生成树的计数及其应用

目录

摘要

有关生成树的最优化问题如最小生成树等是我们经常遇到的，而对生成树的计数及其相关问题则少有涉及。事实上，生成树的计数是十分有意义的，在许多方面都有着广泛的应用。首先介绍了一种指数级的动态规划算法，然后介绍了行列式的基本概念、性质，并在此基础上引入 *Matrix-Tree* 定理，同时通过与一道数学问题的对比，揭示了该定理所包含的数学思想。最后通过几道例题介绍了生成树的计数的应用，并进行总结。

关键字

生成树的计数 *Matrix-Tree* 定理

问题的提出

[例一]高速公路（SPOJ p104 Highways）

一个有 n 座城市的组成国家，城市 1 至 n 编号，其中一些城市之间可以修建高速公路。现在，需要有选择的修建一些高速公路，从而组成一个交通网络。你的任务是计算有多少种方案，使得任意两座城市之间恰好只有一条路径？

数据规模：$1 \le n \le 12$。

[分析]

我们可以将问题转化到成图论模型。因为任意两点之间恰好只有一条路径，所以我们知道最后得到的是原图的一颗生成树。因此，我们的问题就变成了，给定一个无向图 G，求它生成树的个数 $t(G)$。这应该怎么做呢？

经过分析，我们可以得到一个时间复杂度为 $O(3^n * n^2)$的动态规划算法，因为原题的规模较小，可以满足要求。但是，当 n 再大一些就不行了，有没有更优秀的算法呢？答案是肯定的。在介绍算法之前，首先让我们来学习一些基本的预备知识。

预备知识

下面，我们介绍一种重要的代数工具——行列式。为了定义行列式，我们首先来看一下排列的概念。

排列

定义 1 由 1，2，…，n 组成的一个有序数组 $i_1i_2\cdots i_n$ 称为 1，2，…，n 的一个排列。

由排列的定义可知，i_1，i_2，…，i_n 表示了 n 个不同的自然数，同时 i_1，i_2，…，i_n 中的每个自然数都是集合 S_n={1，2，…，n}中的一个元素，换句话说，

$$1\to i_1,2\to i_2,\Lambda\ ,\ n\to i_n$$

定义了集合 S_n 到自身上的一个一一对应。这个一一对应可以用符号

$$\begin{pmatrix}1 & 2 & \Lambda & n\\ i_1 & i_2 & \Lambda & i_n\end{pmatrix}$$

记之，称为**置换**，而上述一一对应可以改写为

$$j\to i_{j_1},\ j_2\to i_{j_2},\Lambda\ ,\ j_n\to i_{j_n}$$

其中 $j_1j_2\cdots j_n$ 是 1，2，…，n 的一个排列。所以这个一一对应也可以用符号

$$\begin{pmatrix}j_1 & j_2 & \Lambda & j_n\\ i_{j_1} & i_{j_2} & \Lambda & i_{j_n}\end{pmatrix}$$

记之，因此对 1，2，…，n 的任一排列 $j_1j_2\cdots j_n$，可定义

$$\begin{pmatrix}1 & 2 & \Lambda & n\\ i_1 & i_2 & \Lambda & i_n\end{pmatrix}=\begin{pmatrix}j_1 & j_2 & \Lambda & j_n\\ i_{j_1} & i_{j_2} & \Lambda & i_{j_n}\end{pmatrix}$$

任取一个排列 $i_1i_2\cdots i_n$，将其中两个相邻的自然数 i_{j-1}，i_j 对换一下，便造出一个新的排列 $i_1i_2\cdots i_{j-2}i_ji_{j-1}\ i_{j+1}\cdots i_n$，称为原来排列的**对换排列**，这样一种步骤成为**对换**。显然，对于任何一个排列经过若干次对换后都可以变成标准排列 12…n。不过，不管经过什么途径作对换，在给定排列 $i_1i_2\cdots i_n$ 后，关于对换的次数有下列重要定理。

定理 1 将任意一个排列 $i_1i_2\cdots i_n$ 通过对换变成标准排列 12…n，所需的对换次数的奇偶性与对换方式无关。

利用这个定理，我们引入

定义 2 一个排列 $i_1i_2\cdots i_n$ 称为**偶（奇）排列**，如果有一种方式，经过偶（奇）数次对换后，可以将排列 $i_1i_2\cdots i_n$ 变为标准排列 12…n。

