Hi everyone! Inspired by the recent Codeforces Round 641, I decided to write an introductory tutorial on generating functions here. I am by no means an expert in generating functions so I will write about what I currently know about them. jqdai0815 has written a really interesting blog here on Codeforces about more advanced applications of generating functions, but I think there is no English tutorial on the basics of this topic yet (or at least on CP sites). Thus, I would like to share about this topic here.

I plan to split this tutorial into two parts. The first part (this post) will be an introduction to generating functions for those who have never learned about them at all, and some standard examples and showcases of generating functions. The second part will be a collection of several applications of generating functions in CP-style problems. If you are already familiar with generating functions, you may just skip to Part 2 directly.

In this part, many examples are from the book generatingfunctionology which I think is also a great introductory material to generating functions.

What are Generating Functions?

Let's say we have a sequence $a_0, a_1, a_2, \ldots$. We associate with a a series A which "encodes" the terms in a with its coefficients.

Formally, for a sequence of numbers $\{a_i\}_{i=0}^{\infty}$, we define the **ordinary generating function (OGF)** of a to be $A(x) = \sum\limits_{i=0}^{\infty} a_i x^i$.

For example, consider the Fibonacci sequence f with the terms $0, 1, 1, 2, 3, 5, 8, \ldots$. Then, $F(x) = 0 + x + x^2 + 2x^3 + 3x^4 + 5x^5 + 8x^6 + \ldots$.

You may imagine that you are putting the (infinitely many) terms of the sequence in a line, and assigning a power of x to each term of the sequence in order. Adding them up, you get an “infinite polynomial” which somewhat encodes the sequence. The nice thing about generating functions is that sometimes the series is nicer to play around with which will sometimes uncover surprising properties of the sequence.

There are other types of generating functions, such as the Exponential Generating Function (EGF) and Dirichlet Generating Function. We will look at some examples of EGF later in this post, but for the next few examples we will focus on OGFs.

Before that, we introduce a simple notation. For a series $A(x) = \sum\limits_{n \geq 0} a_n x^n$, we let $[x^n]A(x) = a_n$ (i.e. the coefficient of x^n in A).

Simple Examples of OGFs

Let’s start with a very simple example. What is the OGF of the sequence $1, 1, 1, \ldots, 1$? By definition, we have $A(x) = 1 + x + x^2 + x^3 + \ldots$. Does this series look familiar?

$A(x)$ is actually a geometric series with common ratio x! Thus, we can use the geometric series formula to write $A(x) = \frac{1}{1-x}$.

Note: Don’t we need to care about convergence issues like $|x| < 1$? Well, it depends on what you want to do with your series. We will work on formal power series most of the time, which allows us to ignore the issue of convergence. However, this also means that we can’t “substitute” x as some fixed value without care. For example, when $x = 2$, the series $A(x)$ diverges but for say $x = -\frac{1}{2}$, $A(x)$ converges. In this post, we won’t deal with the analytic properties of the series most of the time. If you really need to substitute values though, the general rule of thumb is that you can do it if the series converges.

We can manipulate generating functions very much like how we would manipulate other algebraic expressions. Here is a classic example.

Example. Consider the sequence f_n defined by $f_0 = 0$, $f_1 = 1$ and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$. Find the OGF of f (we usually denote it with a capital letter, say F).

Clearly, f_n is the n-th Fibonacci number. We will use the recurrence relation to find the OGF of f_n.

Firstly, we need to make the terms of the series appear. The easiest way to do this is to multiply the recurrence relation by x^n to obtain $f_n x^n = f_{n-1} x^n + f_{n-2} x^n$.

Next, we sum up the terms on both sides over all valid n (in this case $n \geq 2$) to obtain:

$$\sum_{n=2}^{\infty} f_n x^n = x \sum_{n=2}^{\infty} f_{n-1} x^{n-1} + x^2 \sum_{n=2}^{\infty} f_{n-2} x^{n-2}.$$

This is equivalent to:

$$F(x) - f_0 x^0 - f_1 x^1 = x(F(x) - f_0 x^0) + x^2 F(x).$$

$$\Rightarrow F(x) - x = (x + x^2)F(x)$$

$$\Rightarrow F(x)(1 - x - x^2) = x$$

$$\Rightarrow F(x) = \frac{x}{1-x-x^2}$$

Thus, we obtain the OGF of Fibonacci numbers.

Let’s see another example of OGF of a common sequence.

Example. The Catalan numbers c_n are defined by $c_0 = 1$ and $c_{n+1} = \sum_{i=0}^{n} c_i c_{n-i}$ for $n \geq 0$. Find the OGF of c_n.

Again, our strategy is to multiply a power of x to both sides and summing up for all n. We obtain:

$$\sum_{n=0}^{\infty} c_{n+1}x^{n+1} = \sum_{n=0}^{\infty}\sum_{i=0}^{n} c_i c_{n-i} x^{n+1} = x\sum_{n=0}^{\infty}\sum_{i=0}^{n} c_i x^i c_{n-i}x^{n-i}$$

The LHS is easy to intepret: it is just $C(x) - 1$.

How do we interpret the RHS? We claim that it is $xC(x)^2$. Consider the expansion of $C(x)^2$. Which terms contribute to the coefficient of x^n? If we look at $C(x)^2 = (c_0 + c_1x + c_2x^2 + \ldots)(c_0 + c_1x + c_2x^2 + \ldots)$, we see that we can only obtain x^n by picking c_ix^i from the first bracket and $c_{n-i}x^{n-i}$ from the second bracket. Hence, the coefficient of x^n in $C(x)^2$ is $\sum_{i=0}^{n} c_i c_{n-i}$, as desired.

Hence, we have $C(x) - 1 = xC(x)^2$, which is a quadratic equation in $C(x)$! Using the quadratic formula, we can obtain $C(x) = \frac{1 \pm \sqrt{1-4x}}{2x}$. Which sign should we choose? If we choose the + sign, then the numerator $\to 2$ as $x \to 0$, while the denominator $\to 0$, so the ratio will become infinite at 0. However, $C(x)$ can be expanded as a power series at $x = 0$, so $C(x)$ should converge at $x = 0$. Thus, we should choose the minus sign to obtain $C(x) = \frac{1-\sqrt{1-4x}}{2x}$ (indeed, by L'Hopital Rule it converges to $c_0 = 1$ at $x = 0$).

Tip: Try looking for common sequences and see if you can derive the formula for their OGFs from scratch. It is really helpful if you can derive the intuition where you can **see** the functional equation directly from the recurrence.

OGFs in more than one variable

We don’t have to limit ourselves to one variable. We can have multivariable OGFs. Let’s look at the following simple example.

Example. The binomial coefficients $c(n,k)$ is defined by the recurrences $f(n,0) = 1$ for $n \geq 0$, $f(0,n) = 0$ for $n \geq 1$ and $f(n,k) = f(n-1,k) + f(n-1,k-1)$ for $n,k \geq 1$. Find the OGF of $f(n,k)$.

We define the OGF $F(x,y) = \sum_{n\geq 0}\sum_{k\geq 0} f(n,k)x^ny^k$. As usual, we try to relate F with itself using the recurrence. We have

$$\sum_{n\geq 1}\sum_{k\geq 1} f(n,k)x^ny^k = x\sum_{n\geq 1}\sum_{k\geq 1} f(n-1,k)x^{n-1}y^k + xy\sum_{n\geq 1}\sum_{k\geq 1} f(n-1,k-1)x^{n-1}y^{k-1}.$$

Hence, we have

$$F(x,y) - \sum_{n\geq 0} x^n = x(F(x,y) - \sum_{n\geq 0} x^n) + xyF(x,y)$$

$$\Rightarrow F(x,y) - \frac{1}{1-x} = (x+xy)F(x,y) - \frac{x}{1-x}$$

$$\Rightarrow (1-x-xy)F(x,y) = 1$$

$$\Rightarrow F(x,y) = \frac{1}{1-x-xy}$$

From the bivariate OGF, we can deduce some interesting identities in one-variable. For example, we have

$$F(x,y) = \frac{1}{1-x-xy} = \frac{1}{1-x(y+1)} = \sum_{k\geq 0}(y+1)^kx^k$$

Hence, $[x^n]F(x,y) = (y+1)^n$. However, $[x^n]F(x,y) = \sum_{k=0}^{\infty} f(n,k)y^k$, so $\sum_{k=0}^{\infty} f(n,k)y^k = (y+1)^n$. Note that this gives the same result as binomial theorem on $(y+1)^n = \binom{n}{0}y^0 + \binom{n}{1}y^1 + \ldots + \binom{n}{n}y^n$.

It is more interesting to look at $[y^k]F(x,y)$ in terms of an OGF in x. We have

$$F(x,y) = \frac{1}{(1-x)-xy} = \frac{\frac{1}{1-x}}{1-\frac{x}{1-x}y} = \frac{1}{1-x}(1 + \frac{x}{1-x}y + (\frac{x}{1-x})^2y^2 + \ldots)$$

Comparing coefficients, we obtain $[y^k]F(x,y) = \frac{x^k}{(1-x)^{k+1}}$, so using the same argument as before, we have

$$\sum_{n=0}^{\infty} f(n,k)x^n = \frac{x^k}{(1-x)^{k+1}}.$$

This identity is interesting because it allows us to “expand” $\frac{1}{(1-x)^{k+1}}$ in terms of the OGF of binomial coefficients! In particular, we have $[x^{n-k}]\frac{1}{(1-x)^{k+1}} = \binom{n}{k}$, so $[x^n]\frac{1}{(1-x)^k} = \binom{n+k-1}{k-1}$. This identity is very useful especially when dealing with sums involving binomial coefficients where k is fixed and n varies.

Exponential Generating Function

So far we have looked at ordinary generating functions. Now, let me introduce a new type of generating functions called the **exponential generating function**.

Definition: Let $a_0, a_1, a_2, \ldots$ be a sequence of numbers. Then, the EGF of a (say $A(x)$ is defined as $\sum_{i=0}^{\infty} \frac{a_i}{i!} x^i$.

In other words, the EGF is just the OGF but every term a_i is now divided by $i!$. Why the weird choice of division by $i!$? The next example will shed some light on this choice.

Example. Let b_n denote the n-th Bell number, which counts the number of ways to partition $\{1, 2, \ldots, n\}$ into disjoint sets. For example, $b_3 = 5$, because there are 5 ways to partition $[3]$ into sets: 123; 12, 3; 13, 2; 1, 23; 1, 2, 3. Find the EGF of b_n.

Our first step is to look for a recurrence relation. Suppose you have this as a Div. 2 C problem. What is the simplest dp you can come up with?

