Formal Power Series and Generating Functions

Jubayer Nirjhor

January 2019

Prerequisites

- High School Calculus
- Basic Dynamic Programming
- Fast Fourier Transform (only for the applications)

Introduction

"A generating function is a device somewhat similar to a bag. Instead of carrying many little objects detachedly, which could be embarrassing, we put them all in a bag, and then we have only one object to carry, the bag." - Polya

Formal Power Series

Let's get down to some basics first. A **Formal Power Series** is simply a series of form

$$A(x) = a_0 + a_1x + a_2x^2 + \cdots = \sum_{k=0}^{\infty} a_k x^k$$

where $\{a_0, a_1, a_2, ...\}$ is a sequence (finite or infinite). Loosely speaking, you can think of it as a polynomial with infinitely many terms where we do not care about the convergence at different values of x, only the coefficients. So you don't get to ask what happens when you put in $x = 0$ or something: it does not mean anything.

We can add, subtract, and multiply two formal power series $A(x) = \sum_{k\geq 0} a_k x^k$ and $B(x) = \sum_{k\geq 0} b_k x^k$ similarly as we do in case of polynomials

$$A(x) + B(x) = \sum_{k\geq 0} (a_k + b_k)\, x^k, \quad A(x) - B(x) = \sum_{k\geq 0} (a_k - b_k)\, x^k, \quad A(x)B(x) = \sum_{k\geq 0} \left(\sum_{i+j=k} a_i b_j \right) x^k.$$

Similar to real numbers, we define the inverse of a formal power series $A(x)$ as a formal power series $B(x)$ such that $A(x)B(x) = 1$ and denote $B(x)$ by $A^{-1}(x)$. Notice that if $A(x) = \sum_{k\geq 0} a_k x^k$ has $a_0 = 0$, that is, $A(x) = a_1x + a_2x^2 + \cdots = x\left(a_1 + a_2x + a_3x^2 + \cdots\right)$, then multiplying it by any formal power series will produce a multiple of x. So $A(x)$ does not have an inverse (similar to 0 in $\mathbb{R}$). In fact, the converse turns out to be true as well.

Theorem. *A formal power series $A(x) = \sum_{k\geq 0} a_k x^k$ has an inverse if and only if $a_0 \neq 0$.*

As an example, $A(x) = x$ does not have an inverse: $1/x$ does not exist. But $A(x) = 1 - x$ does have an inverse. We can quickly verify:

$$(1-x)\left(1+x+x^2+\cdots\right) = \left(1+x+x^2+\cdots\right) - \left(x+x^2+x^3+\cdots\right) = 1$$

and so $1/(1-x) = 1 + x + x^2 + \cdots$. It's pretty easy to prove that the inverse, if exists, is unique (try it). Equipped with this, we can now define division: $A(x)/B(x) = A(x)B^{-1}(x)$ whenever the inverse of $B(x)$ exists. We can define a few more operations similarly: like $\sqrt{A(x)} = B(x)$ if and only if $B(x)^2 = A(x)$. Differentiation and integration (called *formal differentiation* and *formal integration*) are also defined as is in case of polynomials:

$$\frac{d}{dx}A(x) = \sum_{k\geq 1} k a_k x^{k-1}, \quad \int A(x)\,dx = \sum_{k\geq 0} \frac{a_k}{k+1} x^{k+1}.$$

One last definition: we define by $A(x) \bmod x^n$ the first n terms of $A(x)$ where $n \in \mathbb{N}$. For example: $A(x) \bmod x = a_0$ and $A(x) \bmod x^3 = a_0 + a_1 x + a_2 x^2$.

Ordinary Generating Function

Given a sequence $a = \{a_0, a_1, a_2, ...\}$, its ordinary generating function (OGF) $A(x)$, for all we care about, is the formal power series with coefficients a_i (this is not the actual definition of generating functions, but we can always reduce to this one). How does this representation help us? Let's consider the classic problem of finding the n-th Fibonacci number. Recall that the Fibonacci sequence is defined by $f_0 = 0$, $f_1 = 1$ and $f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$. It's OGF, then, is given by $F(x) = \sum_{k\geq 0} f_k x^k$. Using the recurrence relation, let's manipulate it a bit:

$$F(x) = \sum_{k\geq 0} f_k x^k = 0 + x + \sum_{k\geq 2} f_k x^k = x + \sum_{k\geq 2} \left(f_{k-1} + f_{k-2}\right) x^k = x + \sum_{k\geq 2} f_{k-1} x^k + \sum_{k\geq 2} f_{k-2} x^k$$

$$= x + x\sum_{k\geq 2} f_{k-1} x^{k-1} + x^2 \sum_{k\geq 2} f_{k-2} x^{k-2} = x + x\sum_{k\geq 1} f_k x^k + x^2 \sum_{k\geq 0} f_k x^k = x + xF(x) + x^2 F(x).$$