设排列 $i_1i_2\cdots i_n$ 经过 t 次对换后变为标准排列 12…n，则数值$(-1)^t$和对换方式无关。将它改写成$\delta\begin{matrix}12\Lambda\ n\\ i_1i_2\Lambda\ i_n\end{matrix}$，即

$$\delta\begin{matrix}12\Lambda\ n\\ i_1i_2\Lambda\ i_n\end{matrix}=(-1)^t=\begin{cases}1,\ \text{当}i_1i_2\Lambda\ i_n\text{是}1,2,\Lambda\ ,\ n\text{的偶排列时；}\\ -1,\ \text{当}i_1i_2\Lambda\ i_n\text{是}1,2,\Lambda\ ,\ n\text{的奇排列时。}\end{cases}$$

n 个确定自然数 1，2，…，n 的排列，可以看作是集合 S_n={1，2，…，n}到自身上的一个一一对应。将这个概念推广，任取 n 个元素的集合 S={a_1，a_2，…，a_n}。对于集合 S 到自身上的一一对应

$$a_1 \to a_{i_1},\ a_2 \to a_{i_2},\Lambda,\ a_n \to a_{i_n}$$

$a_{i_1}a_{i_2}\Lambda\, a_{i_n}$ 称为 a_1，a_2,Λ，a_n 的一个排列。容易看出，$a_{i_1}a_{i_2}\Lambda\, a_{i_n}$ 是 a_1，a_2,Λ，a_n 的排列的充要条件是 $i_1i_2\cdots i_n$ 是 1，2，…，n 的排列。

同样，排列 $a_{i_1}a_{i_2}\Lambda\, a_{i_n}$ 变为标准排列 $a_1a_2\Lambda\, a_n$ 的对换总次数的奇偶性和对换方式无关，因此引入符号

$$\delta\begin{matrix}a_1a_2\Lambda\, a_n\\ a_{i_1}a_{i_2}\Lambda\, a_{i_n}\end{matrix}=(-1)^t=\begin{cases}1,\ \text{当}a_{i_1}a_{i_2}\Lambda\, a_{i_n}\text{是}a_1,a_2,\Lambda\,,a_n\text{的偶排列时;}\\ -1,\ \text{当}a_{i_1}a_{i_2}\Lambda\, a_{i_n}\text{是}a_1,a_2,\Lambda\,,a_n\text{的奇排列时。}\end{cases}$$

其中 t 是某一种将 $a_{i_1}a_{i_2}\Lambda\, a_{i_n}$ 变为 $a_1a_2\Lambda\, a_n$ 的对换方式的对换总次数。

下面我们介绍行列式。

行列式

一阶行列式是一个变量 a_{11} 的函数 $\det(a_{11})=a_{11}$，也可以改写成为

$$\det(a_{11})=a_{11}=\sum_{\binom{1}{1}}\delta\begin{matrix}1\\1\end{matrix}a_{11}$$

二阶行列式是四个变量 a_{11}，a_{12}，a_{21}，a_{22} 的函数 $\det\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}=a_{11}a_{22}-a_{12}a_{12}$，也可以改写成

$$\det\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}=\sum_{\binom{12}{i_1i_2}}\delta\begin{matrix}12\\i_1i_2\end{matrix}a_{1i_1}a_{2i_2}$$

三阶行列式是九个变量 a_{ij}(i，j=1，2，3)的函数

$$\det\begin{pmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{pmatrix}=a_{11}a_{22}a_{33}+a_{12}a_{23}a_{31}+a_{13}a_{21}a_{32}-a_{13}a_{22}a_{31}-a_{12}a_{21}a_{33}-a_{11}a_{23}a_{32}$$

，同样可以改写成为

$$\det\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \sum_{\binom{123}{i_1 i_2 i_3}} \delta \begin{matrix} 123 \\ i_1 i_2 i_3 \end{matrix} a_{1i_1} a_{2i_2} a_{3i_3}$$