We can fix the size of the set containing the element 1, say i. Then, there are $\binom{n-1}{i-1}$ ways to choose the other $i-1$ elements of the set, and b_{n-i} ways to partition the remaining $n-i$ elements. Hence, we obtain the simple recurrence formula

$b_n = \sum_{i=1}^{n} \binom{n-1}{i-1} b_{n-i} = \sum_{i=0}^{n-1} \binom{n-1}{i} b_i$ for $n \geq 1$.

By precomputing binomial coefficients, this is an $O(n^2)$ dp, which should be sufficient for a Div. 2 C. However, why stop here? Suppose the problem asks you to find b_n for $n \leq 3 \cdot 10^5$. Can you still solve this?

The answer is yes. Let's use our recurrence to find the EGF of b_n. Note that

$$b_n = \sum_{i=0}^{n-1} \binom{n-1}{i} b_i = \sum_{i=0}^{n-1} \frac{(n-1)!}{i!(n-1-i)!} b_i$$

$$\Rightarrow n\tfrac{b_n}{n!} = \sum_{i=0}^{n-1} \frac{b_i}{i!} \frac{1}{(n-1-i)!}$$

$$\Rightarrow \sum_{n\geq 1} n \frac{b_n}{n!} x^n = \sum_{n \geq 1} x \sum_{i=0}^{n-1} \frac{b_i x^i}{i!} \frac{x^{n-1-i}}{(n-1-i)!}$$

Now we see why EGFs are convenient for us. If our convolutions involve binomial coefficients (which is often the case when we deal with combinatorial objects), then multiplying EGFs kind of "automatically" helps us multiply our terms by a suitable binomial coefficient (more details later).

Back to our problem, we want to write everything in terms of $B(x)$. RHS is easy, since it is just $xB(x)e^x$ (Recall that $\sum_{n\geq 0} \frac{x^n}{n!}$ is the Maclaurin series of e^x). However, the LHS needs a bit of work, since we have the unfortunate nb_n term instead of the b_n term. To deal with this obstacle, we use a common trick when dealing with formal power series. Let us **differentiate** B(x), then multiply by x. Verify that if $A(x)$ is a formal power series with $A(x) = a_0 + a_1x^1 + a_2x^2 + \ldots = \sum_{n\geq 0} a_n x^n$ then

$xA'(x) = a_1x^1 + 2a_2x^2 + 3a_3x^3 + \ldots = \sum_{n \geq 0} n a_n x^n$.

Thus, looking back at our equation, we have

$xB'(x) = xB(x)e^x$, 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, $C(x) = \frac{A(x)-(a_0+a_1x+a_2x^2+\ldots+a_{k-1}x^{k-1})}{x^k}$ 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 $c_n = \displaystyle\sum_{k=0}^{n} a_k b_{n-k}$.

For EGF, $C(x) = A(x)B(x)$ generates the sequence $c_n = \displaystyle\sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k}$ (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 $c_n = \displaystyle\sum_{i_1+i_2+\ldots+i_k=n} a_{i_1}a_{i_2}\ldots a_{i_k}$

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

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 $c_n = \displaystyle\sum_{i=0}^{n} a_i$.

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, $\binom{r}{n} = \frac{r(r-1)\dots(r-(n-1))}{n!}$.

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

Next, we show a special case of the theorem.

Theorem. Let r be a real number and n be a nonnegative integer, then

$$(1+x)^r = \sum_{n\geq 0}\binom{r}{n}x^n.$$

The proof is just differentiating the left side n times and compare the constant term. I leave this as an exercise.

In particular, our mysterious function $\sqrt{1-4x} = (1-4x)^{\frac{1}{2}} = \sum_{n\geq 0}\binom{\frac{1}{2}}{n}(-4x)^n$

$$= \sum_{n\geq 0}\frac{1}{2}\cdot\frac{-1}{2}\cdot\frac{-3}{2}\cdot\ldots\cdot\frac{-(2n-3)}{2}\cdot\frac{1}{n!}\cdot(-4)^n x^n$$

$$= 1 + \sum_{n\geq 1}\frac{(-1)^{n-1}(1\cdot 3\cdot\ldots\cdot(2n-3))}{2^n}\cdot\frac{(-4)^n}{n!}x^n$$

$$= 1 + \sum_{n\geq 1} -2^n\cdot\frac{(2n-2)!}{2^{n-1}(n-1)!}\cdot\frac{1}{n!}x^n$$

$$= 1 + \sum_{n\geq 1}\frac{-2\cdot(2n-2)!}{(n-1)!n!}x^n$$

$$= 1 + \sum_{n\geq 1} -\frac{2}{n}\cdot\binom{2n-2}{n-1}x^n.$$

Hence, $C(x) = \frac{1-\sqrt{1-4x}}{2x} = \frac{1}{2x}\left[1 - 1 - \sum_{n\geq 1} -\frac{2}{n}\cdot\binom{2n-2}{n-1}x^n\right]$

$$= \sum_{n\geq 1}\frac{1}{n}\cdot\binom{2n-2}{n-1}x^{n-1}$$

$= \sum_{n\geq 0}\frac{1}{n+1}\binom{2n}{n}x^n$, as desired.

Some problems involving permutations

For a permutation $p = (p_1, p_2, \ldots, p_n)$, consider the graph formed by the edges $i \rightarrow p_i$. It is well-known that the graph is a union of several disjoint cycles.

Problem. Count the number of permutations of length n with k cycles.

These numbers are also called Stirling numbers of the first kind.

Let $c_n = (n-1)!$ be the number of permutations of length n which is a cycle. Let $C(x) = \sum_{n\geq 0}\frac{c_n}{n!}x^n$ denote the EGF of c. Let f_n be our answer and $F(x)$ be its EGF. The key observation here is that $F(x) = \frac{1}{k!}C(x)^k$.

Suppose for a moment our cycles are labelled from 1 to k. For every permutation, label each element with the label of the cycle it is in. Let's fix the length of cycle i to be a_i (so $\sum a_i = n$). Then, there are c_{a_i} ways to permute the elements in the i-th cycle and $\frac{n!}{a_1!a_2!\ldots a_k!}$ ways to assign cycle labels to the elements of the permutation. Finally, in our actual problem, the order of cycles doesn't matter, so we need to divide by $k!$ in the end.

To summarize, the answer is $\frac{n!}{k!}\displaystyle\sum_{a_1+a_2+\ldots+a_k=n}\frac{c_{a_1}c_{a_2}\ldots c_{a_k}}{a_1!a_2!\ldots a_k!}$. Verify that $\displaystyle\sum_{a_1+a_2+\ldots+a_k=n}\frac{c_{a_1}c_{a_2}\ldots c_{a_k}}{a_1!a_2!\ldots a_k!}$ is $[x^n]C(x)^k$, so $F(x) = \frac{1}{k!}C(x)^k$ (the $n!$ disappears into $F(x)$ because we are dealing with EGFs).

Let's assume this is a CP problem and we are asked to find the answer for (n, k). Then, we can calculate the answer directly using generating functions in $O(n\log n)$ since we can do exponentiation in $O(n\log n)$ ($P(x)^k = \exp(k\ln(P(x)))$).

Actually, what we just did is a special case of the more general Exponential Formula. However, I feel that it is easier to understand the Exponential Formula from these specific examples and you should try to understand it intuitively until it becomes common sense.

Problem. Count the number of permutations of length n such that all cycle lengths are in a fixed set of positive integers S.

We use the same trick as the previous problem, but let $c_i = 0$ if i is not in S.

This time, we need to find $[x^n]\displaystyle\sum_{k\geq 0}\frac{1}{k!}C(x)^k = [x^n]\exp(C(x))$ because we need to sum over all values of k (number of cycles), which can also be computed easily.

Problem. Find the expected number of cycles of a permutation of length n.

To compute the expected number of cycles, we count the sum of number of cycles over all permutations of length n. Let g_n denote the sum of number of cycles over all permutations of length n and $G(x)$ as the EGF of g. Using the same function C in the previous problems, we need to find (note the extra factor k which is the difference between this and the previous examples)

$$[x^n]G(x) = [x^n]\sum_{k\geq 0}\frac{k}{k!}C(x)^k = [x^n]C(x)\sum_{k\geq 1}\frac{1}{(k-1)!}C(x)^{k-1} = [x^n]C(x)\exp(C(x))$$

However, $C(x) = \displaystyle\sum_{k\geq 1}\frac{(k-1)!}{k!}x^k = \sum_{k\geq 1}\frac{x^k}{k} = -\ln(1-x)$. Hence, $C(x)\exp(C(x)) = -\frac{\ln(1-x)}{(1-x)}$.

For $n \geq 1$, $[x^n](-\ln(1-x)) = \frac{1}{n}$. By the Prefix Sum trick, $[x^n]\frac{-\ln(1-x)}{1-x} = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$.

Thus, $[x^n]G(x) = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$. Since $\frac{g_n}{n!}$ is the expected number of cycles of a permutation of length n, our answer is $1 + \frac{1}{2} + \ldots + \frac{1}{n}$, the n-th Harmonic number!

We see that the exponential trick is viable when we are dealing with items that are made up from smaller pieces.

Stirling Numbers of the Second Kind

Problem. Find the number of ways to partition the set $\{1, 2, \ldots, n\}$ into k subsets.

These numbers are also called Stirling numbers of the second kind.

Denote the answer by $f(n, k)$. The trick is to consider the polynomial (also known as **deck enumerator**) $D(x) = \displaystyle\sum_{n\geq 1}\frac{x^n}{n!}$. What is $D(x)^k$? We have $[x^n]D(x)^k = \displaystyle\sum_{a_1+a_2+\ldots+a_k=n, a_i\geq 1}\frac{1}{a_1!a_2!\ldots a_k!}$. This sum has a similar combinatorial interpretation as the ones in the previous problems. Let's assume the partition sets are labelled from 1 to k. Then, a_i denotes the size of the i-th set and there are $\frac{n!}{a_1!a_2!\ldots a_k!}$ ways to assign a set to each element (by the multinomial theorem). However, we have counted each partition $k!$ times, since in our final answer the sets shouldn't be ordered. Thus, $k!f(n, k) = n![x^n]D(x)^k$.

Rearranging gives us $\frac{f(n,k)}{n!} = \frac{[x^n]D(x)^k}{k!}$. Hence, $\displaystyle\sum_{n\geq 0}\frac{f(n,k)}{n!}x^n = \frac{D(x)^k}{k!}$. Introducing the variable y to correspond to the variable k, we have $\displaystyle\sum_{k\geq 0}\sum_{n\geq 0}\frac{f(n,k)}{n!}x^n y^k = \sum_{k\geq 0}\frac{[D(x)y]^k}{k!} = \exp(D(x)y)$.

