Random Number Generator 解题报告

第一部分（10%）$n \le 10^{18}, k \le 3000$

记矩阵$\begin{bmatrix} A_k \\ \dots \\ A_2 \\ A_1 \end{bmatrix}$为$W_i$，记转移矩阵$\begin{bmatrix} C_1 & \dots & C_{k-1} & C_k \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}$为$P$，则$W_i = P^{i-1}W_1$。因此，我们可以用矩阵快速幂在$\Theta(k^3 \log n)$的时间里求出$W_n$，从而求出$A_n$。但这个复杂度明显会超时。

我们发现，由于$A_i = C_1 A_{i-1} + \dots + C_k A_{i-k}$，所以$W_{i+1} = C_1 W_i + \dots + C_k W_{i-k+1}$，即$P^i W_1 = C_1 P^{i-1} W_1 + \dots + C_k P^{i-k} W_1$，将等式两边同时乘上$W_1$的逆矩阵，我们就得到了用$P^{i-1}, \dots, P^{i-k}$来表示$P^i$的式子。进过若干次降次后，我们可以用$P^{k-1}, \dots, P^0$来表示$P^i$。

我们采用倍增的方法来求出P^{n-1}：假设我们已经知道关于P^i的 k-1 次多项式，那么我们计算$P^{2i} = P^i \times P^i$，就得到了一个 2k-2 次的多项式，暴力将其降为 k-1 次即可。

得到P^{n-1}后，我们就可以很容易地求出W_n，进而求出A_n。

时间复杂度：$\Theta(k^2 \log n)$

第二部分（90%）$n \le 10^{18}, k \le 30000$

我们发现，在上面的算法中，有且仅有多项式乘法和降次的时间复杂度达到了$\Theta(k^2)$，下面，我们尝试将它们优化到$\Theta(k \log k)$。

多项式乘法使用快速傅里叶变换（FFT）即可。

在降次方面，我们不断使用$P^i = C_1 P^{i-1} + \dots + C_k P^{i-k}$来降次，实际上相当于将原多项式对$P^k - C_1 P^{k-1} - \dots - C_k P^0$取模。下面介绍一种优秀的多项式取模算法。

假设我们现在想知道 n 次多项式 f(x)模 m 次多项式 g(x)的结果。

设商为 n-m 次多项式 q(x)，余数为 m-1 次多项式 r(x)，则 f(x)=g(x)q(x)+r(x)，所以$x^n f(\frac{1}{x}) = (x^m g(\frac{1}{x}))(x^{n-m} q(\frac{1}{x})) + x^{n-m+1}(x^{m-1} r(\frac{1}{x}))$，记$x^n f(\frac{1}{x})$为$f'(x)$，则$f'(x) = g'(x)q'(x) + x^{n-m+1} r'(x)$，注意到$f'(x)$依旧是 n 次多项式，只是系数与 f(x)全部相反了，g,q,r 也是这样，因此$q'(x) \equiv f'(x)(g'(x))^{-1} (\mathrm{mod}\, x^{n-m+1})$。至此，我们只需求出$g'(x)$

在模 x^{n-m+1} 下的逆元即可。

设 $g'(x)h_i(x)\equiv 1(\bmod x^{2^i})$，易知 $h_0(x)=1$。而由于 $g'(x)(g'(x)h_{i-1}(x)^2-2h_{i-1}(x))\equiv 1-(g'(x)h_{i-1}(x)-1)^2\equiv 1(\bmod x^{2^i})$，所以 $h_i(x)\equiv g'(x)h_{i-1}(x)^2-2h_{i-1}(x)(\bmod x^{2^i})$。因此我们可以用倍增的方法求出 $h_t(x)$，其中 t 是满足 $n-m+1\le 2^t$ 的最小整数。由于 $g'(x)h_t(x)\equiv 1(\bmod x^{2^t})$，所以 $g'(x)h_t(x)\equiv 1(\bmod x^{n-m+1})$。在本题中，n=2k-2,m=k,n-m+1=k-1。至此，一次操作的复杂度降为 $\Theta(k\log^2 k)$。

我们可以先预处理出 $h_t(x)$，每次操作时直接使用即可。

时间复杂度：$\Theta(k\log k\log n)$