通过观察一、二、三阶行列式的定义，我们得出了 n 阶行列式的一般定义。

定义 3 n 阶矩阵 A 的行列式是一实数，记作 $\det A$，它定义为

$$\det A = \sum_{\binom{12\Lambda\, n}{i_1 i_2 \Lambda\, i_n}} \delta \begin{matrix} 12\Lambda\, n \\ i_1 i_2 \Lambda\, i_n \end{matrix} a_{1i_1} a_{2i_2} \Lambda\, a_{ni_n}$$

行列式有下列几种常用的符号：

$$\det A = \det\begin{pmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{pmatrix} = \begin{vmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix} = |A|$$

由行列式的定义可知，利用定义直接计算行列式是很困难的，只有在阶数低时才可以直接用定义计算。为了能够进行计算，需要先导出行列式的若干基本性质，在通过这些性质，将复杂的行列式计算化为简单的行列式计算，也可以将高阶行列式的计算化为低阶行列式的计算。

性质 1 设 A 是 n 阶矩阵，则

$$\det A^T = \det A。$$

这个定理的重要意义在于，它告诉我们行列式的行和列的地位是平等的。确切地说，每个关于行的性质，对列必然成立；反之亦然。

性质 2 行列式的任两行（或两列）互换，行列式变号。

推论 行列式的某两行（或两列）相同时，行列式的值为 0。

性质 3 用实数 λ 乘以一行（或列）后，所得到的行列式等于原来的行列式乘以 λ。

推论 行列式有两行（或两列）成比例，则该行列式的值为零。特别的，当行列式有一行（或列）全为零时，行列式的值为零。

性质 4

$$\begin{vmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{j-1,1} & \Lambda & a_{j-1,n} \\ a_{j1}+b_{j1} & \Lambda & a_{jn}+b_{jn} \\ a_{j+1,1} & \Lambda & a_{j+1,n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{j-1,1} & \Lambda & a_{j-1,n} \\ a_{j1} & \Lambda & a_{jn} \\ a_{j+1,1} & \Lambda & a_{j+1,n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{j-1,1} & \Lambda & a_{j-1,n} \\ b_{j1} & \Lambda & b_{jn} \\ a_{j+1,1} & \Lambda & a_{j+1,n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix}$$

对列也有同样性质。

推论 将行列式的任意行（或一列）乘以实数 λ，再相应地加到另一行（或另一列）上去，则行列式的值不变。例如，当 $j\neq 1$ 时

$$\begin{vmatrix} a_{11}+\lambda a_{j1} & \Lambda & a_{1n}+\lambda a_{jn} \\ a_{21} & \Lambda & a_{2n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & \Lambda & a_{1n} \\ a_{21} & \Lambda & a_{2n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix}$$

下面我们介绍一种计算行列式的方法。首先，不难证明

$$\begin{vmatrix} a_{11} & 0 & \Lambda & 0 \\ a_{21} & a_{22} & \Lambda & a_{2n} \\ \mathrm{M} & \mathrm{M} & & \mathrm{M} \\ a_{n1} & a_{n2} & \Lambda & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \Lambda & a_{1n} \\ 0 & a_{22} & \Lambda & a_{2n} \\ \mathrm{M} & \mathrm{M} & & \mathrm{M} \\ 0 & a_{n2} & \Lambda & a_{nn} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & \Lambda & a_{2n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{vmatrix}$$

今取行列式

$$\det B = \begin{vmatrix} b_{11} & \Lambda & b_{1n} \\ \mathrm{M} & & \mathrm{M} \\ b_{n1} & \Lambda & b_{nn} \end{vmatrix}$$

我们可以将 B 化为上三角形式，记为 B'，同时记录行交换次数 S。根据定理 11，易知：

$$\det B = (-1)^S \det B' = \begin{vmatrix} b'_{11} & b'_{12} & \Lambda & b'_{1n} \\ 0 & b'_{22} & \Lambda & b'_{2n} \\ \mathrm{M} & \mathrm{O} & \mathrm{O} & \mathrm{M} \\ 0 & \Lambda & 0 & b'_{nn} \end{vmatrix}$$