Note: The polynomial $H(x, y) = \displaystyle\sum_{k\geq 0}\sum_{n\geq 0}f(n,k)\frac{x^n}{n!}y^k$ is also known as a **hand enumerator**.

Thus, we have the simple formula $H(x, y) = \exp(D(x)y)$. Note that $D(x) = \displaystyle\sum_{n\geq 1}\frac{x^n}{n!} = e^x - 1$, so we have $H(x, y) = e^{(e^x-1)y}$ (note how similar this is to the EGF of Bell numbers! In fact $H(x, 1)$ is the EGF of Bell numbers (can you see why?)).

To get the answer, we just need to find $n![x^n y^k]H(x, y) = n![x^n]\frac{(e^x-1)^k}{k!}$ which you can compute using polynomial operations efficiently.

Graph Counting

Problem. Find the number of vertex-labeled undirected graphs with n vertices so that each vertex has degree 2.

Every such graph must be a union of disjoint cycles (why?). As usual, we start by considering the generating function for one "component" in the item we need to count. Let d_n denote the number of **undirected** cycles of length n and $D(x)$ denote its EGF. Then, $D(x) = \sum_{n\geq 3} \frac{(n-1)!}{2n!}x^n = \frac{1}{2}\sum_{n\geq 3}\frac{x^n}{n} = \frac{1}{2}\left(-\ln(1-x) - x - \frac{x^2}{2}\right)$.

Let $G(x)$ denote the EGF of our answer. Using the same argument as before, we find that $G(x) = \exp(D(x))$, so we get the formula $G(x) = \exp\left(\ln\left(\sqrt{\frac{1}{1-x}}\right) - \frac{x}{2} - \frac{x^2}{4}\right) = \frac{e^{-\frac{x}{2}-\frac{x^2}{4}}}{\sqrt{1-x}}$, and you can compute the coefficient of x^n to obtain the answer.

Let's look at a trickier example.

Problem. Find the number of bipartite vertex-labeled graphs with n vertices.

It is tempting to try a similar approach as the previous problem. We can relate the number of bipartite graphs with the number of connected bipartite graphs. Can we count the number of connected bipartite graphs easily? Unfortunately, it does not seem to be too easy to count.

Instead, let us color each vertex of the graph with red or blue, and count the number of **colored** bipartite graphs (not necessarily connected). Suppose we choose k vertices to color red and $n-k$ to color blue. Then, there are $\binom{n}{k}$ ways to choose the coloring and $2^{k(n-k)}$ to choose the edges (since each edge must be between 2 components). Thus, the number of colored bipartite graphs on n vertices is $\sum_{k\geq 0}\binom{n}{k}2^{k(n-k)}$. Call this number a_n and its EGF as $A(x)$.

The next step is to relate the number of colored bipartite graphs with the number of colored connected bipartite graphs. Let b_n denote the number of colored connected bipartite graphs on n vertices and $B(x)$ be its EGF. Using a similar argument as before, we have the relation $A(x) = \exp(B(x))$, and thus $B(x) = \ln(A(x))$.

Returning to our original problem, our next step is to count the number of connected bipartite graphs on n vertices (call the count c_n and EGF $C(x)$). However this is easy, since each connected bipartite graph can be colored in exactly two ways (the coloring is fixed once we choose the color of a vertex). Hence, $C(x) = \frac{B(x)}{2}$.

Finally, let d_n be the number of bipartite graphs on n vertices and $D(x)$ be its EGF. Then, we have $D(x) = \exp(C(x))$ using the exponential argument. Thus, $D(x) = \exp(C(x)) = \exp\left(\frac{B(x)}{2}\right) = \exp\left(\frac{\ln(A(x))}{2}\right) = \sqrt{A(x)}$, which is a nice formula!

Placing Rooks and PIE

Let's look at a last example which demonstrates the use of the Inclusion-Exclusion Principle (PIE).

Consider a $n\times n$ chessboard where some cells are colored black and others are colored white. Suppose we magically know the sequence r_k, the number of ways to place k non-attacking rooks on white cells of the chessboard (i.e. no two are on the same row or column, no rooks are on black cells). Let e_k denote the number of ways to place n non-attacking rooks on the chessboard so that exactly k of the rooks are on white squares. Can we find e_k in terms of r_k?

The trick is that exact conditions are usually harder to count while "at least" conditions are easier to count. For a fixed subset of white cells S, denote $N(S)$ as the number of ways to place n non-attacking rooks on the chessboard such that there is at least one rook on each cell in S. Let $n_k = \sum_{|S|=k} N(S)$.

We relate n_k with e_k. Consider a subset T of size t and a way to place n non-attacking rooks so that the white cells they occupy is exactly T. Every k-element subset of T contributes to the sum n_k. Thus, we obtain the recurrence $n_k = \sum_{t\geq 0}\binom{t}{k}e_t$.

Let $N(x)$ and $E(x)$ be the OGFs of n_k and e_k. We can derive a simple relation between $N(x)$ and $E(x)$. Indeed, we have

$N(x) = \sum_{k\geq 0} x^k \sum_{t\geq 0}\binom{t}{k}e_t = \sum_{t\geq 0} e_t \sum_{k\geq 0}\binom{t}{k}x^k = \sum_{t\geq 0} e_t(x+1)^t = E(x+1)$. Thus, we have the simple relation $E(x) = N(x-1)$.

It turns out that n_k is usually much easier to find. In our problem, $n_k = r_k(n-k)!$, since we can first choose our set S as any set of k non-attacking rooks on white cells, then place the other $n-k$ rooks in $(n-k)!$ ways. Thus, we obtain $N(x) = \sum_{k\geq 0} r_k(n-k)!x^k$ and $E(x) = \sum_{k\geq 0} r_k(n-k)!(x-1)^k$. Hence, we can read e_j from the coefficients of $E(x)$.

Proving Some Interesting Theorems via Generating Functions

This is not entirely CP related but here are some cool theorems you can prove easily with generating functions.

Partition in odd parts = Partition in distinct parts

A partition of n into k parts is a multiset of positive integers of size k which sum up to n. For example, $\{3,1,1\}$ is a partition of 5 into 3 parts. Note that the order of elements do not matter, so $\{3,1,1\}$ and $\{1,3,1\}$ are the same partition.

You might have heard of the well-known problem of proving that the number of partitions of n into odd parts is equal to the number of partitions of n into distinct parts. Here, we prove a generalization of it.

Problem. Prove that the number of partitions of n into parts of size not divisible by $k+1$ is equal to the number of partitions of n into parts such that there are at most k parts of each size.

Note that when $k=1$ we reduce this to the standard problem.

Fix k and let $A(x)$ be the OGF of the first object and $B(x)$ be the OGF of the second object. Observe that choosing a partition is the same as choosing the number of times we use each integer in our multiset, so

$$B(x)=\prod_{r\geq 1}(1+x^r+x^{2r}+\ldots+x^{kr})$$
$$=\prod_{r\geq 1}\left(\frac{1-x^{r(k+1)}}{1-x^r}\right)$$
$$=\prod_{r\geq 1,(k+1)\nmid r}\left(\frac{1}{1-x^r}\right)$$
$$=\prod_{r\geq 1,(k+1)\nmid r}(1+x^r+x^{2r}+\ldots)=A(x)$$

Binet's Formula (and solving Linear Recurrences)

Let f_n denote the n-th Fibonacci number (with $f_0=0$, $f_1=1$, $f_n=f_{n-1}+f_{n-2}$). You may have heard of Binet's Formula, which states that $f_n=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right]$.

This might look very random, but it actually comes directly from the generating function. Recall that $F(x)=\frac{x}{1-x-x^2}$ is the OGF of f. The trick here is to use **partial fractions** (we will explore another example in the next part). Let $-\gamma_1,-\gamma_2$ be the roots of the equation $1-x-x^2=0$ (where $\gamma_1=\frac{1+\sqrt{5}}{2}$, $\gamma_2=\frac{1-\sqrt{5}}{2}$). Then, we can write $F(x)=\frac{A}{x+\gamma_1}+\frac{B}{x+\gamma_2}$. With some calculation, we can obtain $F(x)=\frac{1}{\gamma_1-\gamma_2}\left(\frac{1}{1-\gamma_1 x}-\frac{1}{1-\gamma_2 x}\right)=\frac{1}{\sqrt{5}}\left(\sum_{j\geq 0}\gamma_1^j x^j-\sum_{j\geq 0}\gamma_2^j x^j\right)$.

Comparing coefficients, we get $f_n=\frac{1}{\sqrt{5}}(\gamma_1^n-\gamma_2^n)$. Recalling that $\gamma_1=\frac{1+\sqrt{5}}{2}$ and $\gamma_2=\frac{1-\sqrt{5}}{2}$ gives us Binet's Formula.

Note that this method is generalizable for general linear recurrences.

Probability that a random permutation has no cycle length which is the square of an integer

Problem. What is the probability that a random permutation has no cycle length which is the square of an integer?

This problem seems pretty random (no pun intended), but I include it here to show the power of generating functions.

Firstly, suppose we know that our permutation is of length n and there are a_i cycles of length i (so $\sum_{i\geq 1} ia_i=n$). How many such permutations are there? With some simple counting, we can obtain the formula $\frac{n!\prod_{i\geq 1}((i-1)!)^{a_i}}{\prod_{i\geq 1}(i!)^{a_i}\cdot\prod_{i\geq 1}(a_i!)}=\frac{n!}{\prod_{i\geq 1}i^{a_i}\cdot\prod_{i\geq 1}(a_i!)}$ (Hint: Assume the cycles are labelled first, and assign the elements into cycles, then arrange the elements within each cycle. Divide some factorials to handle overcounts).

The sequence $a=(a_1,a_2,\ldots)$ defined above is also called the **cycle type** of a permutation.

Let $c(a)$ denote the number of permutations of length $n=a_1+2a_2+\ldots$ with cycle type a and $p(a)$ denote the probability that a permutation of length $n=a_1+2a_2+\ldots$ has cycle type a. Hence, $c(a)=\frac{n!}{\prod_{i\geq 1}i^{a_i}\cdot\prod_{i\geq 1}(a_i!)}$ and $p(a)=\frac{c(a)}{n!}$

Now, the trick is to consider the **infinite-variable** generating function in $x_1,x_2,\ldots$:

$$C(x,y)=\sum_{n\geq 0}\frac{y^n}{n!}\sum_{a_1+2a_2+\ldots=n,a_i\geq 0}c(a)x_1^{a_1}x_2^{a_2}\ldots$$

From our discussion above, we know how to find $c(a)$, thus we can write $C(x, y)$ as

$$C(x,y) = \left(\sum_{a_1 \geq 0} \frac{(yx_1)^{a_1}}{a_1! 1^{a_1}}\right)\left(\sum_{a_2 \geq 0} \frac{(y^2 x_2)^{a_2}}{a_2! 2^{a_2}}\right) \ldots = \exp(yx_1)\exp\left(y^2 \frac{x_2}{2}\right)\ldots = \exp\left(\sum_{i \geq 1} \frac{y^i x_i}{i}\right).$$

Hence, for a fixed cycle type $a = (a_1, a_2, \ldots)$, $p(a) = [x_1^{a_1} x_2^{a_2} \ldots] \exp\left(\sum_{i \geq 1} \frac{y^i x_i}{i}\right)$.

