转置原理及其应用

rushcheyo

Augest 3, 2020

提要

1. 介绍
2. 例子
3. 多点求值新科技
4. 基变换
5. 一些例题
6. 结束

介绍

模型

线性算法接受一个长为 n 的向量 $\mathbf{x}$，输出一个长度为 m 的向量 $\mathbf{y}$。

该算法想计算的是一个关于 $m \times n$ 矩阵 A 的线性变换：$\mathbf{y} = A\mathbf{x}$。

算法使用的内存位除了 n 个输入位外，还有 k 个额外位 $r_1, r_2, \ldots, r_k$ 存储中间变量。

指令集

约定线性算法能使用的（抽象）指令只有下面 4 种：

1. `swap i j`：将位 i 与位 j 交换；
2. `mul i c`：将位 i 乘上常数 c；
3. `add i j c`：将位 i 乘上常数 c 加到位 $j(j \neq i)$ 上；
4. `output k`：输出位 k（第 t 条 output 指令会输出到 $\mathbf{y}_t$，m 等于 output 指令的数量），为了方便起见该指令必须出现在程序的末尾。

（注：我们在考虑算法耗时时只考虑「算术运算」，即加法和乘法）

意义

可以看到，线性算法的例子在 OI 中相当广泛：离散傅里叶变换（DFT）、多点求值与插值、二项式反演等等都是线性算法。

我们处理的算法问题中很大一部分也都是与线性变换相关的问题。

转置原理给我们提供了看待问题新的角度。

问题

转置原理指出，我们可以将计算 A 的线性算法转换成计算 A^T 的线性算法，乘法次数不变，加法次数比原算法多至多 $m-n$ 次，也就是说我们解决了计算某个线性变换的问题就在几乎相同时间空间下解决了其转置问题。

构造

新算法接受一个长为 m 的向量 $\mathbf{y}'$，输出一个长度为 n 的向量，使用 $n+k$ 个额外位 $r'_1, r'_2, \ldots, r'_k, \mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_n$。

新算法的第 i 条指令是原算法倒数第 i 条指令的「转置指令」。我们给出四种指令的转置指令如下：

1. `swap i j`：不变；
2. `mul i c`：不变；
3. `add i j c`：改为 `add j i c`；
4. `output k`：第 t 条改为 `add y'[t] k 1`；

注意：如果原来位指代 $\mathbf{x}_p$，那么新算法中就是 $\mathbf{x}'_p$。$\mathbf{y}, r$ 同理。

之后直接添加 n 条 `output` 指令，依次输出 $\mathbf{x}'$ 即可。

然后为了保证算法的转置还是其本身，我们需要删掉无用变量，并把只出现一次的变量删去（相当于更名）。

证明

我们先证明计算次数。

假如原来位 i 有 a 次在 `mul` 中被乘，b 次在 add 中被乘，c 次在 add 中被加，d 次被输出，那么在原算法中贡献了 $a+b$ 次乘法，c 次加法，转置后算法中贡献了 $a+c$ 次乘法，$b+d-1$ 次加法（第一次被加时直接赋值，假设不存在无用变量）。

注意到 $\sum b = \sum c$，于是对所有变量求和后乘法次数不变，加法次数的增量为 $\sum d - 1 = m - n$ 条。

证明

再证明该算法的确计算了 A^T。我们考虑拆成 $A = CB$，B 是 $\mathbf{x}$ 到 $n+k$ 个可用位的贡献矩阵，C 是这些位对 $\mathbf{y}$ 的贡献矩阵，那么 $A^T = C^T B^T$，于是只需两部分分开证；而这一部分证明也是显然的，用 $(UV)^T = V^T U^T$ 归纳即可。

补充

解决实际问题时，我们实际上是将输入的一部分用于生成一个线性算法（中的变换系数），再用该算法去运行剩余的输入，因此转置时要分清代码里哪些部分是生成算法，哪些部分是执行线性算法，前部分不能转置。

对于程序里的循环结构，转置会将其倒序执行；对于（有副作用的）函数要考虑好它的输入和输出是哪些，按上面的规则改写即可。

例子

离散傅里叶变换

DFT 的矩阵是对称矩阵，$O(n^2)$ 算法的转置即为其自身。IDFT 也类似。

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。

快速傅里叶变换

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。

快速傅里叶变换

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。

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
```

多项式乘法

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 $c = (\times b)^T a$:

```
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，$(\times b)^T a$ 是 a 与翻转系数后的 b 卷积结果的 $m-1$ 至 $n-1$ 位，实际上也就是下标运算为减法的卷积。

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 DFT 的转置也容易导出。

多点求值新科技

范德蒙德矩阵

范德蒙德矩阵是一个 $n \times n$ 方阵，形如：

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

可以看到多点求值就是计算 $V_\alpha b^T$ 的线性算法。

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：$c_i = \sum_{j=0}^{n-1} \alpha_j^i b_j$，只需计算
$C(z) = \sum_{j=0}^{n-1} \frac{b_j}{1-\alpha_j z}$ 的前 n 项系数，这是一个简单的分治 FFT。

