转置原理的简单介绍

陈宇 安徽师范大学附属中学

摘 要

本文介绍了转置原理（transposition principle），也被称为特勒根（Tellegen）原理，是一组对线性算法（linear algorithm）改写的规则。文中会介绍一些常用线性算法的转置，指出转置方法对一些问题可起到方便解决和优化效率的作用，并会介绍用转置原理优化的一类特殊的矩阵乘向量及其应用。

1 概述

我们假设以下所有对数的操作在一个数域 F 上进行。

定义 1. 输入一个 $n\times 1$ 向量 a，用一个 $n\times n$ 的常数矩阵 A 左乘该向量后，输出其结果 $b = Aa$（b 便为一个 $n \times 1$ 向量）的算法，被称为**线性算法**。我们认为输入的向量是变量，矩阵为常量。为了方便这里只考虑 A 是方阵，否则适当补零即可。

定义 2. 对于形如 $b = Aa$ 的线性算法，我们称形如 $b' = A^T a'$ 的线性算法是该**线性算法的转置**。

而转置原理断言，我们可以将一个线性算法改写成其转置后的算法，同时保持时空复杂度不变。这样我们就可以将算法进行转置后，优化该转置后的算法，再将该转置后的算法改写成解决原问题的算法。这在许多场合可以简化问题。

2 改写

这一节我们将指出，将线性算法改写为其转置，是一个较为机械的过程。

我们首先探讨矩阵转置后对算法过程的影响。

为了方便，我们将程序用到的与变量有关的额外空间，均算在输入向量中，这样我们适当改变输入输出即可得到相同效果。

定义 3. 我们在单位矩阵 I 上略作修改，得到以下两种矩阵：

1. 将 I 的第 i 行和第 j 行交换。该矩阵左乘一个向量的效果便为，将其第 i 位与第 j 位交换；
2. 将 I 的第 i 行第 i 个元素由 1 改为数 a。该矩阵左乘一个向量的效果便为，将其第 i 位乘上 a；
3. 将 I 的第 i 行第 $j(j \neq i)$ 个元素由 0 改为常数 a。该矩阵左乘一个向量的效果便为，将其第 j 位乘上 a 加到第 i 位上。

我们将这三种矩阵称为**初等矩阵**。

矩阵乘向量一般要使用 $O(n)$ 的额外空间。经过处理后，矩阵 A 就可以直接拆成一系列初等矩阵的乘积

$$A = E_1 E_2 \dots E_k$$

从而有：

$$A^T = E_k^T E_{k-1}^T \dots E_1^T$$

考虑初等矩阵的转置：第一种和第二种转置后不变，第三种是把第 i 位加到第 j 位改为第 j 位加到第 i 位。于是转置后的算法，就是将原算法的所有语句倒序执行，并将原来算法中形如 $x \leftarrow x + ay$ 的语句调整为 $y \leftarrow y + ax$。对于加常数操作，可在矩阵中补 1 实现。

实际应用时，我们可能要输入一个矩阵，然后根据该矩阵计算出算法使用的所有常量，得到一个线性算法。

对于函数参数。其中涉及到常量的部分原样不动，变量则是传入改为返回或修改，返回或修改改为传入。

函数调用只需要正常直接倒序，依次将里面的调用的其他函数转置执行即可；递归也可看做是对其他函数的调用。需要注意的是，由于我们要在反向执行的每个时刻，算出正向执行到这里的所有常量，这可能需要先预处理比较方便。

3 例子

3.1 离散傅里叶变换

考虑 DFT 的矩阵：

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix}$$

由于该矩阵是对称矩阵，该算法的转置即为其自身。IDFT 也类似。

3.2 快速傅里叶变换

广为使用的 FFT 是 DIT（decimation in time，按时域抽取）-FFT 形式的，即将多项式系数分为偶数项和奇数项，容易根据单位根性质递归为子问题，不再赘述；考虑改写成非递归，常用的方法是将数组下标的二进制位反转，这样每次取最低位变为最高位，偶数项和奇数项变成左半右半，就容易处理了。

