多项式除法及其应用

余行江 & 彭雨翔

长郡中学

2014 年 2 月 9 日

自我介绍

- 大家好我是来自长郡中学的余行江，网名 sumix173，平时是个潜水党。
- 旁边那位是蒟蒻彭雨翔，网名 Picks。大家可以在幻灯片后半部分感受一下他那鬼畜的理性愉悦。

预备知识

- FFT
  - FFT 是什么高端算法？我怎么没听说过？
  - 这个... 最简单地说，FFT 可以优化多项式乘法。大家一般计算两个多项式的乘积都要做 $O(n^2)$ 的运算，但 FFT 可以做到 $O(n\log n)$ 。
  - 其实也不是什么高端算法，
    - “ FFT 这种东西每天早读的时候读两遍就好了 “ ——Delayyy
- NTT
  - 数论变换下的 FFT 又是什么？
  - 这个可以参考核武的百度空间：
  - AekdyCoin：《一切皆整数的 FFT》
  - 简单地说就是在整数域内处理 FFT ，不必因实数而担心浮点误差了。

问题

- 给出多项式 $A(x), B(x)$ ，设 $\deg A(x) = n, \deg B(x) = m$ ，$\deg P(x)$ 指 $P(x)$ 的次数。
- 求多项式 $C(x), D(x)$ ，满足：

$$A(x) = B(x) \times C(x) + D(x)$$

$$\deg D(x) \le m - 1$$

- 其中所有多项式的系数均在模素数 $1998585857 = 2^{21} \times 953 + 1$ 下。
- 范围
  - $n \le 1000$?
  - $n \le 10000$?
  - $n \le 100000$?

朴素算法及一个较优算法

- 朴素的 $O(n^2)$ 不用多说了，大部分人应该都手算过多项式除法。
- 设 $C(x)$ 的系数项分别为 $C_0, C_1, C_2, ..., C_{n-m}$ ，不难得出 C_i 与 $C_{i+1}, C_{i+2}, ..., C_{n-m}$ 成卷积关系。利用分治 $+$ FFT 可以得到一个 $O(n\log^2 n)$ 的算法。由于这个算法不好继续优化，不详述。

一个优秀的算法

- 令 $M^R(x)$ 为将 $M(x)$ 的系数反转的多项式，若 $\deg M(x) = n$ ，用数学语言描述就是

$$M^R(x) = x^n M(\frac{1}{x})$$

- 可得：

$$x^n A(\frac{1}{x}) = x^m B(\frac{1}{x}) x^{n-m} C(\frac{1}{x}) + x^n D(\frac{1}{x})$$

- 即：

$$\begin{aligned}
A^R(x) &= B^R(x)C^R(x) + x^{n-m+1}D^R(x) \\
A^R(x) &\equiv B^R(x)C^R(x) && \pmod{x^{n-m+1}} \\
C^R(x) &\equiv A^R(x)[B^R(x)]^{-1} && \pmod{x^{n-m+1}} \\
B^R(x)^{-1} &\equiv \frac{1}{B^R(x)} && \pmod{x^{n-m+1}}
\end{aligned}$$

一个优秀的算法

- 由于 $C^R(x)$ 的次数为 $n-m$ ，所以在模意义下的 $C^R(x)$ 即可直接得到 $C(x)$ 。
- 即我们只需要求 $B^R(x)$ 在模 x^{n-m+1} 下的逆元。
- 利用分治和 FFT ，我们可以在 $O(n\log^2 n)$ 内解决该问题，但这并不是我们本次讨论的重点，故略过。
- 在此我们介绍一种 $O(n\log n)$ 的算法。
- 令 $n-m+1$ 为 T , 我们要求的逆元为 $P_T(x)$ ，其中下标 T 指该逆元是在模 x^T 意义下的。
- 当 $T=1$ ，则直接对常数项取逆元即可。
- 现在假设 $P_{\lceil \frac{T}{2} \rceil}(x)$ 已经被求出。我们的目的是通过它求出 $P_T(x)$ 。