再利用刚才的结论，得 $\det B = (-1)^S \prod_{i=1}^{n} b'_{ii}$。

对于矩阵 A，任取 $2p$ 个自然数

$$1\leq i_1<i_2<\cdots<i_p<=n，\ 1\leq j_1<j_2<\cdots<j_p<=n$$

在矩阵 A 中取出第 i_1，i_2，…，i_p 行第 j_1，j_2，…，j_p 列的交叉元素，按原来的次序排成 p 阶矩阵。这个矩阵的行列式称为矩阵 A 的 ***p* 阶子式**，记作

$$A\begin{pmatrix} i_1 & i_2 & \Lambda & i_p \\ j_1 & j_2 & \Lambda & j_p \end{pmatrix} = \begin{vmatrix} a_{i_1 1} & \Lambda & a_{i_1 j_p} \\ \mathrm{M} & & \mathrm{M} \\ a_{i_p 1} & \Lambda & a_{i_p j_p} \end{vmatrix}$$

根据这个定义，下面我们再介绍中一个非常重要的公式——*Binet-Cauchy* 公式。

定理 8(*Binet-Cauchy* 公式) 设 *A* 和 *B* 各为 *p***q* 及 *q***p* 矩阵，则有

$$\det AB = \begin{cases} 0, & \text{当} p > q \text{时；} \\ \det A \det B & \text{当} p = q \text{时；} \\ \sum_{1<=j_1<\Lambda<j_p<=q} A\begin{pmatrix} 1 & 2 & \Lambda & p \\ j_1 & j_2 & \Lambda & j_p \end{pmatrix} B\begin{pmatrix} j_1 & j_2 & \Lambda & j_p \\ 1 & 2 & \Lambda & p \end{pmatrix} & \text{当} p < q \text{时。} \end{cases}$$

特别的，$\det AA^T=(\det A)^2$。

新的方法

介绍

下面我们介绍一种新的方法——*Matrix-Tree* 定理(Kirchhoff 矩阵-树定理)。*Matrix-Tree* 定理是解决生成树计数问题最有力的武器之一。它首先于 1847 年被 Kirchhoff 证明。在介绍定理之前，我们首先明确几个概念：

1、*G* 的度数矩阵 *D*[*G*]是一个 *n***n* 的矩阵，并且满足：当 *i*≠*j* 时,d_{ij}=0；当 *i*=*j* 时，d_{ij} 等于 v_i 的度数。
2、*G* 的邻接矩阵 *A*[*G*]也是一个 *n***n* 的矩阵， 并且满足：如果 v_i、v_j 之间有边直接相连，则 a_{ij}=1，否则为 0。

我们定义 *G* 的 Kirchhoff 矩阵(也称为拉普拉斯算子)*C*[*G*]为 C[*G*]=*D*[*G*]-*A*[*G*]，则 *Matrix-Tree* 定理可以描述为：*G* 的所有不同的生成树的个数等于其 Kirchhoff 矩阵 *C*[*G*]任何一个 *n*-1 阶主子式的行列式的绝对值。所谓 *n*-1 阶主子式，就是对于 *r*(1≤*r*≤*n*)，将 *C*[*G*]的第 *r* 行、第 *r* 列同时去掉后得到的新矩阵，用 C_r[*G*]表示。

下面我们举一个例子来解释 *Matrix-Tree* 定理。如图 *a* 所示，*G* 是一个由 5 个点组成的无向图。

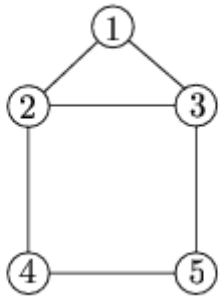

图 *a*

根据定义，它的 Kirchhoff 矩阵 $C[G]$为

$$\begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

我们取 r=2，则有：

$$|C_2[G]| = \begin{vmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 \\ 0 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{vmatrix}$$

$$= 2\cdot\begin{vmatrix} 3 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 2 \end{vmatrix} - (-1)\cdot\begin{vmatrix} -1 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{vmatrix} + 0\cdot\begin{vmatrix} -1 & 3 & -1 \\ 0 & 0 & -1 \\ 0 & -1 & 2 \end{vmatrix} - 0\cdot\begin{vmatrix} -1 & 3 & 0 \\ 0 & 0 & 2 \\ 0 & -1 & -1 \end{vmatrix}$$

$$= 2\left(3\cdot\begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} - 0\cdot\begin{vmatrix} \cdot & \cdot \\ \cdot & \cdot \end{vmatrix} + (-1)\cdot\begin{vmatrix} 0 & 2 \\ -1 & -1 \end{vmatrix}\right) - (-1)\left((-1)\cdot\begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} - 0\cdot\begin{vmatrix} \cdot & \cdot \\ \cdot & \cdot \end{vmatrix} + (-1)\cdot\begin{vmatrix} 0 & 2 \\ 0 & -1 \end{vmatrix}\right)$$