另一方面，我们考虑直接分成前一半后一半：

$$\begin{aligned} DFT(a,n)_k &= \sum_{i=0}^{n-1} a_i \omega_n^{ik} \\ &= \sum_{i=0}^{n/2-1} \left(a_i^{(left)} + a_i^{(right)} \omega_n^{n/2k} \right) \omega_n^{ik} \end{aligned}$$

$\omega_n^{n/2k} = (-1)^k$，通过讨论 k 的奇偶性，可以得到一个将偶数项答案放在左边，奇数项答案放在右边的算法。这样改成非递归后，只需最后将数组下标的二进制位反转。

这个算法被称为 DIF（decimation in frequency，按频域抽取）-FFT。观察算法过程，两个功能相同的算法互为转置，也印证了 DFT 转置是其自身。

3.3 多项式乘法

我们用 $M(n)$ 表示求两个多项式循环卷积的前 $2n$ 位的时间，并假定 $M(n) = O(n \log n)$。

对最高次数分别为 $n-1$ 和 $m-1$ 的多项式 a 和 b，暴力多项式乘法得到次数为 $n+m-1$ 的多项式 $c = ab$ 的程序如下：

```
1  c ← 0
2  for i ← 0 to n + m − 2
3      for j ← max(0, i − n + 1) to min(i, m − 1)
4          c_i ← c_i + a_{i−j} b_j
```

我们将 b 视为常数，直接按上面规则转写，得到乘法的转置：给定次数为 $n-1$ 和 $m-1$ 的多项式 a 和 b，结果为 $n-m$ 次的多项式 $c = mul^T(a,b)$：

```
1  c ← 0
2  for i ← 0 to n + m − 2
3      for j ← min(i, m − 1) downto max(0, i − (n − m))
4          c_{i−j} ← c_{i−j} + a_i b_j
```

从而可看出，$mul^T(a,b)$ 是 a 与翻转系数后的 b 卷积结果的 $m-1$ 至 $n-1$ 位。

多项式乘法的时间自然是 $\frac{1}{2}M(n+m)$，转置乘法的时间则可做到 $\frac{1}{2}M(n)$，这是因为循环卷积溢出部分与我们需要的部分并不相交。同样的做法利用 [4.1] 中的结论也容易导出。

4 范德蒙德矩阵的转置

定义 4. 范德蒙德矩阵是一个 $n \times n$ 方阵，形如：

$$V_\alpha = \begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^{n-1} \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{n-1} & \alpha_{n-1}^2 & \cdots & \alpha_{n-1}^{n-1} \end{bmatrix}$$

定义 5. 对于最高次数为 $n-1$ 的多项式 b（其系数的向量也写成 b），

$$V_\alpha b^T = (b(\alpha_0), b(\alpha_1), \ldots, b(\alpha_{n-1}))^T$$

这个计算一般被称为**多点求值**。

下面我们将说明，转置原理的应用可以大幅度减小多点求值的时间常数。先介绍几个与该问题有关的问题。

4.1 多项式整除和取模

给定 $n-1$ 次多项式 f 和 $m-1 (m \le n)$ 次多项式 g，求 h, r 使得 $f = gh + r$ 使得 $|r| < n$.（假定 m 大约是 $0.5n$）

对于该问题，Sieveking-Kung's algorithm 给出了计算公式：我们令 $rev(a,n)$ 表示 $n-1$ 次多项式 a 翻转系数的结果，有：

$$rev(n-m+1, h) = rev(n, f)\frac{1}{rev(m, g)} \pmod{x^{n-m+1}}$$

形式幂级数求商可以使用 $\frac{7}{3}M(m) = \frac{7}{6}M(n)$ 的时间完成[1]。之后算出 $r = f - gh$ 还需要使用 $M(m) = \frac{1}{2}M(n)$ 的时间，总时间 $\frac{5}{3}M(n)$。

4.2 多个单项式求乘积

给定 $a_1, a_2, \ldots, a_n$，输出多项式 $(1 + a_1x)(1 + a_2x)\ldots(1 + a_nx)$ 的系数。

我们可以直接使用分治解决。具体我们令

$$F(l, r) = \prod_{i=l}^{r}(1 + a_ix)$$

用

$$F(l, r) = F\left(l, \left\lfloor \frac{l + r}{2} \right\rfloor\right) \times F\left(\left\lfloor \frac{l + r}{2} \right\rfloor + 1, r\right)$$

