胡策的统计 解题报告

湖北省武汉市第二中学 吕凯风

1 试题来源

集训队互测 2015

2 试题大意

在 OI 界，有一位无人不知无人不晓，OI 水平前无古人后无来者的胡策，江湖人称一眼秒题胡大爷!

今天胡策在研究无向图的连通性。对于一个无向图定义它的连通值为该图连通块数的阶乘。

为了研究连通值的性质，胡策随手画了一个 n 个结点的简单无向图 G，结点分别编号为 $1, \ldots, n$，他想统计出 G 的所有生成子图的连通值之和。

胡策当然会做啦！但是他想考考你。你只用输出结果对 998244353 （$7 \times 17 \times 2^{23} + 1$，一个质数）取模后的结果。

简单无向图即无重边无自环的无向图。生成子图即原图中删去若干条边（可以是 0 条）后形成的图。

<table>
<tr><th>编号</th><th>n</th><th>特殊限制</th></tr>
<tr><td>1</td><td>≤ 6</td><td rowspan="6">无</td></tr>
<tr><td>2</td><td rowspan="2">≤ 10</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td rowspan="3">≤ 17</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td></tr>
<tr><td>7</td><td rowspan="4">≤ 20</td><td>G 为完全图</td></tr>
<tr><td>8</td><td rowspan="3">无</td></tr>
<tr><td>9</td></tr>
<tr><td>10</td></tr>
</table>

3 算法介绍

本题中，要求的是 G 的所有生成子图的连通块个数的阶乘之和。

设原图为 $G = (V, E)$，V 表示图的点集，E 表示图的边集。

符号 $[P]$ 表示 P 成立时为 1 否则为 0。

3.1 算法一

对于 10% 的数据，$n \le 6$。这意味着 $m \le 15$。

我们可以 $O(2^m)$ 枚举生成子图，然后 $O(m)$ 计算连通值。

时间复杂度 $O(m2^m)$，可以通过 1 号测试点获得 10 分。

3.2 算法二

对于 30% 的数据，$n \le 10$。

我们可以使用 DP 来解决。我们设 $m(S)$ 表示由点集 S 导出的子图中的边数。设 $f_1(S)$ 表示由点集 S 导出的子图的生成子图中连通图的数量。我们可以补集转化一下，转为统计不为连通图的数量。我们只要枚举 S 中编号最小的那个结点 v 所在连通块就可以得到递推式：

$$f_1(S) = 2^{m(S)} - \sum_{T \subsetneq S} [v \in T] f_1(T) 2^{m(S-T)} \tag{1}$$

设 $f_c(S)$ 表示由点集 S 导出的子图的生成子图中连通块恰为 c 的图的数量。我们只要枚举 S 中编号最小的那个结点 v 所在连通块就可以得到递推式：

$$f_c(S) = \sum_{T \subsetneq S} [v \in T] f_1(T) f_{c-1}(S - T) \tag{2}$$

最后 $\sum_{c=1}^{n} f_c(V) \cdot c!$ 就是答案。

由于 $\sum_{k=0}^{n} \binom{n}{k} 2^k = 3^k$，所以枚举子集的复杂度是 $O(3^n)$。

这样我们就得到了时间复杂度 $O(n3^n)$ 的算法，可以通过前 3 个测试点获得 30 分。

3.3 算法三

对于 60% 的数据，$n \leq 17$。

我们直接设 $g(S)$ 为由点集 S 导出的子图的生成子图的连通值之和，连通值我们可以理解成连通块的排列数。我们可以枚举排列中的第一个连通块：

$$g(S) = f_1(S) + \sum_{T \subsetneq S, T \neq \varnothing} f_1(T) g(S - T) \tag{3}$$

这样我们就得到了时间复杂度 $O(3^n)$ 的算法，可以通过前 6 个测试点获得 60 分。

3.4 算法四

有 10% 的数据，G 是完全图。

此时，考虑我们刚才的 DP 状态，S 中元素相同时 DP 值总是相同的。所以我们可以只按大小进行 DP。这样就得到了 $O(n^2)$ 的算法。

可以通过 7 号测试点获得 10 分。结合算法三可以获得 70 分。

3.5 算法五

我们定义集合多项式[1]为一个定义域为集合值域为某个域 F 的函数 f。

我们可以很轻松地定义加减法，即对应的函数值相加减。我们也可以定义一个集合多项式乘以一个 F 中的元素 c，即每个函数值乘以 c。

[1]我并没有查到正式名称，这只是我自己取的名字。

定义两个集合多项式 f 和 g 的乘法为子集卷积，即，若 $h = fg$ 则：

$$h(S) = \sum_{T \subseteq S} f(T)g(S - T) \tag{4}$$

易证乘法具有交换律和对加法的分配律。

我们可以把 F 中的元素 c 看作一个集合多项式，即空集时映射为 c，非空集映射为 0。易证 0 是加法单位元，1 是乘法单位元。

我们可以把乘法的式子变一下，就得到当 $f(\varnothing) \neq 0$ 时 f 的乘法逆元的每个函数值的递推式：

$$g(S) = \frac{1}{f(\varnothing)}\left(h(S) - \sum_{T \subsetneq S} f(T)g(S - T)\right) \tag{5}$$

