Random Number Generator 解题报告

雅礼中学 刘剑成

1 试题来源

Codechef March Challenge 2015 - RNG

2 试题大意

有一个数列A，它的前k项$A_1, A_2, A_3, ..., A_k$都是已知的，现在给出它在$i > k$时的递推式：

$$A_i = (A_{i-1} \times C_1 + A_{i-2} \times C_2 + ... + A_{i-k} \times C_k) \bmod 104857601$$

其中$104857601 = 25 \times 2^{22} + 1$，且$104857601$是一个素数。
要求该数列的第n项。
$1 \leq n \leq 10^{18}$，$1 \leq k \leq 30000$，$0 \leq A_i, C_i < 104857601$。

时间限制：15sec
空间限制：无

3 算法介绍

3.1 算法1

依照题面描述，暴力递推，从第$k+1$项开始枚举。

时间复杂度：$O(nk)$
空间复杂度：$O(k)$

3.2 算法2

用矩阵乘法优化转移，我们将转移方程和初始状态用矩阵表示出来，则可得转移矩阵G与初始矩阵B：

$$G = \begin{pmatrix} C_1 & C_2 & \cdots & C_{k-1} & C_k \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} B = \begin{pmatrix} A_k \\ A_{k-1} \\ \vdots \\ A_2 \\ A_1 \end{pmatrix}$$

接下来，我们只需要用快速幂算出$G^{n-k} \times B$即可。

时间复杂度：$O(k^3 \log n)$

空间复杂度：$O(k^2)$

3.3 算法3

我们可以用矩阵的特征多项式来优化转移的过程。

根据Cayley-Hamilton定理，有$p(G) = 0$，$p(\lambda)$表示矩阵G的特征多项式，即$p(\lambda) = \det(\lambda I_k - G)$，其中$I_k$为$k \times k$的单位矩阵，det表示行列式函数。

拆开行列式，得到$p(\lambda) = \lambda^k - \sum_{i=0}^{k-1} a_i \lambda$，我们将$\lambda = G$带入，可得：

$$p(G) = G^k - \sum_{i=0}^{k-1} a_i G \quad (1)$$

$$G^k = \sum_{i=0}^{k-1} a_i G \quad (2)$$

接下来对于任意次数的转移矩阵幂我们都可以用一个次数不超过$k-1$的矩阵多项式表示出来。

现在问题是这组系数怎么求。可以发现，当我们求出G^x所对应的系数时，G^{2x}的系数即为G^x关于G 多项式的平方，接下来只需要使用等式2即可将系数在$k-1$以上的每一项化为系数不超过$k-1$的项。

时间复杂度：$O(k^2 \log n)$

空间复杂度：$O(k)$

3.4 算法4

可以发现，在算法3中，影响复杂度的有两个部分：第一个求是多项式的平方，第二个是将次数高于$k-1$的所有项用等式2化简至最高项不超过$k-1$的多项式。

注意到本题的模数比较特殊，所以对于多项式的平方，我们直接使用快速傅里叶变化（FFT）优化乘法即可，时间复杂度为$O(k\log k)$ [1]。

对于第二部分，观察每次的操作，即减去了若干倍G^k，再加上若干倍的$a_{k-1}G^{k-1}+a_{k-2}G^{k-2}+...+a_0G^0$，直到最终不存在次数大于$k-1$的项。

可以发现这个就是多项式取模的过程，即将平方后的多项式对于如下多项式取模：

$$G^k-a_{k-1}G^{k-1}-a_{k-2}G^{k-2}-...-a_0G^0$$

同样的，我们可以将多项式乘法经过变化之后$O(k\log k)$解决多项式取模的问题。

3.4.1 多项式求逆

在了解多项式取模之前，我们先需要知道多项式如何求逆。

对于一个多项式$A(x)$，它在模x^T下的逆元$P_T(x)$，即一个多项式$P_T(x)$满足

$$P_T(x)\times A(x) \equiv 1 \pmod{x^T} \tag{3}$$

当$T=1$时，我们可以直接对常数项取逆元。

当$T>1$时，我们使用递归来处理此问题。

假设$P_{\lceil\frac{T}{2}\rceil}(x)$已经被求出，根据已知，我们有：

$$P_{\lceil\frac{T}{2}\rceil}(x)\times A(x) \equiv 1 \pmod{x^{\lceil\frac{T}{2}\rceil}} \tag{4}$$