$$= 2(3(4-1)-0+(-1)(0-(-2)))-(-1)((-1)(4-1)-0+(-1)(0-0))+0-0$$

$$= 2(9-2)-(-1)(-1)(3)=11$$

这 11 棵生成树如图 *b* 所示。

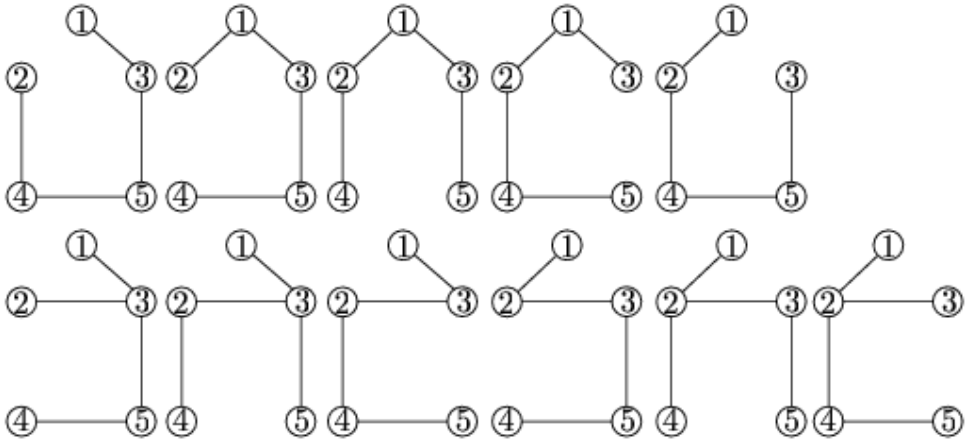

图 *b*

这个定理看起来非常“神奇”，只需要计算一个矩阵的行列式，就可以得到这个图不同的生成树的个数！这是为什么呢？让我们尝试着来证明一下吧。

证明

我们不难看出，这个定理的关键是图的 Kirchhoff 矩阵。这个矩阵有什么特殊的性质呢？经过分析，我们可以发现：

1、对于任何一个图 G，它的 Kirchhoff 矩阵 C 的行列式总是 0。这是因为 C 每行每列所有元素的和均为 0。

2、如果 G 是不连通的，则它的 Kirchhoff 矩阵 C 的任一个主子式的行列式均为 0。

证明：

如果 G 中存在 $k(k>1)$个连通分量 G_1，G_2，…，G_k，那么，我们可以重新安排 C 的行和列使得属于 G_1 的顶点首先出现，然后是 G_2……。回忆行列式的性质 2，设我们一共进行了 t 次行交换，显然，我们同样需要进行 t 次列交换。因此，我们总共进行了 $2t$ 次交换，所以，行列式的符号没有改变。

重新安排后的矩阵如下图所示：

$$\begin{pmatrix} \boxed{G_1} & 0 & \cdot & \cdot & 0 \\ 0 & \boxed{G_2} & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \boxed{G_k} \end{pmatrix}$$

带方框的 G_i 表示了 $C[G_i]$。注意在方框以外的地方都是 0 因为任意两个连通分量之间没有边相连。

设 r 是第 i 个连通分量中的第 j 个顶点，那么

$$\left|C_r[G]\right| = \left|C[G_1]\right| \cdot \left|C[G_2]\right| \cdot \Lambda \cdot \left|C_j[G_i]\right| \Lambda \cdot \left|C[G_k]\right|$$

$$= 0 \cdot 0 \cdot \Lambda \cdot \left| C_j[G_i] \right| \Lambda \cdot 0 = 0$$

3、如果 G 是一颗树，那么它的 Kirchhoff 矩阵 C 的任一个 n-1 阶主子式的行列式均为 1。

证明：

为了证明这条重要的性质，我们通过不断的变换 $C_r[G]$从而得到一个上三角矩阵并且使得主对角线上所有的数字都是 1。

第一步：我们把所有的顶点按照在树上离 v_r 的距离从近至远排列。首先是 v_r，然后是离 v_r 距离为 1 的点，接下来是离 v_r 距离为 2 的点……。距离相同的点可以任意排列。我们按照这个顺序将所有的定点重新编号。同时，把以 r 为根的有根树上 v_i 的父结点 v_j 称为 v_i 的父亲，显然，在刚才得到的顺序中，v_j 出现在 v_i 之前。