Let us return to our problem. Call a cycle type a good if $a_j = 0$ for all perfect squares j. We want to find $\lim_{n \to \infty} \sum_{a \text{ good}} [y^n x_1^{a_1} x_2^{a_2} \ldots] \exp\left(\sum_{i \geq 1} \frac{y^i x_i}{i}\right)$. We can "substitute" $x_j = 1$ for all non-perfect squares j to indicate that we don't care about the power of x_j if j is not a perfect square. so we reduce our problem to finding (noting that $a_j = 0$ for all perfect squares j)

$$\lim_{n \to \infty} [y^n] \exp\left(\sum_{i = z^2} \frac{y^i x_i}{i} + \sum_{i \neq z^2} \frac{y^i}{i}\right)$$

$$= \lim_{n \to \infty} [y^n] \exp\left(\sum_{i = z^2} \frac{y^i (x_i - 1)}{i} + \sum_{i \geq 1} \frac{y^i}{i}\right)$$

$$= \lim_{n \to \infty} [y^n] \exp\left(\sum_{i = z^2} \frac{y^i (x_i - 1)}{i} - \ln(1 - y)\right)$$

$$= \lim_{n \to \infty} [y^n] \frac{1}{1 - y} \exp\left(\sum_{i = z^2} \frac{y^i (x_i - 1)}{i}\right)$$

If we let $A(y)$ be the OGF of $a_n = [y^n] \exp\left(\sum_{i = z^2} \frac{y^i (x_i - 1)}{i}\right)$, then by the Prefix Sum trick, our limit is equal to $\lim_{n \to \infty} \sum_{i \geq 0} a_i$ (assuming the sum converges).

Intuitively, we can get the "sum to infinity" by substituting $y = 1$ into $\exp\left(\sum_{i = z^2} \frac{y^i (x_i - 1)}{i}\right)$, and we are only interested in the terms without x_i (for i a perfect square), so we let these $x_i = 0$, to obtain

$\lim_{n \to \infty} [y^n] \frac{1}{1 - y} \exp\left(\sum_{i = z^2} \frac{y^i (x_i - 1)}{i}\right) = \exp\left(\sum_{i = z^2} -\frac{1}{i}\right) = e^{-\frac{\pi^2}{6}}$, which is actually our answer (recall that $\sum_{i \geq 1} \frac{1}{i^2} = \frac{\pi^2}{6}$).

Snake Oil Trick in Proving (or Finding) Combinatorial Identities

To end the first part of this tutorial, I will briefly introduce a trick to simplify combinatorial identities using generating functions. The idea is that instead of dealing with the sum directly, it is easier to deal with the series obtained from the generating functions.

Problem. Find the sum $\sum_{k \geq 0} \binom{k}{n - k}$ for a fixed positive integer n.

Suppose the answer to our problem is $f(n)$. The idea is that it is easier to consider the OGF of f, which is $F(x) = \sum_{n \geq 0} f(n) x^n$. We have

$$F(x) = \sum_{n \geq 0} f(n) x^n$$

$$= \sum_{n \geq 0} \sum_{k \geq 0} \binom{k}{n - k} x^n$$

Now, we switch summation signs to obtain

$$= \sum_{k \geq 0} \sum_{n \geq 0} \binom{k}{n - k} x^n$$

The key idea is to make the inner sum easy to compute, and we know how to compute $\sum_{n \geq 0} \binom{k}{n - k} x^{n-k}$, since it is just $\sum_{r \geq 0} \binom{k}{r} x^r$ in disguise with $r = n - k$!

Thus, we factor out x^k to obtain

$$= \sum_{k \geq 0} x^k \sum_{n \geq 0} \binom{k}{n - k} x^{n-k}$$

$$= \sum_{k\geq 0} x^k(1+x)^k$$
$$= \sum_{k\geq 0}(x(1+x))^k$$
$$= \frac{1}{1-x-x^2}$$

Do you recognize the last expression? It is actually $\frac{1}{x}F(x)$ where $F(x)$ is the OGF of the Fibonacci numbers! Thus, $\frac{1}{1-x-x^2} = \sum_{n\geq 0} f_{n+1}x^n$, and by comparing coefficients we get $f(n) = f_{n+1}$, the $(n+1)$-th Fibonacci number!

There are many other similar applications of the Snake Oil Method but I won't go into detail here. In general, this method might be useful in CP if you encounter some math problems and reduce the problem into double or triple summation of binomial coefficients but you need an $O(n)$ solution. Sometimes, you can forcefully simplify your summations using the Snake Oil method. We will use the trick of introducing a power series and swapping summation signs again to simplify some expressions in the next part of this article.

As an exercise, try to prove the following identity with the Snake Oil method:

$\sum_{r=0}^{k}\binom{m}{r}\binom{n}{k-r} = \binom{n+m}{k}$ (there is an obvious bijective proof, can you see why this is true?). This identity is very useful in simplifying sums involving binomial coefficients.

This ends the first part of my tutorial on generating functions. The next part will focus more on applications on GFs in CP problems, so stay tuned!

UPD: Part 2 is now available here.

P.S. Let me know if I made any errors or typos in the post (which is likely to happen).

Welcome to Part 2 of my tutorial on generating functions. The first part focused on introducing generating functions to those without any background in generating functions. In this post, I will demonstrate a few applications of generating functions in CP problems. Let us start with some relatively straightforward examples.

Note: Unless stated otherwise, all computations are done modulo a convenient prime (usually 998244353). Also, $[n]$ denotes the set $\{1, 2, \ldots, n\}$.

Blatant Applications in Counting Problems

Problem. AGC 005 Problem F You have a tree T with n vertices. For a subset S of vertices, let $f(S)$ denote the minimum number of vertices in a subtree of T which contains all vertices in S. For all $1 \leq k \leq n$, find the sum of $f(S)$ over all subsets S with $|S| = k$.

Constraints: $n \leq 2 \cdot 10^5$.

Solution

First, we need to do some basic counting. For a set of vertices S, let $t(S)$ denote the minimal subtree that contains all vertices of S.

Fix k. It is clear that we have to somehow double count the sum of $f(S)$. If we look at a vertex v, it is not that easy to count the number of sets S of size k such that $t(S)$ contains v. However, if we look at an edge e, then it is easy to see that $t(S)$ contains e if and only if all elements of S is in the same connected component formed by deleting the edge e from the tree. In other words, if the edge e splits the tree into two components of size (number of vertices) a and $n-a$ respectively, then the number of S with $|S| = k$ and $e \in t(S)$ is exactly $\binom{n}{k} - \binom{a}{k} - \binom{n-a}{k}$. Thus, the sum of $f(S)$ over all subsets S is just $\binom{n}{k}$ (since number of vertices = number of edges + 1) plus the sum of $\binom{n}{k} - \binom{a}{k} - \binom{n-a}{k}$ over all edges e.

We can find the value of a for each edge e by simple dfs. Suppose we have computed the value of a for each edge e, say $a_1, a_2, \ldots, a_{n-1}$. If we can compute $\sum_{i=1}^{n-1}\binom{a_i}{k}$ fast for each $1 \leq k \leq n$, then we are done, so we will focus on this task.

Let c_i denote the number of times i appear in the array $a_1, a_2, \ldots, a_{n-1}$. Hence, we need to find $ans_k = \sum_{i=0}^{n} c_i\binom{i}{k}$.

It is almost customary to try and write binomial coefficients in terms of factorials and look for simplifications.

$$ans_k = \sum_{i=0}^{n} c_i\binom{i}{k} = \sum_{i=0}^{n} c_i\frac{i!}{k!(i-k)!} = \frac{1}{k!}\sum_{i=0}^{n} c_i(i!) \cdot \frac{1}{(i-k)!}$$

Obviously we only need to know how to compute the sum $\sum_{i=0}^{n} c_i(i!) \cdot \frac{1}{(i-k)!}$ fast for all k. If you have studied about generating functions, you see that our sum looks very much like the convolution of two sequences. If we can write our summand in terms of $f(k)g(i-k)$, then we can define $F(x) = \sum_{i\ge 0} f(i)x^i$ and $G(x) = \sum_{i\ge 0} g(i)x^i$ and read off $\sum_{i\ge 0} f(i)g(k-i)$ from the coefficients of $F(x)G(x)$.

Clearly, we can set $f(i) = c_i(i!)$. We want to set $g(i) = \frac{1}{(-i)!}$. However, you might worry that $(-i)!$ is not defined for Error: Misplaced &. It's ok, since we can always shift our sequence can set $g(M+i) = \frac{1}{(-i)!}$ instead for some large integer M. Then, we have $g(i) = \frac{1}{(M-i)!}$, which we can now safely compute. Instead of reading off the coefficient of x^k from $F(x)G(x)$, we read off the coefficient of x^{k+M}.

Convolutions of this type appear in countless FFT problems and usually the hard part is to reduce your sums into a form that can be seen as the convolution of two polynomials. However, generating functions are much more powerful than this, as we shall see the next examples.

Problem. The Child and Binary Tree You have a set of positive integers $C = \{c_1, c_2, \dots, c_n\}$. A vertex-weighted rooted binary tree is called good if all the vertex weights are in C. The weight of such a tree is the sum of weights of all vertices. Count the number of distinct good vertex-weighted rooted binary trees with weight s for all $1 \le s \le m$. Note that the left child is distinguished from the right child in this problem (see the samples for more info).

Constraints: $n, m \le 10^5$

Solution

This problem was created 6 years ago and has a 3100 rating on CF, but if you know about generating functions in 2020 (in the era where polynomial operations template is available) it is almost trivial.

Firstly, the obvious step is to let f_s denote the answer for s and $F(x)$ as the OGF of f. Let us derive a recurrence for f_s.

Clearly, $f_0 = 0$. For Error: Misplaced &, we iterate over all possible weights for the root, then iterate over all possible weights of the left subtree, giving the recurrence $f_s = \sum_{c\in C}\sum_{i\ge 0} f_i f_{s-c-i}$ (for convenience, $f_i = 0$ for Error: Misplaced &).