$$P_T(x)\times A(x) \equiv 1 \pmod{x^T} \tag{5}$$

[1]不知道FFT的同学可以戳这里http://en.wikipedia.org/wiki/Fast_Fourier_transform

用等式4减去等式5，得到：

$$\begin{aligned}
A(x)\left[P_{\lceil \frac{T}{2} \rceil}(x) - P_T(x)\right] &\equiv 0 \pmod{x^{\lceil \frac{T}{2} \rceil}} &(6)\\
P_{\lceil \frac{T}{2} \rceil}(x) - P_T(x) &\equiv 0 \pmod{x^{\lceil \frac{T}{2} \rceil}} &(7)\\
\left[P_{\lceil \frac{T}{2} \rceil}(x) - P_T(x)\right]^2 &\equiv 0 \pmod{x^T} &(8)\\
P^2_{\lceil \frac{T}{2} \rceil}(x) + P^2_T(x) - 2P_{\lceil \frac{T}{2} \rceil}(x) \times P_T(x) &\equiv 0 \pmod{x^T} &(9)
\end{aligned}$$

两边同时乘以$A(x)$，由等式5可得：

$$P_T(x) + A(x) \times P^2_{\lceil \frac{T}{2} \rceil}(x) - 2A(x) \times P_{\lceil \frac{T}{2} \rceil}(x) \equiv 0 \pmod{x^T} \tag{10}$$

即转移方程为：

$$P_T(x) = \left[2A(x) \times P_{\lceil \frac{T}{2} \rceil}(x) - A(x) \times P^2_{\lceil \frac{T}{2} \rceil}(x)\right] \bmod x^T$$

后面的多项式乘法可以使用FFT来解决，由于每次递归都会使多项式的长度减半，所以对T次多项式求逆的复杂度为$O(T \log T)$。

3.4.2 多项式取模

假设我们将一个n次多项式$A(x)$对于m次多项式$B(x)$取模($m \le n$)，相当于求出多项式$D(x)$，满足：

$$A(x) = B(x) \times C(x) + D(x) \tag{11}$$

其中$D(x)$的最高项为$m-1$，则$C(x)$的最高项为$n-m$。

假设将一个k次多项式$M(x)$反转之后的多项式为$M^R(x)$，即：

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

将$\frac{1}{x}$代入多项式，则有：

$$\begin{aligned}
A(\frac{1}{x}) &= B(\frac{1}{x}) \times C(\frac{1}{x}) + D(\frac{1}{x}) &(12)\\
x^n A(\frac{1}{x}) &= x^n B(\frac{1}{x}) \times C(\frac{1}{x}) + x^n D(\frac{1}{x}) &(13)\\
A^R(x) &= B^R(x) \times C^R(x) + x^{n-m+1} D^R(x) &(14)
\end{aligned}$$

将整个等式对x^{n-m+1}取模，可得：

$$A^R(x) \equiv B^R(x) \times C^R(x) \pmod{x^{n-m+1}} \tag{15}$$

$$C^R(x) \equiv A^R(x) \times \left[B^R(x)\right]^{-1} \pmod{x^{n-m+1}} \tag{16}$$

对于$\left[B^R(x)\right]^{-1}$，我们可以直接使用多项式求逆在$O(n\log n)$解决。

接下来，我们只需要使用多项式乘法求出$C^R(x)$，将其翻转即为$C(x)$，代回等式11中即可求出$D(x)$，总时间复杂度为$O(n\log n)$

接下来只需要在每次倍增的时候使用多项式取模即可在$O(k\log k\log n)$的时间内解决此题。

时间复杂度：$O(k\log k\log n)$

空间复杂度：$O(k)$