第二步：将 $C_r[G]$的 n-1 行和 n-1 列按照刚才得到的顺序重新排列。因为行列都需要交换，所有总交换次数是偶数，$C_r[G]$的行列式不边。

第三步：按照刚才得到的顺序的逆序处理，对于每个 i，如果 v_i 的父亲 v_j 不等于 v_r，则把第 i 列加到第 j 列上去。

这样处理过之后，我们来看一下现在的 $C_r[G]$是否和我们的希望相同。我们通过归纳法来证明。显然，最后一列符合要求，因为最后一个点只和它的父亲之间有边，它的度为 1。假设当前处理的是第 i 列，并且第 i+1 至第 n-1 列都符合要求。我们设 v_i 的父亲是 v_j，它有 k 个孩子 v_{i_1}，$v_{i_2}\Lambda$，v_{i_k}，显然，只有第 i_1，i_2，…，i_k 列才会加到第 i 列上来。所有这些列，都在第 i 行有个-1 而在对角线上有个 1，而第 i 列在第 j、i_1，i_2，…，i_k 行上为-1 而在第 i 行上为 k+1。每一列加上来的时候，第 i 行第 i 列减一同时与之相对的那一列加一变为 0。最后第第 i 行第 i 列变为 1，第 i_1，i_2，…，i_k 行变为 0。也就是说，最后 $C_r[G]$会变成上三角矩阵并且主对角线上所有的数字都是 1。这就证明了我们的结论。

在证明 *Matrix-Tree* 定理的过程中，我们使用了无向图 G 的关联矩阵 B。B 是一个 n 行 m 列的矩阵，行对应点而列对应边。B 满足，如果存在一条边 e=$\{v_i,v_j\}$，那在 e 所对应的列中，v_i 和 v_j 所对应的那两行，一个为 1、另一个为-1，其他的均为 0。至于谁是 1 谁是-1 并不重要，如下图所示。

$$\begin{array}{c} \\ v_i \\ \\ v_j \\ \\ \end{array} \begin{vmatrix} & e & \\ \vdots & \vdots & \vdots \\ \cdots\cdots & 1 & \cdots\cdots \\ \vdots & \vdots & \vdots \\ \cdots\cdots & -1 & \cdots\cdots \\ \vdots & \vdots & \vdots \end{vmatrix}$$

接下来，我们考察 BB^T。容易证明，$(BB^T)_{ij}$ 等于 B 的第 i 行与第 j 列的点积。

所以，当 $i=j$ 时，$(BB^T)_{ij}$ 等于与 v_i 相连的边的个数，即 v_i 的度数；而当 $i\neq j$ 时，如果有一条连接 v_i、v_j 的边，则$(BB^T)_{ij}$ 等于-1，否则等于 0。这与 Kirchhoff 矩阵的定义完全相同！因此，我们得出：$C=BB^T$！也就是说，我们可以将 C 的问题转化到 BB^T 上来，这样做有什么用呢？

设 B_r 为 B 去掉第 r 行后得到的新矩阵，易知 $C_r=B_rB_r^T$。根据 *Binet-Cauchy* 公式，我们可以得到：

$$\det(C_r)=\det(B_rB_r^T)=\sum_{\substack{x\subset E\\|x|=n-1}}\det(B_r^x)\det(B_r^{xT})=\sum_{\substack{x\subset E\\|x|=n-1}}\det(B_r^xB_r^{xT})=\sum_{\substack{x\subset E\\|x|=n-1}}(\det(B_r^x))^2$$

B_r^x 是把 B_r 中属于 x 的列抽出后形成的新矩阵。我们注意到，当 x 中的边形成环时，总有 $\det(B_r^x)=0$。例如，如果新图中存在一个大小为 3 的环，那么我们可以重新安排 B_r^x 如下图所示：

$$\begin{bmatrix} \boxed{\begin{matrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & -1 \end{matrix}} & * \\ 0 & * \end{bmatrix}$$