于是就证明了 $f(\varnothing) \neq 0$ 时一定有乘法逆元。

这样我们就证明了集合多项式是个交换环。集合多项式具有跟多项式类似的性质，我们就可以把集合多项式当作母函数使用。

我们来看原问题。设 $b(S) = 2^{m(S)}$，那么我们可以发现：

$$1 + b = \sum_{k \geq 0} \frac{f^k}{k!} = e^f \tag{6}$$

这个式子的左边表示的是每个点集的导出子图的生成子图的数量，右边是枚举生成子图的连通块个数再除以连通块个数的阶乘去重，所以左右相等。右边就是 e^f 的幂级数形式，我们可以把右边记作 e^f。

那么我们解得：

$$f = \ln(1 + b) = \sum_{k \geq 1} \frac{(-1)^{k+1}}{k} b^k \tag{7}$$

而最后要求的结果是：

$$\sum_{k \geq 0} f^k = \frac{1}{1 - f} \tag{8}$$

所以这个问题就是要你求：

$$\frac{1}{1 - \ln(1 + b)} \tag{9}$$

那么怎么进行子集卷积呢？$O(3^n)$ 显然是可做的。下面介绍一个 $O(n^2 2^n)$ 的算法。

考虑另外一个问题。我们定义两个集合多项式的子集并卷积[2]，已知 f 和 g，求 h 满足：

$$h(C) = \sum_{A \subseteq C} \sum_{B \subseteq C} [A \cup B = C] f(A) g(B) \tag{10}$$

考虑两边同时求子集和，则：

$$\begin{aligned}
\sum_{C \subseteq S} h(C) &= \sum_{C \subseteq S} \sum_{A \subseteq C} \sum_{B \subseteq C} [A \cup B = C] f(A) g(B) &(11)\\
&= \sum_{A \subseteq S} \sum_{B \subseteq S} f(A) g(B) &(12)\\
&= \left(\sum_{A \subseteq S} f(A) \right) \left(\sum_{B \subseteq S} g(B) \right) &(13)
\end{aligned}$$

所以解决这个问题我们只需要对 f 和 g 作子集和然后乘起来，作子集和的逆变换。

我们可以把子集和理解成高维前缀和，每一维 $\{0, 1\}$ 表示元素是否存在。然后我们可以依次枚举每一维做前缀和，最后然后就能在 $O(n2^n)$ 时间内求子集和。逆变换就是把这个过程反过来。

对于子集卷积，我们要考虑的问题是已知 f 和 g，求 h 满足：

$$h(C) = \sum_{A \subseteq C} \sum_{B \subseteq C} [A \cup B = C][A \cap B = \varnothing] f(A) g(B) \tag{14}$$

其实我们可以换一个等价表述：

$$h(C) = \sum_{A \subseteq C} \sum_{B \subseteq C} [A \cup B = C][|A| + |B| = |C|] f(A) g(B) \tag{15}$$

所以我们可以设 f' 和 g'，它们都是值域为多项式的集合多项式。其中 $f'(S) = f(S)x^{|S|}$，$g'(S) = g(S)x^{|S|}$。现在我们把 f' 和 g' 用子集并卷积乘起来得到 h'，然后 $h(S)$ 的值即为 $x^{|S|}$ 在多项式 $h'(S)$ 中的系数。时间复杂度即 $O(n^2 2^n)$

注意到 f' 并不需要那么严格，只要 $f(S)$ 中 $x^0, x^1, \ldots, x^{|S|-1}$ 前的系数都是 0 且 $x^{|S|}$ 前的系数是 $f(S)$ 那么这个算法就能成功运行。我们称这种与 f 对应的集合多项式 f' 为集合占位多项式[3]。

[2]我并没有查到正式名称，这只是我自己取的名字。

[3]这只是我自己取的名字。

因此，我们如果想要把三个集合多项式乘起来，只需要分别写出对应的集合占位多项式然后分别做子集和，然后再把对应函数值的多项式乘起来，然后再做子集和的逆变换就行了，而不必分别做两次子集卷积。

注意到加减法也可以在集合占位多项式上直接进行操作，所以我们就得到一个推论：设一个多项式 T，如果你想对于一个集合多项式 f 求 $T(f)$，那么你只需要写出 f 的子集占位多项式然后做子集和，然后把函数值（函数值是多项式）分别带入 T，然后做子集和的逆变换就行了。

所以原问题也就迎刃而解。原问题中，$T(x) = \frac{1}{1-\ln(1+x)}$，瓶颈在于对于一个多项式 f 求 $T(f)$，你可以用一个 $O(n^2)$ 的递推求出。

于是我们就得到了一个 $O(n^2 2^n)$ 的优美算法，可以获得 100 分。

3.6 算法六

把算法五最后的程序写出来，我们发现可以从组合的意义上理解整个算法。

容易分析出，最终结果一定能写成一些项之和，每项都是 b 的某些不相交子集的函数值乘起来再乘上某个系数。

我们首先对每个点集 S 和 k 求出结点数为 k 的子图的数目。注意这里不必是生成子图。接下来，我们跑算法四里的那个 DP。

注意到这个 DP 会算错。因为这个忽略了具体集合形态只考虑集合大小的 DP 会多算一些项，这些项中含有某两个子集相交。怎么办？考虑最后算出的结果，对于 S 和 k 我们能知道子集都是 S 的子集，且子集大小之和为 k 的函数值乘积乘以某个系数求和后的值，而我们只希望保留子集并大小恰好为 k 的项。于是我们就可以用容斥解决。

这样我们就从另一个角度诠释了算法五，算法并没有变，可以获得 100 分。或许有组合数学能力超强的选手能够直接想到该算法。

3.7 得分估计

在集训队互测中，预计大部分人能拿到 70 分，少部分人拿到 30 分。预计有 0 ~ 2 人获得 100 分。