Computing the Nth term of a P-finite sequence

1 Introduction

We have seen before how to efficiently compute the Nth term of a C-finite sequence.

		C-finite (lecture 4)	P-finite
first N terms	arith	$\mathrm{O}(N)$	$\mathrm{O}(N)$
	bit	$\mathrm{O}(N^2)$	$\boldsymbol{\tilde{\mathrm{O}}(N^2)}$
Nth term	arith	$\mathrm{O}(\log N)$	$\boldsymbol{\tilde{\mathrm{O}}(\sqrt{N})}$
	bit	$\mathrm{O}(\mathrm{M}(N))$	$\boldsymbol{\tilde{\mathrm{O}}(N)}$

In the P-finite case, one can obviously compute the first N terms in $\mathrm{O}(N)$ arithmetic operations.

Over $\mathbb{Z}$, the bit size of the Nth term is $\mathrm{O}(N \log N)$ and can reach $\Omega(N \log N)$. The naive method for computing the first N terms takes

$$\leqslant \mathrm{M}(C \cdot \log N) + 2\,\mathrm{M}(C \cdot \log N) + 3\,\mathrm{M}(C \cdot \log N) + \cdots + N\,\mathrm{M}(C \cdot \log N) = \mathrm{O}(N^2\,\mathrm{M}(\log N))$$

bit ops. As the total size of the output can reach $\Omega(N^2)$, it is already quasi-optimal.

For a single term, the complexity is far from linear in the output size, but the fast methods (e.g., using binary powering) that work in the C-finite case do not apply.

Remark. Algorithms for computing the Nth term of a P-finite sequence apply to coefficients but also *partial sums* $\sum_{n=0}^{N-1} y_n x^n$ of D-finite series and are thus useful fo evaluating D-finite functions. ◃

Definition. We will say that the recurrence

$$b_s(n)\,u_{n+s} + \cdots + b_0(n)\,u_n = 0 \qquad \text{(REC)}$$

is *nonsingular* if $b_s(n) \neq 0$ for $n \in \mathbb{N}$. ◃

Assumption. For simplicity we will restrict ourselves here to nonsingular recurrences. (Everything generalizes to singular recurrences along the lines of what we saw in the other half of the lecture.) ◃

2 A baby steps-giant steps algorithm

We start with the problem of evaluating u_N in good *arithmetic* complexity.

2.1 Case of $N!$

Suppose first $N = \ell^2$. Write

$$N! = \underbrace{\underbrace{1 \times 2 \times \cdots \times \ell}\times\underbrace{(\ell+1)\times(\ell+2)\times\cdots\times(2\,\ell)}\times\cdots\times\underbrace{(\ell^2-\ell+1)\times\cdots\times\ell^2}}_{\sqrt{N}\text{ blocks of }\sqrt{N}\text{ factors.}}$$

Algorithm. Input: N, output: $N!$ in $\mathbb{K}$ (e.g., $\mathbb{K} = \mathbb{Z}/p\,\mathbb{Z}$)

1. Let $\ell = \lfloor\sqrt{N}\rfloor$.
2. Compute $F(X) = (X+1)\cdots(X+\ell) \in \mathbb{K}[X]$. $\mathrm{O}(\mathrm{M}(\ell)\log\ell)$
   (*baby steps*)
3. Evaluate F at $0, \ell, 2\,\ell, \ldots, (\ell-1)\,\ell$. $\mathrm{O}(\mathrm{M}(\ell)\log\ell)$
   (*giant steps*)
4. Multiply: $F(0)\cdot F(1)\cdots F((\ell-1)\,\ell)\cdot(\ell^2+1)\cdots N$, return. $\mathrm{O}(\ell)$

Total $\mathrm{O}(\mathrm{M}(\ell)\log\ell) = \mathrm{O}(\mathrm{M}(\sqrt{N})\log N)$. ◁

This simple algorithm has far-reaching consequences, such as the following.

Theorem. There is a *deterministic* integer factoring algorithm that runs in time

$$N^{1/4+o(1)}.$$ ◁

The exponent $1/4$ here was the best known until 2020!

Proof.

- If N is composite, then $N \wedge (\lfloor\sqrt{N}\rfloor!)$ is a proper factor.
- One can compute $\lfloor\sqrt{N}\rfloor! \bmod N$ in $\tilde{\mathrm{O}}(N^{1/4})$ **bit** operations ($\tilde{\mathrm{O}}(N^{1/4})$ arith ops each of cost $\mathrm{O}(\mathrm{M}(\log N))$) using the previous algorithm over $\mathbb{Z}/N\,\mathbb{Z}$.
- The gcd costs $\tilde{\mathrm{O}}(\log N)$. □

Remark. Slightly better: skip step 4 and take $\mathrm{O}(N^{1/4})$ separate gcds. ◁

2.2 General case of P-finite sequences

The same idea applies after reducing to a first-order recurrence.
More precisely, given (u_n) satisfying (REC), write

$$\begin{bmatrix} u_{n+1} \\ \vdots \\ u_{n+s-1} \\ u_{n+s} \end{bmatrix} = -\frac{1}{b_s}\underbrace{\begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ b_0 & \cdots & \cdots & b_{s-1} \end{bmatrix}}_{B(n)\in\mathbb{K}[n]_{\leqslant d}^{s\times s}}\underbrace{\begin{bmatrix} u_n \\ \vdots \\ u_{n+s-2} \\ u_{n+s-1} \end{bmatrix}}_{U_n}. \qquad \text{(MREC)}$$

Then

$$U_N = B(N-1)\cdots B(1)\,B(0)\,U_0.$$