So we have $F(x) = x + xF(x) + x^2F(x)$. Make sure you understood each step (we've used the fact that $f_0 = 0$ to obtain $xF(x)$). We can now easily solve for $F(x)$ and obtain $F(x) = x/\left(1 - x - x^2\right)$. Out of nowhere we've got ourselves an expression for the generating function of f that does not contain any f_i. This little expression 'enumerates' all the Fibonacci numbers in itself. From the previous section, we know that the formal power series $1 - x - x^2$ has an inverse. Let's find it. We can factorize $1 - x - x^2 = (1 - \alpha x)(1 - \beta x)$ where $\alpha = \left(1 + \sqrt{5}\right)/2$ and $\beta = \left(1 - \sqrt{5}\right)/2$. Then:

$$F(x) = \frac{x}{1-x-x^2} = \frac{x}{(1-\alpha x)(1-\beta x)} = \frac{1}{\alpha-\beta}\left(\frac{1}{1-\alpha x} - \frac{1}{1-\beta x}\right) = \frac{1}{\alpha-\beta}\left(\sum_{k\geq 0}\alpha^k x^k - \sum_{k\geq 0}\beta^k x^k\right)$$

$$\implies F(x) = \sum_{k\geq 0} \frac{\alpha^k - \beta^k}{\alpha - \beta} x^k = \sum_{k\geq 0} f_k x^k \quad \implies f_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} = \frac{\left(1+\sqrt{5}\right)^n - \left(1-\sqrt{5}\right)^n}{2^n\sqrt{5}}.$$

Notice that we've used $(1-z)^{-1} = \sum_{k\geq 0} z^k$ from the previous section. We can apply this approach to any linear recurrence of form $a_n = ca_{n-1} + da_{n-2}$ and in fact, to any recursively defined sequence. As an exercise, solve `Timus 1605` finding a closed form expression first.

Another well-known generating function is given by the generalized Binomial Theorem:

$$(1+x)^r = \sum_{k\geq 0} \binom{r}{k} x^k, \quad \left[\binom{r}{k} = \frac{r(r-1)\cdots(r-k+1)}{k!}, \;\; r \in \mathbb{R},\; k \in \mathbb{N}\right].$$

Let's consider another classic problem: counting the number of binary trees on n nodes. Denoting this number by t_n (considering the empty tree to be valid), we can formulate the following recurrence relation:

$t_0 = 1, \quad t_n = \sum_{i+j=n-1} t_i t_j$. Here we simply distribute all the nodes except the root: i nodes to the left subtree and j childs to the right. Let's try to find the generating function $T(x)$:

$$T(x) = \sum_{k\geq 0} t_k x^k = 1 + \sum_{k\geq 1} t_k x^k = 1 + \sum_{k\geq 1} \left(\sum_{i+j=k-1} t_i t_j \right) x^k$$

$$= 1 + x \sum_{k\geq 1} \left(\sum_{i+j=k-1} t_i t_j \right) x^{k-1} = 1 + x \sum_{k\geq 0} \left(\sum_{i+j=k} t_i t_j \right) x^k = 1 + xT(x)^2.$$

Don't be surprised! We've applied the product formula of two formal power series from the previous section. So we have $T(x) = 1 + xT(x)^2 \implies T(x) = \left(1 \pm \sqrt{1-4x}\right)/2x$. Recall that $1/x$ does not exist as a formal power series. So we simplify further:

$$T(x) = \frac{1 \pm \sqrt{1-4x}}{2x} = \frac{\left(1 \pm \sqrt{1-4x}\right)\left(1 \mp \sqrt{1-4x}\right)}{2x\left(1 \mp \sqrt{1-4x}\right)} = \frac{1-(1-4x)}{2x\left(1 \mp \sqrt{1-4x}\right)} = \frac{2}{1 \mp \sqrt{1-4x}}.$$