计算。

这样的时间为

$$T(n) = 2T(n/2) + M(n/2) = \frac{1}{2}M(n)\log_2 n + O(M(n))$$

4.3 幂和

给定 $a_1, a_2, \ldots, a_n$，令 $f(k) = \sum_{i=1}^{n} a_i^k$，求出 $f(0), f(1), \ldots, f(n - 1)$。

考虑形式幂级数

$$F(z) = \sum_{k\geq 0} f(k)z^k = \sum_{k\geq 0}\sum_{i=1}^{n} a_i^k z^k = \sum_{i=1}^{n}\frac{1}{1 - a_iz}$$

即求 $F(z) \mod z^n$。

易知

$$\prod_{i=1}^{n}(1 - a_iz)F(z) = \sum_{i=1}^{n}\prod_{j\neq i}(1 - a_jz)$$

令其为 $A(z)F(z) = B(z)$，只需分别求出 $A, B \mod z^n$ 后用一次形式幂级数求商即可。

我们令 $A(l, r), B(l, r)$ 定义与 [5.1] 中的 $F(l, r)$ 类似，那么 A 类似求出，而

$$B(l, r) = A\left(l, \left\lfloor \frac{l + r}{2} \right\rfloor\right)B\left(\left\lfloor \frac{l + r}{2} \right\rfloor + 1, r\right) + B\left(l, \left\lfloor \frac{l + r}{2} \right\rfloor\right)A\left(\left\lfloor \frac{l + r}{2} \right\rfloor + 1, r\right)$$

这样分治计算，容易看出是 $\frac{3}{2}M(n)\log_2 n + O(M(n))$ 的时间。

4.4 多点求值与插值

多点求值传统上是使用一个分治算法。考虑 b 求 α 中的点值，当 $|b| \geq |\alpha|$ 时，可以将 b 对 $(x - \alpha_0)(x - \alpha_1)\ldots(x - \alpha_{|\alpha|-1})$ 取模，从而令 $|b|$ 总是小于 $|\alpha|$。这样，我们就可以每次将点值分为两半递归下去。

这样的时间为 [5.3] 的时间加上每次分治调用 [5.2] 的时间，经过计算得为 $\frac{11}{3}M(n)\log_2 n+O(M(n))$。对于流行的牛顿法求逆未优化的实现，这个速度还会慢一倍以上。

下面，我们考虑范德蒙德矩阵的转置：

$$V_\alpha^T=\begin{bmatrix}1 & 1 & \cdots & 1\\ \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1}\\ \alpha_0^2 & \alpha_1^2 & \cdots & \alpha_{n-1}^2\\ \vdots & \vdots & \ddots & \vdots\\ \alpha_0^{n-1} & \alpha_1^{n-1} & \cdots & \alpha_{n-1}^{n-1}\end{bmatrix}$$

考虑转置后的问题 $V_\alpha^T b=c$，那么 $c_i=\sum_{j=0}^{n-1}\alpha_j^i b_j$，这与 [5.3] 中的问题基本类似，进行微小的修改后即得一个 $\frac{3}{2}M(n)\log_2 n+O(M(n))$ 时间的分治算法。

将该算法改写为转置前版本，会将原来一层分治的一次乘法变为两次，这会导致时间翻倍。改写后的算法时间为 $\frac{5}{2}M(n)\log_2 n+O(M(n))$。

插值的算法主要调用一次多点求值，与多点求值的时间也相同。

4.5 实践

我们对多点求值的实现提交在 https://www.luogu.com.cn/record/29053749，该题中 n 的量级为 2^{16}。我们的程序于目前（2020-1-08 之前）最快，并比未经底层优化的最快程序快一倍以上，符合以上理论计算。