Speaking in generating functions, this is just the equation

$$\sum_{n\ge 1} f_n x^n = \sum_{n\ge 1}\sum_{c\in C} x^c \sum_{i\ge 0} f_i x^i f_{s-c-i} x^{s-c-i}.$$

This motivates us to define $F(x) = \sum_{n\ge 0} f_n x^n$ and $C(x) = \sum_{c\in C} x^c$. Thus, $F(x) - 1 = C(x)F(x)^2$ (you should be able to deduce this directly from the equation above).

Our goal is to find $F(x)$, so we solve for $F(x)$ using the quadratic formula (remember what we did to obtain the OGF of Catalan numbers?) to obtain $F(x) = \frac{1\pm\sqrt{1-4C(x)}}{2C(x)}$.

Similar to the case with Catalan numbers, we choose the negative sign since otherwise as $x \to 0$, the numerator goes to 2 while the denominator goes to 0, but we should converge to a finite limit since $F(0) = 1$. Thus, $F(x) = \frac{1-\sqrt{1-4C(x)}}{2C(x)}$.

In the language of generating functions, we are pretty much done, but there is one minor implementation detail here. $C(x)$ has constant term 0, and thus we cannot take the reciprocal directly. However, we just need to "rationalize" the numerator and rewrite our function as

$F(x) = \frac{1-\sqrt{1-4C(x)}}{2C(x)} = \frac{(1-\sqrt{1-4C(x)})(1+\sqrt{1-4C(x)})}{2C(x)(1+\sqrt{1-4C(x)})} = \frac{4C(x)}{2C(x)(1+\sqrt{1-4C(x)})} = \frac{2}{1+\sqrt{1-4C(x)}}$. You can verify that the constant term of the denominator is nonzero, so we can calculate the first m terms of the series with usual polynomial operations and solve the problem in $O(m\log m)$.

To be honest, it is unfair to say that the problem is easy because the main difficulty was to compute the square root and reciprocal of a power series. However, in the present times, these algorithms can be easily found online and in online rounds you can just use templates to perform these operations and thus I would consider this problem to be straightforward.

Problem. Descents of Permutations (from Enumerative Combinatorics 1) Call a permutation $p_1, p_2, \dots, p_n$ of $[n]$ k-good if Error: Misplaced & iff $k \mid i$. Find the number of k-good permutations of length n.

Constraints: $n, k \le 5\cdot 10^5, n \ge 3, k \ge 2$

Solution

There is a simple $O\left(\frac{n^2}{k}\right)$ dp solution (try to find it), but that won't be sufficient for this problem. We need to use generating functions. Let $f(n)$ denote the answer (k is fixed).

For a permutation $p = p_1, p_2, \ldots, p_n$, define the descent set as the set of integers $1 \le i \le n-1$ such that Error: Misplaced &. We are looking for the number of length n permutations with descent set equal to $S = \{k, 2k, 3k, 4k, \ldots\}$.

The idea is that counting "exact" conditions is hard, but counting "at least" conditions is easier. Let $f(S)$ denote the number of permutations with descent set exactly equal to S and $g(S)$ denote the number of permutations with descent set that is a subset of S. Clearly, we have $g(S) = \sum_{T \subseteq S} f(T)$. By the Principle of Inclusion-Exclusion, we have $f(S) = \sum_{T \subseteq S} (-1)^{|S|-|T|} g(T)$ (see here, this is a commonly used form of PIE).

$g(S)$ is much easier to count. Suppose $S = \{s_1, s_2, \ldots, s_m\}$ where Error: Misplaced & for all $1 \le i \le k-1$. This means that Error: Misplaced & must hold for all j that is not in S. A better way to visualize this is that we divide our permutation into several increasing blocks, the first block has size s_1, the second block has size $s_2 - s_1$ and so on. The last block has size $n - s_m$. The only restrictions is that the elements in each block must be in increasing order. Hence, it is clear that the probability that a random permutation satisfies this condition is just $\frac{1}{s_1!(s_2-s_1)!(s_3-s_2)!\ldots(n-s_m)!}$ (multiply the probability that each block is ordered correctly). Hence, $g(S) = \frac{n!}{s_1!(s_2-s_1)!(s_3-s_2)!\ldots(n-s_m)!}$.

Let's substitute this back to our equation. For simplicity, let $D = k, 2k, 3k, \ldots, \cap [n-1]$. Our problem reduces to finding $f(D)$.

Any subset $T = \{s_1, s_2, \ldots, s_{m-1}\}$ (elements sorted in increasing order) of D can be described by a sequence of positive integers $b_1, b_2, \ldots, b_m$ where $b_i = s_i - s_{i-1}$ ($s_0 = 0$, $s_m = n$ for simplicity) denote the gap between consecutive elements of T. For example, when $T = \{3, 9, 12\}$ and $n = 13$, we can describe it with the sequence $3, 6, 3, 1$. Note that k divides $b_1, b_2, \ldots, b_{m-1}$ and b_m has the same remainder as n mod k (call such sequences good).

This allows us to simplify the formula of $g(T)$ to $\frac{n!}{b_1!b_2!\ldots b_m!}$.

Hence,

$$f(D) = \sum_{T \subseteq D} (-1)^{|D|-|T|} g(T)$$

$$= \sum_{\sum b_i = n, b_i \text{ good}} (-1)^{|D|-(m-1)} \frac{n!}{b_1!b_2!\ldots b_m!}$$

For simplicity, let $n = kq + r$ where $1 \le r \le k$. Since we only care about the answer for $k \mid n - r$, we look at the EGF on these terms only, i.e. consider $F(x) = \sum_{q \ge 0} \frac{f(kq+r)}{q!} x^q$.

We have

$$F(x) = \sum_{q \ge 0} \frac{f(kq+r)}{(kq+r)!} x^{kq+r}$$

$$= \sum_{q \ge 0} \sum_{m \ge 1} \sum_{\sum b_i = kq+r, b_i \text{ good}} (-1)^{q-m+1} \frac{(kq+r)!}{b_1!b_2!\ldots b_m!} \cdot \frac{x^{kq+r}}{(kq+r)!}$$

$$= \sum_{m \ge 1} \sum_{q \ge 0} \sum_{\sum b_i = kq+r, b_i \text{ good}} (-1)^{q-m+1} \frac{x^{b_1} \cdot x^{b_2} \cdot \ldots \cdot x^{b_m}}{b_1!b_2!\ldots b_m!}$$

$$= \sum_{m \ge 1} \left(\sum_{i \ge 1} \frac{(-1)^{i-1} \cdot x^{ki}}{(ki)!} \right)^{m-1} \cdot \left(\sum_{i \ge 0} \frac{(-1)^i \cdot x^{ki+r}}{(ki+r)!} \right)$$

Take a moment to digest the last identity (try expanding the brackets). The idea is that we are able to make $b_1, b_2, \ldots, b_m$ independent of each other and simplify our sum into the convolution (or power) of several polynomials.

Continuing our simplifications, we have

$$= \left(\sum_{i \ge 0} \frac{(-1)^i \cdot x^{ki+r}}{(ki+r)!} \right) \cdot \sum_{m \ge 1} \left(\sum_{i \ge 1} \frac{(-1)^{i-1} \cdot x^{ki}}{(ki)!} \right)^{m-1}$$

$$= \left(\sum *i \ge 0 \frac{(-1)^i \cdot x^{ki+r}}{(ki+r)!} \right) \cdot \frac{1}{1 - \left(\sum *i \ge 1 \frac{(-1)^{i-1} \cdot x^{ki}}{(ki)!} \right)}$$

$$= \frac{\sum *i \ge 0 \frac{(-1)^i \cdot x^{ki+r}}{(ki+r)!}}{\sum *i \ge 0 \frac{(-1)^i \cdot x^{ki}}{(ki)!}}$$

We can compute the first n terms of the last expression in $O(n \log n)$ time, and we're done.

I think exponential generating functions are especially good at handling sums involving multinomial coefficients as you can separate the factorials into different polynomials and reduce it to convolution of EGFs.

Expected Value of Stopping Time

I think this is also a beautiful application of generating functions. The recent problem Slime and Biscuits can be solved using the trick I will demonstrate here (there is a short editorial using this method here). Let's look at a different example.

Problem. Switches There are n switches, each of which can be on or off initially. Every second, there is a probability of $\frac{p_i}{S}$ that you will flip the state of the i-th switch. The game ends when all switches are off. What is the expected number of seconds the game will last?

Constraints: $n \leq 100$, $\sum p_i = S$, Error: Misplaced &, $S \leq 50000$

Solution

It is hard to compute when a game ends, and it is also hard to compute the probability that the game ends in exactly k moves. However, it is relatively easier to compute the probability that the state of switches are all off after exactly k moves. Let $a(k)$ denote the probability that all switches are off after exactly k moves, and $A(x)$ be the EGF (we'll see why we choose EGF soon) of a. How to find $A(x)$?

Suppose we fix a sequence $a_1, a_2, \dots, a_n$ such that $\sum a_i = k$ where a_i denotes the number of flips of the i-th switch. The probability of achieving this sequence is $\frac{k!}{a_1!a_2!\dots a_n!}\left(\frac{p_1}{S}\right)^{a_1}\left(\frac{p_2}{S}\right)^{a_2}\dots\left(\frac{p_n}{S}\right)^{a_n}$ (the product of the number of sequences of switch flips such that switch i is flipped exactly a_i times and the probability the sequence of switch flips occur). Let $q_i = \frac{p_i}{S}$ for convenience. Hence, $\frac{a(k)}{k!} = \frac{q_1^{a_1}}{a_1!} \cdot \frac{q_2^{a_2}}{a_2!} \cdot \dots \cdot \frac{q_n^{a_n}}{a_n!}$.

Let's assume that all switches are off at the initial state for now. Then, the EGF is $A(x) = \prod_{i=1}^{n}\left(\frac{(q_i x)^0}{0!} + \frac{(q_i x)^2}{2!} + \dots\right)$, since we require each switch to be flipped an even number of times. In the general case, our EGF is similar, but some switches are required to be flipped an odd number of times instead. Motivated by this special case, we let $E_i(x) = \sum_{j \text{ even}} \frac{(q_i x)^j}{j!}$ and $O_i(x) = \sum_{j \text{ odd}} \frac{(q_i x)^j}{j!}$. Then, if $s_i = 1$ (the switch is initially on), we choose $O_i(x)$ as the i-th term of our product (in the formula for $A(x)$), while if $s_i = 0$, we choose $E_i(x)$ as the i-th term of our product.