A really good question right now would be: what sign do we take? This question takes us back to the very basics. We do know that $T(x)$ is a formal power series, so the right side must be a formal power series as well. What condition must we impose on the right side expression to make that sure? We need the denominator to be invertible! But $\sqrt{1-4x}$ starts with a 1, so $1 - \sqrt{1-4x}$ would have the constant term 0 making it not invertible (see previous section). So we must take the plus sign, and hence $T(x) = 2\left(1 + \sqrt{1-4x}\right)^{-1}$. Expanding this expression as a formal power series is still a to-do: we can first invert it as $(1+z)^{-1}$ and then expand $\sqrt{1-4x}$ using Binomial Theorem. But that's too much work. We don't want to do the dirty work by hand, instead let the computer do it.

Algorithms for the Dirty Work

In most of the problems, we can figure out the generating function by hand, but expanding the function as a single formal power series is tedious. The function often involves product, inverse, square root or composition of several such operations on formal power series. In this section, we look at some efficient algorithms to perform some of these operations. We focus on generating the first n terms of the resultant series in reasonable time, when we're given the first needed $\mathcal{O}(n)$ terms of the operand series.

- **Addition and Scaling.** To find $A(x) + B(x),\ A(x) - B(x),\ cA(x)$: we can simply iterate over the coefficients. Complexity: $\mathcal{O}(n)$.
- **Formal Product.** To find $A(x)B(x)$: we can consider $A(x)$ and $B(x)$ as polynomials with the first n coefficients and perform Fast Fourier Transform to get the first n terms of the product series. Complexity: $\mathcal{O}(n \lg n)$.
- **Formal Derivative and Integral.** To find $\frac{d}{dx}A(x),\ \int A(x)\,dx$: again, we can simply iterate and update. Complexity: $\mathcal{O}(n)$.
- **Inverse.** We'll find inverse of $A(x)$. Let's denote $B_k(x) = A^{-1}(x) \bmod x^k$. $B_1(x)$ is simply $1/a_0$. Now given $B_k(x)$ we can compute $B_{2k}(x)$ as follows:

$$AB_k \equiv 1 \quad \implies AB_k - 1 \equiv 0 \pmod{x^k}$$

$$\implies (AB_k - 1)^2 \equiv 0 \quad \implies A^2B_k^2 - 2AB_k + 1 \equiv 0 \pmod{x^{2k}}$$

$$\implies 1 \equiv 2AB_k - A^2B_k^2 \quad \implies B_{2k} \equiv 2B_k - AB_k^2 = B_k\left(2 - AB_k\right) \pmod{x^{2k}}$$

Notice that squaring doubles the number of terms in the series, which is why this works. Also notice the last step where we multiply both sides by the inverse of $A(x)$ to product a $B_{2k}(x)$ on the left. Complexity: $\mathcal{O}(n \lg n)$.

- **Square Root.** We'll find the square root of $A(x)$. Let's denote $B_k(x) = \sqrt{A(x)} \bmod x^k$. Again, we can take $B_1(x) = \sqrt{a_0}$ (so a_0 must be nonnegative). And given $B_k(x)$ we can compute $B_{2k}(x)$ as follows (notice where we use the identity $(a-b)^2 + 4ab = (a+b)^2$):

$$B_k^2 \equiv A \quad \Longrightarrow B_k^2 - A \equiv 0 \pmod{x^k}$$

$$\left(B_k^2 - A\right)^2 \equiv 0 \quad \Longrightarrow \left(B_k^2 + A\right)^2 \equiv 4AB_k^2 \quad \Longrightarrow \left(B_k + AB_k^{-1}\right)^2 \equiv 4A \pmod{x^{2k}}$$

$$\Longrightarrow B_k + AB_k^{-1} \equiv 2B_{2k} \quad \Longrightarrow B_{2k} \equiv \frac{B_k + AB_k^{-1}}{2} \pmod{x^{2k}}$$

  Notice the last step where we take the square root of both sides to produce B_{2k} on the right side. Complexity: $\mathcal{O}(n \lg n)$.

Newton-Raphson Method of Iteration

...

Exponential Generating Function

...

Problems and Solutions

...