之前多点求值在比赛中并不常见，并不一定是因为用处不多，而很大一部分原因是多点求值需要套用多种算法，记忆和编写较繁琐；同时算法本身常数很大，出题人不一定能在合理的时限内卡掉暴力，选手辛辛苦苦写完后又可能因为常数原因无法通过。而上述改进算法只需要使用分治和多项式乘法，常数也减小很多，一定程度上扩展了多点求值的实用性。

5 一类特殊矩阵乘向量的快速算法

5.1 描述

我们考虑一个线性变换 $A\alpha=\beta$。现在，考虑 $n\times n$ 矩阵 A 的系数的二元生成函数：$A(x,y)=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}A_{i,j}x^i y^j$。如果 $A(x,y)$ 可以表示成 $u(x)v(y)f(g(x)h(y))$，其中 g 和 h 由一系列（常数个）我们将给出的简单函数复合而成，那么我们可以在 $\tilde{O}(n)$ 时间内完成该矩阵和其逆矩阵左乘向量；具体地，若 g,h 中含有 $\exp,\log$，则复杂度为 $O(M(n)\log n)$，否则为 $O(M(n))$（该复杂度优于之前所有已知方法）。为保证下面算法的运算有意义，还要求 gh 的

常数项为 0，g 和 h 的一次项都非 0；u 和 v 常数项非 0，f 任意项非 0（如完全不需要逆可以省去）。

5.2 复合

定义 6. *考虑 $n-1$ 次多项式 F 和一个线性变换：$F(G(x)) \mod x^n$（$G(x)$ 是常数），称其为* ***右复合***

这个计算并没有好的方法，但如果 $G = g_1 \circ g_2 \circ \ldots \circ g_k$，其中 g_i 是一些简单函数，那么我们可以运用复合的结合律：$A(B(C(x))) = A \circ B(C(x))$，之后将 F 进行 k 次对简单函数的复合就可算出。少部分函数由于不封闭会用到 $C(x)$ 的信息，需特别注意。如果符合上面提到的条件，除此之外的计算可以每一步都截断到 n 位而不丢失精度。

我们认为简单函数是以下几类：

1. 加法 $x + k$

   有

   $$F(x+k) = \sum_{i=0}^{n-1} F_i (x+k)^i = \sum_{i=0}^{n-1} F_i \sum_{j=0}^{i} \binom{i}{j} x^j k^{i-j} = \sum_{j=0}^{n-1} \frac{x^j}{j!} \sum_{i=j}^{n-1} i! \frac{k^{i-j}}{(i-j)!}$$

   显然可用 $M(n)$ 时间完成。

2. 乘法 kx，只需将 k 的若干次幂乘入系数，时间 $O(n)$。

3. 幂 $x^k (k \in \mathbb{Z}^+)$，将下标变换即可，没有进行任何算术运算。

4. 逆 x^{-1}，注意幂级数不能有负指数，考虑 $F(x^{-1}) = (rev(n, F))(x) x^{1-n}$，其中 x 是另一个被复合的幂级数，翻转系数没有进行算术运算，计算后面幂级数逆的系数需要 $O(M(n))$ 时间。

5. 根 $x^{1/k} (k \in \mathbb{Z}^+)$，这里需要保证 x 的 k 次方根唯一，推之前的 $g(x)h(y)$ 形式时就应当通过验证一些项去舍掉不合法的解。

   设 x 对应后面的幂级数是 g，且 $g = h^k$（这里要保证 h 的唯一性）。我们将 F 的下标做带余除法 F_{ik+j}，然后按模 k 分类：

   $$F_i(x) = \sum_j F_{jk+i} h^j$$

   那么有

$$F(h) = \sum_i F_i(g)h^i$$

递归为 $\lfloor n/k \rfloor$ 个规模为 k 的子问题后拼起来即可。求出 h 需要 $O(M(n))$ 时间[1]，之后需用 $O(kM(n))$ 时间求出 h 的 $0 \dots k-1$ 次幂，因此需保证 k 是常数。

6. 指数 $\exp(x) - 1$，减一是为了和对数互为逆，不妨不考虑减一，那么有：

$$F(e^x) = \sum_{i=0}^{n-1} F_i e^{ix} = \sum_{i=0}^{n-1} F_i \sum_{j\geq 0} \frac{i^j}{j!} x^j = \sum_{j\geq 0} \frac{x^j}{j!} \sum_{i=0}^{n-1} F_i i^j$$