这样，$\det(B_r^x)$等于左上角 3 阶子式的行列式乘以右下角 n-4 阶矩阵的行列式。而左上角的 3 阶子式是退化的，它每行每列的和都是 0。因此$\det(B_r^x)$也等于 0。类似的证明可以推广到环的大小任意的情况。

显然，$B_r^xB_r^{xT}$ 可以看成是仅由所有的顶点和属于 x 的边构成的新图的 Kirchhoff 矩阵的一个 n-1 阶主子式。根据图的 Kirchhoff 矩阵的性质，如果将所有属于 x 的 n-1 条边加入图中后形成一颗树，那么 $\det(B_r^xB_r^{xT})$为 1；而如果没有形成树，则必然存在一个环，那么 $\det(B_r^x)=0$。也就是说，我们考察**边集所有大小为 *n*-1 的子集，如果这个子集中的边能够形成一颗树，那么我们的答案加 1，否则不变**。这就恰好等于原图生成树的个数!因此，我们成功地证明了 *Matrix-Tree* 定理！

如果图中有重边，*Matrix-Tree* 定理同样适用，具体的证明方法类似，请读者

自己思考。因为计算行列式的时间复杂度为 $O(n^3)$，因此，生成树的计数也可以在 $O(n^3)$的时间内完成。

理解

刚才的分析可能有些“深奥”，下面让我们从直观上来理解一下这个定理。

我们首先来看一道简单的数学问题：试求方程

$$x_1 + x_2 + x_3 = 2$$

所有非负整数解的个数。

这是我们大家都很熟悉的一道组合计数问题。我们通常的做法是设有 2 个 *1* 和两个△，我们将这 4 个元素任意排列，那么不同的排列的个数就等于原方程解的个数，即 C_4^2。为什么要这样做呢？

我们将所有 6 种排列列出后发现，一种排列就对应了原方程的一个解：

△△11 对应 x_1=0，x_2=0，x_3=2；
△1△1 对应 x_1=0，x_2=1，x_3=1；
△11△对应 x_1=0，x_2=2，x_3=0；
……

也就是说，我们通过模型的转化，找出了原问题和新问题之间的对应关系，并利用有关的数学知识解决了转化后的新问题，也就同时解决了原问题。这种转化的重要意义在于：在不同问题之间的架起了互相联系的桥梁。

回到我们讨论的 *Matrix-Tree* 定理上来。我们同样是经过模型的转化后（将图模型转化为矩阵模型），发现 *Binet-Cauchy* 公式展开式中的每一项对应着边集一个大小为 *n*-1 的子集。其中，值为 1 的项对应一颗生成树，而没有对应生成树的项值为 0。这样，将问题转化为求展开式中所有项之和。再利用已有的数学知识，就可以成功解决这个问题。

我们将这两个问题进行对比，可以发现：在第一个问题中，方程的解所扮演的角色与图的生成树在第二问题中所扮演的角色类似；而第一个问题中排列方案所扮演的角色与第二个问题中展开式中的每一项所扮演的角色相同。同时，在每个问题中，两个对象之间又是互相对应的。这两个问题的不同之处仅仅在于：*相互之间的对应关系更加隐蔽、复杂，需要更加强大的数学理论来支撑。*