一个优秀的算法

- 根据已知，我们有：

$$P_{\lceil \frac{T}{2} \rceil}(x) \times B^R(x) \equiv 1 \pmod{x^{\lceil \frac{T}{2} \rceil}}$$

$$P_T(x) \times B^R(x) \equiv 1 \pmod{x^T}$$

- 得到：

$$\begin{aligned} P_{\lceil \frac{T}{2} \rceil}(x) - P_T(x) &\equiv 0 \pmod{x^{\lceil \frac{T}{2} \rceil}} \\ (P_T(x) - P_{\lceil \frac{T}{2} \rceil}(x))^2 &\equiv 0 \pmod{x^T} \\ P_T^2(x) - 2 \times P_T(x) \times P_{\lceil \frac{T}{2} \rceil}(x) + P_{\lceil \frac{T}{2} \rceil}(x)^2 &\equiv 0 \pmod{x^T} \end{aligned}$$

- 两边同乘 $B^R(x)$ ，利用逆元消去一些东西，变成：

$$P_T(x) - 2 \times P_{\lceil \frac{T}{2} \rceil}(x) + B^R(x) \times P_{\lceil \frac{T}{2} \rceil}(x)^2 \equiv 0 \pmod{x^T}$$

$$P_T(x) \equiv 2 \times P_{\lceil \frac{T}{2} \rceil}(x) - B^R(x) \times P_{\lceil \frac{T}{2} \rceil}(x)^2 \pmod{x^T}$$

- 我们便能利用 FFT 来运算后面的多项式乘法了。利用这种类似倍增的思想，我们只需要做 $O(\log n)$ 次此操作。

一个优秀的算法

- 伪代码：

$$
\begin{aligned}
&InverseElement(A(x), T):\\
&\quad if\ T == 1:\\
&\quad\quad return\ A[0]^{-1}\\
&\quad P_{\lceil\frac{T}{2}\rceil}(x) = InverseElement(A(x), \lceil\frac{T}{2}\rceil)\\
&\quad return\ 2 * P_{\lceil\frac{T}{2}\rceil}(x) - A(x) * P_{\lceil\frac{T}{2}\rceil}(x) * P_{\lceil\frac{T}{2}\rceil}(x)\%x^T
\end{aligned}
$$

- 粗略来看总复杂度是 $O(n\log^2 n)$ 的，但是由于每次 FFT 的长度减半，仔细算算：

$$\sum_{i=1}^{\lceil\log n\rceil} 2^i \times \log(2^i) = \log 2 \sum_{i}^{\lceil\log n\rceil} i \times 2^i$$

- 这就是经典的数列求和了，可以发现它是 $O(n\log n)$ 的。

一个优秀的算法

- 那么整个多项式除法的伪代码为：

$$A^R(x) = Reverse(A(x)), B^R(x) = Reverse(B(x))$$
$$[B^R(x)]^{-1} = InverseElement(B^R(x), n-m+1)$$
$$C^R(x) = A^R(x) * [B^R(x)]^{-1} \% x^{n-m+1}$$
$$C(x) = Reverse(C^R(x))$$
$$D(x) = A(x) - B(x) * C(x)$$

求出 $[B^R(x)]^{-1}$ 后，只需要带回原式，求出 $C(x)$ ，然后再次进行 FFT 来求得 $D(x)$ 便可得到多项式除法的余数。

一个优秀的算法

- 最后，阻拦我们的只是 $B(x)$ 的常数项为 0 的情况了。因为在这种情况下，$B(x)$ 是没有逆元的。
- 注意到我们所求的是 $\dfrac{A(x)}{B(x)}$，设 $B(x)$ 的 x^0 项至 x^k 项系数为 0。
- 那么 $A(x)$ 的 x^0 至 x^k 项不会被消去，必然成余式的一部分。
- 所需要做的只是将 $A(x), B(x)$ 去掉 x^0 至 x^k 项求出商式与余式，将余式乘上消去的因子后加上最开始去掉的 $A(x)$ 即可。

以上，我们在 $O(n \log n)$ 内完美解决了本问题。

多项式除法的应用

- 这除了算多项式除法还能干啥？
- 你看了下面的应用就知道了。

Bernoulli 数

- 求 $Bernoulli$ 数的前 N 项模 1998585857 .
- 范围
  - $n \leq 1000$?
  - $n \leq 10000$?
  - $n \leq 100000$?