梳理

我们考虑转置后问题的程序。为了分清生成算法和执行算法，我们使用两个 pass。

第一个 pass 为，求出 $[1, n]$ 的线段树上每个区间 $1 - \alpha_j z$ 的乘积 T；

第二个 pass 同样先求出左右子树的答案，然后用 $F \leftarrow F_l T_r + F_r T_l$ 上传；更清晰的，写成两条语句 $F \leftarrow F_l T_r$，$F \leftarrow F_r T_l$（这里箭头表示 +=）；

最后我们计算根节点的 F/T。

梳理

那么转置回去后，第一个 pass 不变，然后我们将输入的多项式系数减法卷积上根节点的 $1/T$ 得到根节点的 F'。

第二个 pass 现在就变成了，用父亲更新儿子再递归下去。具体的语句是 $F_l \leftarrow (\times T_r)^T F$，$F_r \leftarrow (\times T_l)^T F$。

传统解释？

实际上也可以理解成，在原来多项式取模的多点求值中，我们只需要知道每层 T 翻转系数的逆。

而翻转系数的乘法其实就是乘法的转置，具有结合律。我们可以求出所有单项式的乘积的逆，再每次乘法转置上右边，就能得到左边翻转系数的逆。

另一种理解是，$[x^0](\times b)^T a$ 实际上给出了 a 和 b 的内积，于是我们只要求出一堆乘法转置的常数项。只需要常数项就对分治区间的长度产生了限制。

这种理解的用处我们之后可以看到。

1s 1e6?

可以看到新多点求值在好写程度和速度上都碾压原来使用多项式取模的多点求值。

但 1s 1e6 的话……

基变换

引入

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）。

右复合

考虑 $n-1$ 次多项式 F 和一个对 F 的线性变换：$F(G(x)) \bmod x^n$（$G(x)$ 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。

下面介绍一些简单函数的复合方法。

加

$f(x+k)$，显然是一次卷积。

乘

$f(xk)$，线性乘一遍就好。

幂

$f(x^k)$ 下标变换而已。

逆元

$f(x^{-1})$，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。

根

$f(x^{1/k})(k \in \mathbb{Z}^+)$，（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，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。

指数

$f(\exp x - 1)$，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 $O(M(n)\log n)$。

对数

$f(\log(x+1))$，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 $0, 1, \ldots, n-1$，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 $0, 1, \ldots, n-1$ 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 $O(M(n)\log n)$。

回归正题

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。

一些例题

真实无妄她们的人生之路

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

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 $f_1, f_2, \ldots, f_n$。

Source：Comet OJ #2 F

真实无妄她们的人生之路

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 $g = \sum_i a_i C/C_i$。这是一个简单的分治 FFT。

随机游走

给定 $n, a, b(n \leq 10^5)$，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 $1 \leq a \leq n$，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。

随机游走

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 $f_{i,j} = f_{i-1,j-1} + af_{i-1,j} + bf_{i-2,j} (i, j \geq 0)$，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 $F(x, y) = \sum_{i,j} f_{i,j} x^i y^j$，那么方程 $F = 1 + x(a + y)F + x^2 F$ 成立，解得：

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

这可以转为上面的形式，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 $O(M(n))$。

给 $n, b, a[\,]$，对 $0 \le m \le n$ 计算：

$$f_m(n) = \sum_{i=0}^{n} a_i \sum_{j=0}^{i} b^j \binom{m}{j} \binom{n-m}{i-j}$$

$n \le 10^6$，其中 25% 的 Subtask 满足 $n \le 5000$。

首先由~~转置原理~~倒腾一下阶乘，发现只需计算：

$$g_m(n) = \sum_{i=0}^{n} \sum_{j=0}^{i} a'_i b^j \binom{i}{j} \binom{n-i}{m-j}$$

那么可以在 (i, j) 格点上放权值 $a'_i b^j$，之后只要算 $(0, 0)$ 到 (n, m) 所有格路权值和，简单 DP 即可。用序列的组合意义可以推出一样的做法。

前面提到，$[z^0](a\times)^T b$ 给出了 a 和 b 的点积 $a \cdot b$。

于是，

$$\begin{aligned} f_m(n) &= [z^0]\left((1+z)^{n-m}(1+bz)^m\times\right)^T a \\ &= [z^0]\left(\left(\frac{(b-1)z}{1+z}+1\right)^m\times\right)^T ((1+z)^n\times)^T a \end{aligned}$$

令 $a'' = ((1+z)^n\times)^T a, t = \frac{(b-1)z}{1+z}$，$g_m = t^m \cdot a''$，
$f_m = (t+1)^m \cdot a'' = \sum_{i=0}^m \binom{m}{i} t^i \cdot a'' = \sum_{i=0}^m \binom{m}{i} g_i$，算出 g 后一次卷积就能得到 f。不难发现算 g 也只要一次卷积。

结束

作者

营员交流的内容由安徽师范大学附属中学陈宇、赵雨扬、曾致远（按姓氏笔画排列）合作完成。

参考文献

1. Tellegen's Principle into Practice, A. Bostan, G. Lecerf, É. Schost, 2005.
2. Power Series Composition and Change of Basis, Alin Bostan, Bruno Salvy, Eric Schost, 2008.

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、任清宇等学长的指导。

感谢罗恺、李白天、张好风等同学验稿。

谢谢大家！