$xB'(x) = xB(x)e^x$, which implies $\frac{B'(x)}{B(x)} = e^x$. If you are familiar with calculus, you will recognize that if we integrate both sides, we get $\ln B(x) = e^x + c$. Since $b_0 = 1$, $B(0) = 1$ and we have $c = -1$. Thus, $B(x) = e^{e^x - 1}$ is our desired EGF.

So, how to find b_n in faster than $O(n^2)$ time. The idea is that we can find the first n terms of $e^{P(x)}$ in $O(n \log n)$ time, so we just need to compute the first few terms of our EGF and read off the answer! In this 2-part article, I will omit explaining how to do certain well-known polynomial operations in $O(n \log n)$ time or $O(n \log^2 n)$ time like $\sqrt{P(x)}$, $\ln(P(x))$ etc. There are already tutorials written for them (for example cp-algorithms). Hence, I will just quote that we can do those polynomial operations since that is not the main focus of this article.

Algebraic Manipulation of Generating Functions

Here are some common ways to manipulate generating functions and how they change the sequence they are representing. In this section, a_i, b_i will represent sequences and $A(x)$ and $B(x)$ are their corresponding generating functions (OGF or EGF depending on context which will be stated clearly). As an exercise, verify these statements.

Addition

For both OGF and EGF, $C(x) = A(x) + B(x)$ generates the sequence $c_n = a_n + b_n$.

Shifting

For OGF, $C(x) = x^k A(x)$ generates the sequence $c_n = a_{n-k}$ where $a_i = 0$ for $i < 0$. For EGF, you need to intergrate the series $A(x)$ k times to get the same effect.

For OGF, $C(x) = \frac{A(x) - (a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1})}{x^k}$ generates the sequence $c_n = a_{n+k}$.

For EGF, $C(x) = A^{(k)}(x)$ generates the sequence $c_n = a_{n+k}$, where $A^{(k)}(x)$ denotes A differentiated k times.

Multiplication by n

For both OGF and EGF, $C(x) = xC'(x)$ generates the sequence $c_n = na_n$.

In general, you can get the new generating function when you multiply each term of the original sequence by a polynomial in n by iterating this operations (but I do not include the general form here to avoid confusion).

Convolution

This is really the most important operation on generating functions.

For OGF, $C(x) = A(x)B(x)$ generates the sequence $c_n = \sum_{k=0}^{n} a_k b_{n-k}$.

For EGF, $C(x) = A(x)B(x)$ generates the sequence $c_n = \sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k}$ (verify this!).
This is also why EGF is useful in dealing with recurrences involving binomial coefficients or factorials.

Power of Generating Function

This is just a direct consequence of convolution, but I include it here because it is so commonly used.

For OGF, $C(x) = A(x)^k$ generates the sequence $c_n = \sum_{i_1 + i_2 + \ldots + i_k = n} a_{i_1} a_{i_2} \ldots a_{i_k}$

For EGF, $C(x) = A(x)^k$ generates the sequence
$c_n = \sum_{i_1+i_2+\ldots+i_k=n} \frac{n!}{i_1!i_2!\ldots i_k!} a_{i_1}a_{i_2}\ldots a_{i_k}$

Prefix Sum Trick

This only works for OGF, but is useful to know. Suppose want to generate the sequence $c_n = a_0 + a_1 + \ldots + a_n$. Then, we can take $C(x) = \frac{1}{1-x}A(x)$.

Why does this work? If we expand the RHS, we get $(1 + x + x^2 + \ldots)A(x)$. To obtain the coefficient of x^n which is c_n, we need to choose x^i from the first bracket and $a_{n-i}x^{n-i}$ from $A(x)$, so summing over all i gives us $c_n = \sum_{i=0}^{n} a_i$.

List of Common Series

Before we delve into applications, I want to compile a short list of series that we will use frequently below. They are

$\frac{1}{1-x} = 1 + x + x^2 + \ldots = \sum_{n\geq 0} x^n$

$-\ln(1-x) = x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots = \sum_{n\geq 1} \frac{x^n}{n}$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{n\geq 0} \frac{x^n}{n!}$

$(1-x)^{-k} = \binom{k-1}{0}x^0 + \binom{k}{1}x^1 + \binom{k+1}{2}x^2 + \ldots = \sum_{n} \binom{n+k-1}{n} x^n$

Our goal in many problems will be to reduce the EGF or OGF involved in the problem into some composition of functions that we know above.

You can find a more complete list on Page 57 on generatingfunctionology.

Generating Functions in Counting Problems

Generating functions is a powerful tool in enumerative combinatorics. There are so many applications that I can only cover a small fraction of them here. If you are interested in more examples of counting using generating functions, you can try the books generatingfunctionology and Enumerative Combinatorics.

Here, I will show some classical examples of counting problems involving generating functions. In the next post, I will focus on CP problems which utilizes generating functions.

Catalan Numbers, revisited

We have shown before that the OGF of the Catalan numbers is $C(x) = \frac{1-\sqrt{1-4x}}{2x}$. Suppose we want to find a closed-form formula for c_n. Of course, it is well-known that $c_n = \frac{1}{n+1}\binom{2n}{n}$, but let's pretend we don't know that yet. We want to "expand" our generating function $C(x)$, but there is a troublesome square root in our way.

This is where the **generalized binomial theorem** comes to our rescue. Before that, we need to define generalized binomial coefficients.

Definition. Let r be any complex number and n be a nonnegative integer. Then, $\binom{r}{n} = \frac{r(r-1)\ldots(r-(n-1))}{n!}$.

This is the same as the usual binomial coefficients, but now we no longer require the first term to be a nonnegative integer.