There's an even more compact way to write this "observation". Recall that $\frac{e^x + e^{-x}}{2} = \cosh x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$. We can use a similar idea here. To express $E_i(x)$, we can try to add or subtract $\exp(q_i x)$ with $\exp(-q_i x)$ to "filter out" the even or odd power terms (we will see a generalization of this trick called the roots of unity filter later in the post). Verify that $E_i(x) = \frac{\exp(q_i x) + \exp(-q_i x)}{2}$ and $O_i(x) = \frac{\exp(q_i x) - \exp(-q_i x)}{2}$. Thus, we can even write this as $\frac{\exp(q_i x) + (-1)^{s_i}\exp(-q_i x)}{2}$.

To summarize, $A(x) = \prod_{i=1}^{n} \frac{[\exp(q_i x) + (-1)^{s_i}\exp(-q_i x)]}{2}$.

Ok, so we can find $a(k)$. Let $c(k)$ denote the probability that the game ends (all switches are off for the **first time**) after exactly k moves. How can we relate $c(k)$ and $a(k)$? Here is the trick. Consider any sequence of k moves resulting in all switches being off. After i moves (possibly $i = k$), the switches are all off for the first time and the game ends. For the remaining $k - i$ moves, we need to flip each switch an even number of times. Thus, if we let $b(k)$ denote the probability that we flip each switch an even number of times after exactly k moves, then $a(k) = \sum_{i=0}^{k} c(i)b(k-i)$. Does this look familiar? Yes, it is just normal convolution (but on OGFs instead of EGFs).

Firstly, let's find the EGF of $b(k)$. This is just a special case of $a(k)$ when $s_i = 0$, so we have $B(x) = \prod_{i=1}^{n} \frac{[\exp(q_i x) + \exp(-q_i x)]}{2}$.

To relate $c(k)$ with $a(k), b(k)$, we need to look at the OGFs of a and b (call them $A_o(x)$ and $B_o(x)$), since the recurrence we found is related to the convolution of OGFs. Thus, defining $C(x)$ and $C_o(x)$ analogously, we have $A_o(x) = C_o(x)B_o(x)$, so $C_o(x) = \frac{A_o(x)}{B_o(x)}$.

Our answer is $\sum_{k=0}^{\infty} kc(k)$ (recall the definition of expected value and $c(k)$). This is just $C_o'(1)$. By Quotient Rule, this is equivalent to finding $\frac{A_o'(1)B_o(1) - A_o(1)B_o'(1)}{B_o(1)^2}$.

Let's see if we can find $A_o(x)$ from $A(x)$. It is hard to extract the coefficients of $A(x)$ if we are looking at a product of sums like $\prod_{i=1}^{n} \frac{[\exp(q_i x) + (-1)^{s_i}\exp(-q_i x)]}{2}$. To turn this into a sum, let's **expand** the brackets! Note that since $q_i = \frac{p_i}{S}$, if we expand the whole thing, we will get a sum where each term is of the form $c_i \exp(\frac{i}{S}x)$ for some $-S \leq i \leq S$ (since we are multiplying terms of the form $\exp(\frac{j}{S}x)$ and the sum of p_i is S). In other words, $A(x) = \sum_{-S}^{S} a_i \exp\left(\frac{i}{S}x\right)$ for some constants a_i.

How do we expand the terms quickly? We can use dp! Go through the brackets one by one, and maintain $dp[i][j]$ which is the coefficient of $\exp\left(\frac{j}{S}x\right)$ after expanding i brackets. You can do dp transitions in $O(1)$ to get the final answer in $O(nS)$ time.

From $A(x) = \sum_{i=-S}^{S} a_i \exp\left(\frac{i}{S}x\right)$, we can find a formula for $A_o(x)$. Indeed, $\exp\left(\frac{i}{S}x\right) = \sum_{j\geq 0}\left(\frac{(\frac{i}{S}x)^j}{j!}\right)$, so by removing the $j!$ terms we get a formula for $A_o(x)$. Specifically,

$$A_o(x) = \sum_{i=-S}^{S} a_i \sum_{j\geq 0}\left(\frac{i}{S}x\right)^j = \sum_{i=-S}^{S}\frac{a_i}{1-\frac{i}{S}x}.$$

Similarly, we can derive $B_o(x) = \sum_{i=-S}^{S}\frac{b_i}{1-\frac{i}{S}x}$ for some constants b_i.

We want to compute $A_o(1)$, $A_o'(1)$ (and similar for B). However, we run into a slight problem of dividing by zero if we try to do it directly, since $1-\frac{S}{S}(1)=0$. However, since $C_o(x)=\frac{A_o(x)}{B_o(x)}$, we can multiply both $A_o(x)$ and $B_o(x)$ by the troublesome $1-x$ term. Formally, let $E(x)=(1-x)A_o(x)$ and $F(x)=(1-x)B_o(x)$. Then, we only need to compute $E(1), E'(1), F(1)$ and $F'(1)$ (and as we shall see they are computable and easy to compute).

$E(1)$ is trivial, since $(1-x)A_o(x) = \sum_{i=-S}^{S}\frac{a_i(1-x)}{1-\frac{i}{S}x} = \sum_{i=-S}^{S-1}\frac{a_i(1-x)}{1-\frac{i}{S}x} + a_S$. Since all the terms except a_S when we substitute $x=1$, $E(1)=a_S$.

$E'(1)$ is not hard either. By Quotient Rule,

$E'(x) = \left[\sum_{i=-S}^{S-1}\frac{a_i(1-x)}{1-\frac{i}{S}x}\right]' = \sum_{i=-S}^{S-1}\frac{-a_i\left(1-\frac{i}{S}x\right)-a_i(1-x)\left(-\frac{i}{S}\right)}{\left(1-\frac{i}{S}x\right)^2}$. Substituting $x=1$ gives us $\sum_{i=-S}^{S-1}\frac{-a_i}{1-\frac{i}{S}}$, which is easy to compute in $O(S)$ time.

Thus, we have an easy-to-code $O(Sn)$ time solution.

In general, the trick of introducing A and B can be used to solve other problems that asks for the **first stopping time** of some system if you have multiple possible ending states and the time taken to reach from one ending state to another is equal, and the probability to reach an ending state in a fixed amount of moves is easy to compute.

Roots of Unity Filter

Next, we introduce a trick that is more rarely used in competitive programming but nevertheless interesting to learn. The motivation is the following classic problem.

Problem. Roots of Unity Filter Find the sum $\sum_{i\equiv r \pmod m}\binom{n}{i}$ modulo an arbitrary MOD.

Constraints: $1\leq n\leq 10^{18}, 2\leq m\leq 2000, 0\leq r\leq n-1, 10^8\leq MOD\leq 10^9$

Solution

This is a very standard problem in math olympiads, but let's see how to solve it in CP. Firstly, we need to know the trick behind the problem. Let's look at the case $m=2, r=0$, i.e. find $\sum_{i\equiv 0 \pmod 2}\binom{n}{i}$.

It is well-known that this is just 2^{n-1}, but where does it come from. Let us look at the generating function $\sum_{i\geq 0}\binom{n}{i}x^i$. If we plug in $x=1$, we get the sum of all binomial coefficients. However, we want to "filter" out the terms with even (or odd) power. What values can we easily substitute? Another easy candidate to try is $x=-1$, which gives us $\sum_{i\geq 0}\binom{n}{i}(-1)^i$. If we write out the terms, we get the equations:

$(1+1)^n = \binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\binom{n}{3}+\ldots$

$(1-1)^n = \binom{n}{0}-\binom{n}{1}+\binom{n}{2}-\binom{n}{3}+\ldots$

Notice that the odd-numbered terms are gone when we add up the equations! Thus, adding up the equations and dividing by 2, we obtain $\binom{n}{0}+\binom{n}{2}+\ldots=2^{n-1}$.

How to generalize the above method? Let's say we want to find $\binom{n}{0}+\binom{n}{3}+\binom{n}{6}+\ldots$.

We can split our sum into three parts:

$\binom{n}{0}x^0+\binom{n}{3}x^3+\binom{n}{6}x^6+\ldots$

$\binom{n}{1}x^1+\binom{n}{4}x^4+\binom{n}{7}x^7+\ldots$

$\binom{n}{2}x^2 + \binom{n}{5}x^5 + \binom{n}{8}x^8 + \ldots$

To leave only the sum of binomial coefficients in each group, we need to substitute x so that $x^3 = 1$.

Clearly, $x = 1$ works. What other values of x work?

The values of x such that $x^3 = 1$ are also called the 3rd **roots of unity**. In this case, $x = e^{\frac{2\pi i}{3}}, e^{\frac{4\pi i}{3}}$ are the other two roots of unity.

Let $S(n, i)$ denote the sum of $\binom{n}{k}$ for all $k \equiv i \pmod 3$. (so we want to find $S(n, 0), S(n, 1), S(n, 2)$).

Let $\omega = e^{\frac{2\pi i}{3}}$, then $1, \omega, \omega^2$ are the 3rd roots of unity. Note that $\omega^3 = 1$ by definition. We substitute $x = \omega^i$ for $0 \le i \le 2$, to get the following system of equations:

$S(n, 0) + S(n, 1) + S(n, 2) = (1 + 1)^n$

$S(n, 0) + \omega S(n, 1) + \omega^2 S(n, 2) = (1 + \omega)^n$

$S(n, 0) + \omega^2 S(n, 1) + \omega^4 S(n, 2) = (1 + \omega^2)^n$

How to solve this system of equations? We need an important identity of the roots of unity, which is $1 + \omega^k + \omega^{2k} + \ldots + \omega^{(m-1)k} = 0$ whenever k is not divisible by m (if $\omega^m = 1$). This is because by the geometric series formula, we have $1 + \omega^k + \omega^{2k} + \ldots + \omega^{(m-1)k} = \frac{1-\omega^{mk}}{1-\omega^k} = 0$ if $\omega^k \ne 1$.

Hence, in this specific case, $1 + \omega + \omega^2 = 0$ and $1 + \omega^2 + \omega^4 = 0$.

Summing all three equations gives us:

$3S(n, 0) = (1 + 1)^n + (1 + \omega)^n + (1 + \omega^2)^n$

How to obtain $S(n, 1)$ and $S(n, 2)$ easily? Here is a simple trick. Instead of looking at $(x + 1)^n$ only, we look at $x(x + 1)^n$ and $x^2(x + 1)^n$ as well and repeat the same process. Now, all coefficients are shifted properly and thus we can take the sum and divide by 3 to find $S(n, 1)$ and $S(n, 2)$ as in the previous case.