具体应用

下面我们通过几道例题来看一下生成树的计数问题。

［例二］员工组织（UVA p10766 Organising the Organisation）

Jimmy 在公司里负责人员的分级工作，他最近遇到了一点小麻烦。为了提高公司工作的效率，董事会决定对所有的员工重新分级！

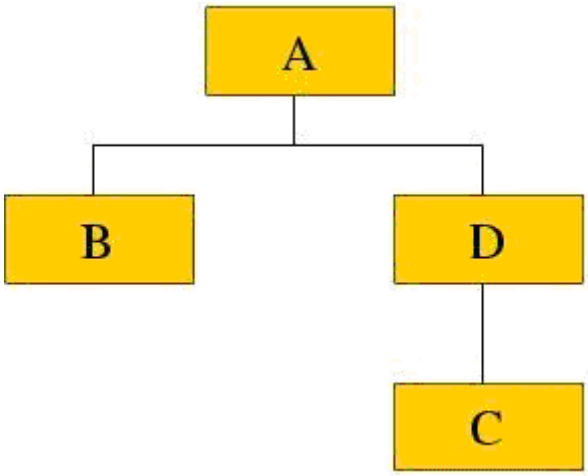

图 *a* 员工分级图

分级后的情况如图 *a* 所示，*A* 直接领导 *B* 和 *D*，*D* 直接领导 *C*。具体来说，除了一个总经理例外，其他所有的员工有且只有一个直接领导。由于员工直接的人际关系，可能出现 *a* 和 *b* 都不愿意让对方成为自己直接领导的情况。公司里的 *n* 位员工 1-*n* 编号，并且董事会已经决定让标号为 *k* 的员工担任总经理。Jimmy 的任务就是一共有多少种不同的员工分级方案。

数据规模：$1 \le n \le 50$。

［分析］

这道题目的解法比较明显。如果 *a* 和 *b* 直接没有矛盾，就在他们之间连一条边。显然，最后我们得到的员工之间的关系图就使原图的一颗生成树。虽然我们规定了生成树的根，但是因为无向图生成树的个数与根无关，所以我们只需要直接利用 *Matrix-Tree* 定理计算原图的生成树的个数即可。

［例三］国王的烦恼（原创）

Byteland 的国王 Byteotia 最近很是烦恼，他的国家遭遇到了洪水的袭击！百年未遇的洪水冲毁了 Byteland 许多重要的道路，使得整个国家处于瘫痪状态。Byteland 由 n 座城市组成，任何两个不同城市之间都有道路相连。为了尽快使国家恢复正常秩序，Byteotia 组织了专家进行研究，列举出了所有可以正常同行的道路（其他的道路可能还在洪水中☹）。其中有的已经被冲毁，需要重新修复;有的则可以继续使用。很奇怪的是，所有可以继续使用的道路并没有形成环。为了最大限度的节省时间，Byteotia 希望只修复最少的道路就可以使得整个国家连通。Byteotia 本来准备对每一种方案进行评估，选择最优的，不过很快他发现方案的个数实在是太多了。因此，他找到了你——Byteland 最优秀的计算机专家，帮他计算一下所有可能的修复方案的个数。

数据规模：$1 \le n \le 500$。

［分析］

这道题目乍看起来很难，因为要求统计修复道路个数最少的方案的个数，同时还有不需要修复的道路需要考虑。仔细分析后我们发现题目中一个十分重要的条件就是：所有可以继续使用的道路并没有形成环！这就为我们的解题创造了条件。我们都知道，让一个 n 个点的图连通最少需要 n-1 条边，而这些边形成一颗树，而树的一个重要性质就是无环，因此所有可以继续使用的道路都一定存在于最后得到的树中。这样，我们就成功地将问题转化为：求一个图生成树的个数，其中有某些边必选。

这也是一道生成树的计数问题。但是，由于必选边的存在，使得我们无法直接应用 *Matrix-Tree* 定理。应该怎么办呢？

我们知道，如果我们要求生成树中必须包含一条边 e，那么整个图 G 生成树的个数 $t(G)$就等于 $t(G-e)$，即将 e 收缩后得到的新图的生成树的个数。因此，我们需要：

1、将所有的必选边压缩；

2、求压缩后的新图的生成树的个数。

压缩一条边的时间复杂度为 $O(n)$，而最多只有 n-1 条边需要压缩，因此，这一步的复杂度为 $O(n^2)$。根据前面的分析，计算一个图生成树的个数的时间复杂度为 $O(n^3)$。因此，我们成功解决了整个问题，时间复杂度为 $O(n^3)$。

总结

本文介绍了生成树的计数算法及其包含的数学思想以及应用。在讨论的过程中，我们将图的生成树和矩阵的行列式这两样看起来毫无关系的事物紧密地联系在了一起。我们首先通过模型转化将图的生成树的计数转化为矩阵行列式的计算，再通过数学证明论证了矩阵的行列式与图的生成树之间的对应关系。从中，我们不难发现：***扎实的数学功底是解决问题的保证，创造性地联想是解决问题的灵魂。***

参考文献

[1]：线形代数 居余马 等编著
[2]：代数学引论 许以超 编著
[3]：Graph Theory II Reinhard Diestel
[4]：连通图中含某些指定边的生成树的计数 胡茂林
[5]：Counting Spanning Tree Martin Rubey