Bernoulli 数介绍及朴素算法

- 相信大家对 $Bernoulli$ 数都有所了解，这个有什么用就不详述了。
- 我们考虑一下 $Bernoulli$ 数的递推式：

$$\sum_{i=0}^{n} \binom{n+1}{i} B_i = 0$$

- 当然，这是可以利用分治 +FFT 做到 $O(n \log^2 n)$ 的，用类似前面的做法就可以了。

求 Bernoulli 数一个优秀的算法

- Bernoulli 数的母函数
  - 我们不妨考虑 $Bernoulli$ 数的指数型生成函数（即其的母函数）：
  - $$G_e(x)=\sum_{i=0}^{\infty}\frac{B_i}{i!}x^i=\frac{x}{e^x-1}$$
  - 将 e^x 进行泰勒展开：
    $$e^x=\sum_{i=0}^{\infty}\frac{x^i}{i!}$$
  - $$G_e(x)=\frac{x}{(\sum_{i=0}^{\infty}\frac{x^i}{i!})-1}=\frac{x}{\sum_{i=1}^{\infty}\frac{x^i}{i!}}=\frac{1}{\sum_{i=0}^{\infty}\frac{x^i}{(i+1)!}}$$

求 Bernoulli 数一个优秀的算法

- 转换
  - 如果我们要求出 $Bernoulli$ 数的前 n 项，那么我们只要求出：
  - $$G_e(x) = \frac{1}{\sum_{i=0}^{\infty} \frac{x^i}{(i+1)!}} \pmod{x^n}$$
  - 看出了什么？直接对分母的多项式求其对 x^n 的逆元即可！$O(n \log n)$ 。

常系数线性递推

- 问题
  - 给出一个递推式：$f_i = \sum_{j=1}^{d} a_j \times f_{i-j}$ 及递推初值 $f_0, \dots, f_{d-1}$ ，求 f_n 模 1998585857 。
  - 范围
    - $n \le 10^{18}, d \le 100$
    - $n \le 10^{18}, d \le 1000$
    - $n \le 10^{18}, d \le 10000$

一个朴素算法

- 这是经典的常系数线性递推方程，我们可以使用矩阵乘法在 $O(d^3 \log n)$ 时间内解决。

一个优秀的算法

- 该做法比较复杂，这里只是简略讲一下，具体可以参见叉姐与宽大爷的论文：《矩阵乘法递推的优化》。
- 我们可以利用矩阵的特征多项式来加速这个算法。
- 根据 Cayley-Hamilton 定理，$p(M)=0$，其中 $p(\lambda)=\lambda^d-\sum_{i=0}^{d-1}a_i\lambda^i$ 是它的特征多项式。
- 由于 $M^d=\sum_{i=0}^{d-1}a_iM^i$，两边同乘 M^k 即得任意次数的转移矩阵幂。
- 即我们可以证明转移矩阵 M 的幂都可以被表示成 $M^0\sim M^{d-1}$ 的线性表示。但该如何得到这组系数？

一个优秀的算法

- 假如已经求得了 M^k 的系数，我们只需要计算这个矩阵多项式的平方，利用线性表示将不低于 d 次的项表示到 d 次以下，在 $O(d^2)$ 的复杂度算得 M^{2k} 的系数。
- 利用快速幂可以在 $O(d^2 \log n)$ 的时间内得到这组系数。
- 考虑上述算法的复杂度瓶颈在于系数在平方时花费的复杂度太大。我们尝试优化它。
- 对于乘法，我们可以直接利用 FFT 在 $O(d \log d)$ 的时间内解决多项式乘法。

一个优秀的算法

- 而关键是将多项式中超出 $d-1$ 次的项线性表示为在 $d-1$ 以内的部分。由于
$$p(M)=0$$
我们不妨设乘法的结果为 $A(x)$ ，并设 $D(x)$ 为次数不超过 d 的多项式，则有多项式 $C(x)$ 满足
$$A(x)=p(x)\times C(x)+D(x)$$
- 由于 $p(M)=0$ ，实际上我们只需要求出 $D(x)$ 。而 $D(x)$ 就是 $A(x) \bmod p(x)$ 。直接套用多项式除法在 $O(d\log d)$ 时间内解决。
- 这样子的复杂度就是 $O(d\log d\log n)$ 。

FAQ

感谢

- 感谢 CCF 提供交流的平台
- 感谢向期中老师的指导
- 感谢欧阳前宇、吕凯风、罗雨屏、杜瑜皓、郭晓旭、陈牧歌、杨宽等人的支持

引用与参考

- 陈鸿《一切皆整数的 FFT》
- 胡渊鸣《城市规划解题报告》
- 郭晓旭、杨宽《矩阵乘法递推的优化》
- 杜瑜皓《O(dlogdlogn) 的方法求常系数线性递推数列第 n 项》
- 《具体数学》