我们考虑求出

$$\sum_{j\geq 0} x^j \sum_{i=0}^{n-1} F_i i^j$$

当然要截断到 n 位；之后我们将每个位置除以 $j!$ 即可。

上述问题与之前提过的多点求值的转置问题是相同的，用分治和多项式乘法即可解决。复杂度 $O(M(n)\log n)$。

7. 对数 $\log(x+1)$，直接做并不好分析，我们不妨考虑复合指数算法的逆。

复合指数算法可概述为，将 F 运行多点求值的转置问题 $0, 1, \dots, n-1$，然后将每个位置乘 $\frac{1}{j!}$。

将该算法转置可得，先将每个位置乘 $\frac{1}{j!}$，再对 $0, 1, \dots, n-1$ 运行多点求值。其逆显然为，运行多点插值后将每个位置乘 $j!$。再转置回去复杂度不变，故为 $O(M(n)\log n)$。

更好的一点是，上面的函数都存在复合逆。

5.3 算法

我们回到开始的问题。先假设 $u = v = 1$，有：

$$F_{i,j} = \sum_k g_i^k f_k h_j^k$$

那么，该线性变换是三个线性变换的复合：g 右复合，点乘 f，h 右复合的转置，按上一节方法顺次计算即可。

加上 u, v 后，原线性变换变为三个线性变换的复合：乘 u，原线性变换，乘 v 的转置，同样可以轻松计算。

在限制条件下，上面这些操作都存在逆且可以直接取逆，故逆矩阵也可以相应计算。

综上，我们以很低的复杂度解决了该类问题。

6 应用

6.1 基变换

定义 7. *考虑一系列多项式 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$，若任何 n 次多项式都可以被它们线性表出，则这是模 x^n 多项式空间的一组基。OI 中常见的基有单项式基和下降幂基。*

多项式可以在不同的基下被表示。在线性代数中我们知道，从一个基的表示转换到另一个基，对应着一个线性变换。多项式基的变换往往对应 [6] 中提到的问题。我们有以下单项式基到其他基互相变换的结果：

- $O(M(n))$ 时间可计算：拉盖尔（Laguerre）多项式、厄米（Hermite）多项式、雅可比（Jacobi）多项式、斐波那契（Fibonacci）多项式、欧拉（Euler）多项式、伯努利（Bernouli）多项式、Mott 多项式、Spread 多项式、贝塞尔（Bessel）多项式
- $O(M(n)\log n)$ 时间可计算：下降幂、贝尔（Bell）多项式、第二类伯努利多项式、Poisson-Charlier 多项式、Actuarial 多项式、Narumi 多项式、彼得斯（Peters）多项式、Meixner-Pollaczek 多项式、Meixner 多项式、Krawtchouk 多项式

上面的许多基在 OI 中的优美应用尚待挖掘。

6.2 例题

6.2.1 真实无妄她们的人生之路

题目来源 Comet OJ #2 F

题目大意 有 $n(n \le 10^5)$ 件物品，第 $i(1 \le i \le n)$ 件物品有属性 $p_i(0 < p_i \le 1)$；

主人公等级初始为 0，使用第 i 件物品会有 p_i 的概率让等级加一，$1 - p_i$ 概率不变；

若最后等级为 j，则会产生 a_j 的攻击力。令 f_i 表示使用除 i 外的 $n-1$ 个物品后，主人公产生的期望攻击力，求出 $f_1, f_2, \ldots, f_n$。在 $F_{998244353}$ 下计算。

解法 设 $C_i(x) = 1 - p_i + p_i x, C(x) = \prod_i C_i(x)$，则 C/C_i 与 a 点积结果的系数之和便为 f_i。

将 a 看做输入向量，这是一个线性变换问题 $Aa = f$，其中 $A_{i,j} = C/C_i[x^j]$。考虑转置问题 $A^T a = g$，则有 $g = \sum_i a_i C/C_i$。这可以用类似幂和问题的分治与多项式乘法求出。