In summary, you should get something like:

$3S(n, 0) = (1 + 1)^n + (1 + \omega)^n + (1 + \omega^2)^n$

$3S(n, 2) = (1 + 1)^n + \omega(1 + \omega)^n + \omega^2(1 + \omega^2)^n$

$3S(n, 1) = (1 + 1)^n + \omega^2(1 + \omega)^n + \omega(1 + \omega^2)^n$ (note that $\omega^4 = \omega$)

The remaining problem is how to evaluate the values from this formula. We have a problem because ω seems to represent a complex number here ($\omega = e^{\frac{2\pi i}{3}}$).

The idea is that we do not necessarily need to work in the world of complex numbers. What we require of ω is for it to satisfy $\omega^3 = 1$ and $\omega^k \ne 1$ for $1 \le k \le 2$. Let us compute our answer as a **polynomial** in ω, but modulo $\omega^3 - 1$ (which means that we will have $\omega^3 = 1$, $\omega^4 = \omega$, etc...). Also, obviously the coefficients will be computed modulo MOD.

For example, $(1 + 2\omega + \omega^2)(1 + \omega + 3\omega^2) = 3\omega^4 + 7\omega^3 + 6\omega^2 + 3\omega + 1 = 3\omega + 7 + 6\omega^2 + 3\omega + 1 = 6\omega^2 + 6\omega + 8$.

Hence, at any point of time we will always have a polynomial in ω with degree ≤ 2.

To compute something like $(1 + \omega)^n$, we can use the usual divide-and-conquer method for computing large powers. Multiplication of polynomials can be implemented naively.

We can generalize this method for any m. Let ω be the m-th root of unity, so $\omega^m = 1$. Let $S(n, r)$ denote the sum of $\binom{n}{k}$ over all $k \equiv r \pmod m$.

$(1 + w)^n = \sum_{i \ge 0} \binom{n}{i} w^i$ for any number w. We want to make the coefficients of the form $\binom{n}{jm+r}$ to match with the powers $w^0, w^m, w^{2m}, \ldots$ because these will help us obtain the sum $S(n, r)$. So, it is more helpful to consider the polynomial $w^{m-r}(1 + w)^n = \sum_{i \ge 0} \binom{n}{i} w^{i+m-r}$.

Substituting $w = 1, \omega, \omega^2, \ldots, \omega^{m-1}$ and summing up, we get

$$\sum_{j=0}^{m-1} w^{j(m-r)}(1 + w^j)^n = \sum_{j=0}^{m-1} \sum_{i \ge 0} \binom{n}{i} \omega^{j(i+m-r)} = \sum_{i \ge 0} \binom{n}{i} \sum_{j=0}^{m-1} (\omega^{i+m-r})^j.$$

Recall that $1 + w + w^2 + \ldots + w^{m-1} = 0$ whenever $w = \omega, \omega^2, \ldots, \omega^{m-1}$ (by the geometric series formula) and $= m$ whenever $w = 1$. Note that $\omega^{i+m-r} = 1$ if and only if $i \equiv r \pmod m$. Hence,

$\sum_{i \ge 0} \binom{n}{i} \sum_{j=0}^{m-1} (\omega^{i+m-r})^j = m \cdot \sum_{i \equiv r \pmod m} \binom{n}{i} = mS(n, r)$ (note the multiple of m).

It remains to compute $\frac{1}{m}\sum_{j=0}^{m-1} w^{j(m-r)}(1+w^j)^n$ modulo MOD. The easiest way to do this is to first calculate the polynomial $F(x) = x^{m-r}(1+x)^n$ modulo $x^m - 1$. We can do this via binary exponentiation and multiplying polynomials naively in $O(m^2 \log n)$ time (remembering to set $x^m = 1, x^{m+1} = x, \ldots$ after each multiplication).

After we obtained the polynomial, we are actually done. The sum we want to find is $\frac{1}{m}[F(1) + F(\omega) + F(\omega^2) + \ldots + F(\omega^{m-1})]$. Letting $F(x) = \sum_{i=0}^{m-1} a_i x^i$, we realize that the desired sum is

$\frac{1}{m}\sum_{i=0}^{m-1} a_i \sum_{j=0}^{m-1} (\omega^j)^i = a_0$ since all the terms with $i \geq 1$ sum up to 0. Hence, we just need to output the constant term of $F(x)$.

Note: This problem is also solvable in $O(m \log m \log n)$ if MOD is a FFT-friendly modulo by using FFT to multiply the polynomials during exponentiation.

Next, we look at a nice harder example.

Problem. Rhyme Compute the sum of $\frac{n!}{x_1!x_2!\ldots x_k!}$ over all tuples of positive integers $(x_1, x_2, \ldots, x_k)$ such that $d|x_i$ and $\sum x_i = n$, modulo 19491001 (a prime).

Constraints: $n \leq 10^9, k \leq 2000, d \in \{4, 6\}$.

Solution

This problem is mentioned in jqdai0815's blog, but I will explain it in detail here. A funny story is that I came up with this problem and was trying to solve it one day, but then the next day I saw this problem on his post :) Anyway, I think this is a nice application of generating functions which deserves to be mentioned here.

By now, it should be clear what the first step should be. We want to seperate the terms $\frac{1}{x_i!}$ into different polynomials and convolute them, and the most obvious way to do it is to consider the following product:

$\left(1 + \frac{x^d}{d!} + \frac{x^{2d}}{2d!} + \ldots +\right)\left(1 + \frac{x^d}{d!} + \frac{x^{2d}}{2d!} + \ldots +\right)\ldots\left(1 + \frac{x^d}{d!} + \frac{x^{2d}}{2d!} + \ldots +\right)$ (k times).

It is easy to see that the coefficient of x^n is precisely the sum of $\frac{1}{x_1!x_2!\ldots x_k!}$ over all valid tuples of x_i.

Let $F(x) = 1 + \frac{x^d}{d!} + \frac{x^{2d}}{2d!} + \ldots$. How do we write $F(x)$ is a simpler way? For the case $d = 2$, we know that this is just $\cosh x = \frac{e^x + e^{-x}}{2}$. The idea is the same as the previous problem: we want to substitute different values of x into our "original function" (in this case it is $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots$) to "filter out" the powers that are not divisible by d.

Let's try to play with roots of unity again. Let ω be the m-th root of unity (so $\omega^m = 1$ and $1 + \omega + \omega^2 + \ldots + \omega^{m-1} = 0$). We substitute x as $\omega^i x$ for $0 \leq i \leq d-1$ into e^x and see what happens (note that this is where e^{-x} comes from for $d = 2$).

$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^d}{d!} + \ldots$

$e^{\omega x} = 1 + \frac{\omega x}{1!} + \frac{\omega^2 x^2}{2!} + \ldots + \frac{\omega^d x^d}{d!} + \ldots$

...

$e^{\omega^{d-1}x} = 1 + \frac{\omega^{d-1}x}{1!} + \frac{\omega^{2(d-1)}x^2}{2!} + \ldots + \frac{\omega^{d(d-1)}x^d}{d!} + \ldots$

If we sum all these equations, by the identity $1 + (\omega^i) + (\omega^i)^2 + \ldots + (\omega^i)^{d-1} = 0$ for i not divisible by d, we obtain

$e^x + e^{\omega x} + e^{\omega^2 x} + \ldots + e^{\omega^{d-1}x} = d\left(1 + \frac{x^d}{d!} + \frac{x^{2d}}{2d!} + \ldots +\right)$

Back to our problem, our goal is to find the coefficient of x^n in the following product:

$\frac{1}{d^k}\left(e^x + e^{\omega x} + e^{\omega^2 x} + \ldots + e^{\omega^{d-1}x}\right)^k$.

The next step requires some knowledge on roots of unity. The good thing about $d = 4, 6$ is that $\phi(4) = \phi(6) = 2$, so the d-th cyclotomic polynomial has degree 2 (it is the minimal polynomial of the primitive d-th roots of unity). It can be obtained via the expansion of $\prod_{\gcd(i,d)=1} (x - \omega^i)$ (you can find more details on the wikipedia page).

For example, for $d = 4$, we have $\omega^2 + 1 = 0$ and for $d = 6$, we have $\omega^2 - \omega + 1 = 0$. In both cases, we can always represent ω^i in the form of $a\omega + b$ by repeatedly reducing the maximal power using this equation.

Thus, if we let $\omega^i = a_i + b_i\omega$, our goal is to find $[x^{n}]\frac{1}{d^{k}}\left(\displaystyle\sum_{i=0}^{d-1} e^{(a_i+b_i\omega)x}\right)^{k}.

Notice that the terms of the form e^{ax} and $e^{b\omega x}$ are in some sense separated. We make the substitution $u = e^x$ and $v = e^{\omega x}$ to make things simpler:

$[x^n]\frac{1}{d^k}\left(\sum_{i=0}^{d-1} u^{a_i}v^{b_i}\right)^k$.

Notice that $-1 \le a_i, b_i \le 1$ (you can explicitly write out these values from the definition). If we expand this bivariate polynomial in u and v, we'll get a sum where all the terms are of the form $u^i v^j$ if $-k \le i, j \le k$ (consider the way to choose the terms in the expansion).

To avoid dealing with negative terms, let us multiply by $(uv)^k$. Let $F(u,v) = \sum_{i=0}^{d-1} u^{a_i+1}v^{b_i+1}$. We want to find $G(u,v) = F(u,v)^k$. Note that $G(u,v) = \sum_{0\le i,j\le 2k} g_{i,j}u^i v^j$ for some constants $c_{i,j}$ (define $f_{i,j}$ similarly).

While we can do something like 2D FFT, it is probably not going to pass the time limit. The next step is a magical trick mentioned in jqdai0815's article. The idea is that we want to find a recurrence on the coefficients $g_{i,j}$ so that we can use dp to compute them in $O(k^2)$ time. Let's **differentiate** both sides of $G(u,v) = F(u,v)^k$ with respect to u (or v, it doesn't matter). Let $g(u,v)$ and $f(u,v)$ denote the partial derivative of $G(u,v)$ and $F(u,v)$ with respect to u. Then, by the Chain Rule,

$$g(u,v) = kF(u,v)^{k-1}f(u,v)$$

Noting that $F(u,v)^{k-1} = \frac{G(u,v)}{F(u,v)}$, we have

$$F(u,v)g(u,v) = kG(u,v)f(u,v)$$

Comparing the coefficients of $u^i v^j$ on both sides, and noting that the coefficient of $u^i v^j$ of $f(u,v)$ is $(i+1)f_{i+1,j}$, we obtain the recurrence

$$\sum_{0\le i_1\le i, 0\le j_1\le j}(i+1-i_1)g_{i+1-i_1,j-j_1}f_{i_1,j_1} = k\sum_{0\le i_1\le i, 0\le j_1\le j}(i_1+1)f_{i_1+1,j_1}g_{i-i_1,j-j_1}$$

The good thing about this equation is that if we find the values of $g_{i,j}$ in increasing order of i followed by increasing order of j, the value of $g_{i,j}$ is only dependent on previous values! A subtle note is that $f_{0,0} = 0$ but $f_{0,1} \ne 0$. Thus, if you only leave the summand with $i_1 = 0$ and $j_1 = 1$ on the LHS and throw everything to the RHS, you can compute $g_{i,j}$ with dp. Note that there are only $O(1)$ nonzero values of $f_{i,j}$, so you can actually just iterate over $i_1, j_1 \le 2$ to compute the dp transitions in $O(1)$.

