无法接受。考虑优化统计答案的部分。

3.5.1 算法5.1

在枚举划分时，设仍未被划分的点集为 S，则我们每次会枚举 $T \subset S, T \neq \emptyset$，将 T 加入划分中，并根据 T 的奇偶性做相应的更新。观察一种划分 P 对答案的贡献为 P 中大小为奇数联通块的 F 值 $\times P$ 中大小为偶数联通块最小值 + 1 的乘积 $\times P$ 中每个子集的容斥系数的乘积。考虑通过动态规划来解决。

令 $sum(S, T)$ 表示 S 集合内的点已经被划分，大小为奇数的联通块中最小值点集为 T，大小为偶数的联通块最小值 + 1 的乘积 $\times$ 已划分所有联通块的系数乘积的和。

在代码实现时，可以发现 $T \subset S$，故可以用三进制数表示 S, T 的状态。具体地，对于一个点 u，令 $b(u) = \begin{cases} 1, & u \in T \\ 2, & u \in (S \setminus T) \\ 0, & otherwise \end{cases}$，$dp(\sum_{u=1}^{n} 3^{u-1} b(u)) = sum(S, T)$。$dp$ 的转移直接枚举 $b(u) = 0$ 的点集的子集即可。

dp 总状态数为 $O(3^n)$。计算其转移复杂度，对于一个状态 $\sum_{u=1}^{n} 3^{u-1} b(u)$，如果将转移时枚举的点集中的所有点 $b(u) = 0$ 看作 $b(u) = 3$，那么一种转移方式对应一个 n 位 4 进制数，故转移时间复杂度为 $O(4^n)$。总时间复杂度为 $O(2^n \cdot n^2 log_2 C + 4^n)$.

空间复杂度为 $O(3^n + 2^n \cdot n)$。

可以通过全部测试点，期望得分 100 分。

3.5.2 算法5.2

令 $V = \{1, 2, ..., n\}$，可以注意到，一种划分，就是若干个两两交集为空、并集为 V 的 V 的子集，这不禁让我们想到用集合幂级数来帮我们解决问题。

对于 $S \subset V$，设 u 为 S 中 *limit* 最小的点，定义 $min(S) = \begin{cases} \{u\}, & |S| = 1 \bmod 2 \\ \emptyset, & |S| = 0 \bmod 2 \end{cases}$，

令 $g(S, T) = \begin{cases} 0, & T \neq min(S) \\ coef(S), & T = min(S), |T| > 0 \\ coef(S) \cdot (limit_u + 1), & T = min(S), |T| = 0 \end{cases}$。观察 $sum(S, T)$，可以发现 $sum(S, T) = \sum_{\{X_1, X_2, ..., X_k\}, \{Y_1, Y_2, ..., Y_k\}} [\cup_{i=1}^{k} X_i = S][\cup_{i=1}^{k} Y_i = T][\forall i \neq j, X_i \cap X_j = \emptyset][\forall i \neq j, Y_i \cap Y_j = \emptyset] \Pi_{i=1}^{k} g(X_i, Y_i)$。那么将 sum, g 看成集合幂级数，则 $sum = e^g$。

此处集合幂级数运算不同于大多数题目中只有一个集合做子集卷积，在实现时可以采用如下方法。与**算法5.1**中一样，将 (S, T) 转化成 $X = \sum_{u=1}^{n} 3^{u-1} b(u)$。令 $bits3(X) = \sum_{u=1}^{n} [b(u) > 0]$。定义集合 $S(0), S(1), S(2)$，$S(0) = \{0\}, S(1) = \{0, 1\}, S(2) = \{0, 2\}$。对每一个 $X = \sum_{u=1}^{n} 3^{u-1} b(u)$ 定义集合 $trans(X) = \{\sum_{u=1}^{n} 3^{u-1} c(u) \mid c(u) \in S(b(u))\}$。此情况下，莫比乌斯变换为 $G(X, k) = \sum_Y [bits3(Y) = k][Y \in trans(X)] g(Y)$，莫比乌斯反演为其逆变换。求 $sum = e^g$ 过程为：对 g 做莫比乌斯变换，对每一项做形式幂级数 exp 运算，然后做莫比乌斯反演便得到 sum。

时间复杂度为 $O(2^n \cdot n^2 log_2 C + 3^n \cdot n^2)$。

空间复杂度为 $O(3^n \cdot n)$。

然而在 n 较小时，实际运行效率不如**算法5.1**，只能通过前七个测试点，期望得分 70 分。

4 总结

本题主要有以下几部分:

1. 对 $m = 0$ 问题的求解。这部分主要在**算法2**中讨论。

   算法2.1和**算法2.2**为较为经典的动态规划题型，而要得到**算法2.3**需要选手观察并利用 $\oplus$ 这一位运算的性质，以及从最大数的最高位这一特别位置的角度入手，再结合所填数必须 $\leq$ *limit* 这一限制，对具体数值情况加以分

析讨论。这种技巧不是很为常见，打破了以往用数位动态规划解决类似问题的常规套路。

2. 通过容斥的方法将 $m > 0$ 的问题转化成若干个 $m = 0$ 的问题加以解决。这部分主要在**算法3**、**算法4**、**算法5**中讨论。

   所填数有限制这类计数题目近几年开始出现，强制一些限制不满足而忽视其它限制这一容斥方法也应运而生。也希望本题能对这一容斥方法产生推广作用。

   在想到用容斥解决问题之后，本题还考察了以下两方面：

   (a) 容斥系数的计算。这部分主要在**算法3**、**算法4**中讨论。

   虽然已经有一些通过这类容斥方法解决的题目，但较多题目仍只要求任意两数不能相等，即本题中 $m = \frac{n(n-1)}{2}$ 的情况。这一情况较为特殊，故相应的容斥系数较为固定。本题在此基础上进行了拓展。比较**算法3.2**和**算法4.3**可以发现，这两种算法在容斥时都是枚举点集 S 要求其内部点 col 值相同，从本质上来说是一样的。**算法3.2**中容斥系数的计算方法也可以通过集合幂级数算法来优化。从两个不同角度入手解决此题，给予了选手发挥的空间，希望能能帮助选手更深入的理解容斥系数。

   在理解了容斥基础上，需要对容斥系数进行快速计算。在**算法4.3**中提及的将容斥系数看成集合幂级数的思想值得借鉴。

   (b) 计算出容斥系数后优化枚举子集划分。这部分主要在**算法5**中讨论。

   算法5.2中从集合幂级数的角度思考子集划分，在具体代码实现时，此处用到的集合幂级数并不同于大多数题目中只有一个子集，这也考察了选手集合幂级数方面知识的理解和掌握。

从题目总体上来说，正解的过程并不多，代码量不大，但思维难度不低，考察范围较广。作为一道OI题，在部分分的设计上鼓励了多种方法解决此问题。

本人希望能以这题起到抛砖引玉的作用，引发大家对于容斥、集合幂级数等算法的思考。如果有人能想出更优秀的做法或对某一部分有不同的理解，欢迎与我交流。