6.2.2 随机游走

题目大意 给定 $n,a,b(n\le 10^5)$，在一个无穷大的二维平面上，一个人出发点是 $(0,0)$，每步可以沿向量 $\overrightarrow{(1,1)}$，$\overrightarrow{(1,0)}$ 和 $\overrightarrow{(2,0)}$ 走，分别带有 $1,a,b$ 的权值；

此人随时可以停止。若停在 (x,y)，则会产生 w_y 的权值；

定义一条路径的权是这些权值的乘积。对于 $1\le a\le n$，求出结束在 x 坐标为 a 的所有点的所有路径权值之和 g_a。在 $F_{998244353}$ 下计算。

解法 不考虑 w_y，令 $f_{x,y}$ 表示到达 x,y 的路径权值和，$f_{0,0}=1$，此外 $f_{i,j}=f_{i-1,j-1}+af_{i-1,j}+bf_{i-2,j}(i,j\ge 0)$，此外 $f_{i,j}=0$，那么 $g_a=\sum_y f_{a,y}w_y$，这是一个线性变换问题。

我们令 f 的二元生成函数为 $F(x,y)=\sum_{i,j} f_{i,j}x^iy^j$，那么方程 $F=1+x(a+y)F+x^2F$ 成立，解得：

$$F=\frac{1}{1-(a+y)x-x^2}=\frac{1}{(1-x^2)\left(1-\frac{a+y}{1-x^2}\right)}$$

这可以转为 [6] 中的形式，其中

$$u(x)=\frac{1}{1-x^2},v(y)=1,f(x)=\frac{1}{1-x},g(x)=\frac{1}{1-x^2},h(y)=a+y$$

套用算法即得 $O(M(n))$；实际上这就是到一类正交多项式的基变换问题。

6.2.3 二叉树计数

题目大意 给定 $n(n\le 10^5)$，定义一个有 x 个叶子的二叉树权值为 w_x，设 $f(t)$ 表示所有 t 个点的无标号二叉树的权值和，请求出 $f(1),f(2),\dots,f(n)$。在 $F_{998244353}$ 下计算。

解法 与上面类似的，令 $f_{i,j}$ 为 i 个点 j 个叶子的二叉树个数，F 为其二元生成函数，那么可列出 $F=x(y+2F+F^2)$，解得：

$$F=\frac{1-2x-\sqrt{(2x-1)^2-4x^2y}}{2x}=\frac{1}{2x}(1-2x)\left(1-\sqrt{1-\left(\frac{1}{2x-1}+1\right)^2y}\right)$$

我们令

$$u(x)=1-2x,v(y)=1,f(x)=1-\sqrt{1-x},g(x)=\left(\frac{1}{2x-1}+1\right)^2,h(y)=y$$

套用算法即可算出里层，外面乘上 $\frac{1}{2x}$ 简单处理即可，复杂度 $O(M(n))$。

7 总结

本文探究了转置原理及其应用。转置原理给出了一种处理线性变换问题的崭新思路，从算法变化的角度考察了矩阵的变化。利用转置原理大幅优化了多点求值的常数和编程难度，揭示了许多看似不同经典问题蕴含的相同本质。在最后文章解决了一类范围很广的矩阵乘向量问题，让许多原来困难的计数问题被轻松解决，这些方面都可以看到转置原理广泛的应用、强大的威力和无穷的潜力。除本文提到的外，其他转置原理的巧妙运用还待读者发掘。希望本文可以对读者起到启发思想、拓宽出题与解题思路的作用。

8 致谢

感谢中国计算机学会提供交流和学习的平台。
感谢国家集训队高闻远教练的指导。
感谢叶国平老师在学习生活方面的帮助。
感谢朱震霆、周雨扬等学长同学的指导。
感谢罗恺、李白天、张好风等同学为本文审稿。

参考文献

[1] 关于优化形式幂级数计算的 Newton 法的常数（http://negiizhao.blog.uoj.ac/blog/4671）。

[2] Tellegen's Principle into Practice, A. Bostan, G. Lecerf, ?. Schost, 2005。

[3] Power Series Composition and Change of Basis, Alin Bostan, Bruno Salvy, Eric Schost, 2008。