The base case is $g_{i,j}$ with i or j equal to 0, which can be found by binomial theorem after substituting $v = 0$ (I leave this as an exercise).

To summarize, you can find the expansion of $[x^n]\frac{1}{d^k}\left(\sum_{i=0}^{d-1} u^{a_i}v^{b_i}\right)^k$ in $O(k^2)$ time.

How to compute the answer after you reduce it to $[x^n]\frac{1}{d^k}\sum_{-k\le i,j\le k} g_{i+k,j+k}u^i v^j$?. Let's look at each individual term $[x^n]u^i v^j$. It is exactly equal to $[x^n]\exp(x(i+j\omega)) = \frac{1}{n!}(i+j\omega)^n$. The $\frac{1}{n!}$ cancels off with the numerator of $\frac{n!}{x_1!x_2!\ldots x_k!}$. Hence, it is sufficient to compute $(i+j\omega)^n$ for $-k \le i, j \le k$.

We can try to use the same trick as the previous problem: Maintaining a polynomial of the form $a + b\omega$ while doing binary exponentiation. However, it turns out to be a bit too slow (at least my implementation of it). Here, we can exploit the fact that we are computing the answer modulo $p = 19491001$.

You can find that 7 is a primitive root of 19491001, and $p-1$ is divisible by both 4 and 6. Thus, if we take $\omega = 7^{\frac{p-1}{d}}$, we will preserve the property that $\omega^d = 1$ and $\omega^i \ne 1$ for $1 \le i \le d-1$ (equivalently, $7^{\frac{p-1}{d}}$ is a primitive d-th root of unity in $\mathbb{Z}_p$). Thus, the computations can be done in integers which is faster.

In any case, this gives an $O(k^2 \log n)$ solution with relatively small constants.

Generating Functions that you can't compute "naively"

In some problems, the constraints might be too large to even compute the generating functions you found with FFT or algorithms involving polynomial operations. In this case, you usually need to analyze some special properties of the generating function you are dealing with (and thus it is helpful to recognize some common generating functions).

We start with a relatively easy example.

Problem. Perfect Permutations Find the number of permutations of length n with exactly k inversions, modulo 10^9+7.

Constraints: $1 \le n \le 10^9$, $0 \le k \le 10^5$, $n \ge k$. There are 100 testcases per input file.

Solution

Recall that for a permutation $p_1, p_2, \ldots, p_n$, an inversion is a pair of indices \$i<j$such that$p_i > p_j\$.

In my post 4 years ago, I mentioned how to solve an easier variant of this problem using a doubling trick to perform dp. Now, let's describe a much simpler and faster solution using generating functions.

Firstly, let us rephrase the problem. Suppose we start with the sequence 1 and add the elements $2, 3, \ldots, n$ to the permutation one by one. Note that by adding 2, we can increase the number of inversions by 0 or 1 in exactly one way each. Similar, among the i possible ways to add i into the sequence, there is exactly one way to increase the number of inversions by 0 to $i-1$ each. Thus, our problem is equivalent to finding the number of sequences $a_0, a_1, \ldots, a_{n-1}$ such that $a_i \le i$ for all i and $\sum a_i = k$.

Let us rephrase this in the language of generating functions. The idea is that we can "pick" any element from 0 to $i-1$ in the i-th "bracket", so it is natural to consider the function

$$F(x) = (1)(1+x)(1+x+x^2)(1+x+x^2+x^3)\ldots(1+x+x^2+\ldots+x^{n-1})$$

The coefficient of x^k in $F(x)$ gives us the answer.

Unfortunately, this polynomial is too large to compute directly. Let us rewrite it using the geometric series formula.

$$F(x) = \frac{1-x}{1-x} \cdot \frac{1-x^2}{1-x} \cdot \frac{1-x^3}{1-x} \cdot \ldots \cdot \frac{1-x^n}{1-x}$$
$$= \frac{(1-x)(1-x^2)\ldots(1-x^n)}{(1-x)^n}$$
$$= \prod_{i=1}^{n}(1-x^i) \cdot (1-x)^{-n}$$

We know how to find the coefficient of $[x^i]$ in $(1-x)^{-n}$ for $0 \le i \le k$ (recall that $[x^i]\ (1-x)^{-n} = \binom{n+i-1}{i}$). Hence, it is sufficient to find $[x^j]\prod_{i=1}^{n}(1-x^i)$ for all $0 \le j \le k$.

As $n \ge k$, we actually only need to compute $(1-x)(1-x^2)\ldots(1-x^k)$ (as larger terms here don't contribute to the coefficients of lower powers of x). We will present an $O(k)$ solution which can solve the problem even when $k \le 10^6$.

The trick is that since the larger terms $1-x^{k+1}$, $1-x^{k+2}$, etc doesn't matter in our product, why not just add all of them and consider the infinite product $\prod\limits_{n\ge1}(1-x^n)$. It turns out that this is a well-known generating function, whose series expansion has a simple form given by the Pentagonal number theorem.

The theorem states that $\prod\limits_{n\ge1}(1-x^n) = 1 + \sum\limits_{i\ge1}(-1)^i\left(x^{\frac{i(3i+1)}{2}} + x^{\frac{i(3i-1)}{2}}\right)$. There is a nice bijective proof of this (which you can find on Wikipedia or Enumerative Combinatorics Volume 1), but we only need to use the formula here.

From the series expansion, it is obvious how we can extract the coefficients in $O(k)$ time (actually even in $O(\sqrt{k})$ time).

Note that since n is large, to compute $\binom{n+i-1}{i}$ for all i, you need to write it in the form $\frac{(n+i-1)(n+i-2)\ldots(n)}{i!}$ and calculate it in increasing order of i by multiplying a suitable factor each time i increases.

Overall, we get an $O(k)$ time solution, which is more than enough for this problem.

Next, we look at a problem that has a natural statement in generating functions, but it turns out that the computation in generating functions is quite tricky. The official editorial has a nice and simpler solution using a magical observation, but to demonstrate the power of generating functions I will show an alternative method (which seems more straightforward and generalizable).

Problem. Sum of Fibonacci Sequence Let $d_{n,k}$ be defined by the recurrence $d_{1,1} = d_{1,2} = 1$, $d_{1,k} = d_{1,k-1} + d_{1,k-2}$ for $k \ge 3$, and $d_{n,k} = \sum\limits_{i=1}^{k} d_{n-1,i}$ for $n \ge 2$, $k \ge 1$.

Compute $d_{n,m}$ modulo 998244353.

Constraints: $1 \le n \le 200000$, $1 \le m \le 10^{18}$

Solution

Let's get straight to the generating functions. Define $P_n(x)$ as the OGF for $d_{n,k}$. $d_{1,k}$ is just the Fibonacci sequence, so we know that $P_1(x) = \frac{x}{1-x-x^2}$.

How to obtain $P_n(x)$? Thanks to our wonderful "Prefix Sum Trick", we know that multiplying $P_1(x)$ by $\frac{1}{1-x}$ gives us $P_2(x)$ because $d_{2,k}$ is just the prefix sum of $d_{1,k}$. Similarly, we have $P_n(x) = \frac{1}{(1-x)^{n-1}}P_1(x) = \frac{1}{(1-x)^{n-1}} \cdot \frac{x}{1-x-x^2}$.

However, now we have some problems, because we need to calculate the coefficient of x^m in this function with m up to 10^{18}. There is no way we can expand this naively and thus we need to do something clever.

The main annoyance is that we are dealing with a product in the denominator. We know how to compute $[x^m]\frac{1}{(1-x)^{n-1}}$ and $[x^m]\frac{x}{1-x-x^2}$ fast (the former is just some binomial coefficient while the latter is just the Fibonacci sequence). However, we don't know how to compute their convolution fast. The trick here is to forcefully separate these two functions by **partial fractions**. Note that the theory of partial fractions tell us that

$$\frac{x}{(1-x)^{n-1}\cdot(1-x-x^2)} = \frac{A(x)}{(1-x)^{n-1}} + \frac{B(x)}{1-x-x^2}$$

where $A(x)$ is a polynomial of degree $\leq n-2$ and $B(x)$ is a polynomial of degree ≤ 1. If we can find $A(x)$ and $B(x)$, then our problem will be much easier to solve. However, $A(x)$ and $B(x)$ is hard to find explicitly on paper (though you can guess $B(x)$ by some pattern from small cases). How do we proceed?

Here is a painless way to do it. Firstly, we clear the denominators to obtain the identity

$$A(x)(1-x-x^2)+B(x)(1-x)^{n-1}=x.$$

Since this is an identity, it remains true for any value of x we substitute! What convenient values of x can we substitute? We want to make either $1-x-x^2$ or $1-x$ equal to 0 to leave us with only one polynomial to deal with. Substituting $x=1$ doesn't tell us too much since $A(x)$ has degree $\leq n-2$ and we can't determine it with only 1 point of information. However, what if we let $1-x-x^2=0$? We can solve the quadratic equation to get two roots $a+b\sqrt{5}$ and $a-b\sqrt{5}$ for some constants a,b. Substituting $x=a\pm b\sqrt{5}$, we have the nice pair of equations

$$B(a\pm b\sqrt{5})(1-(a\pm b\sqrt{5}))^{n-1}=a\pm b\sqrt{5}.$$

Since $B(x)$ is linear, if we let $B(x)=mx+c$ we can solve for the coefficients of B using these 2 simultaneous equations! An implementation detail is that since we are dealing with $\sqrt{5}$ here, it is helpful to store the numbers as a pair (a,b) which denotes $a+b\sqrt{5}$ and do arithmetic on these pairs. The value $(1-(a\pm b\sqrt{5}))^{n-1}$ can be found via binary exponentiation in $O(\log n)$ time (or even naive multiplication works here).

In any case, after some work you can find $B(x)$. How do we find $A(x)$? Just refer to the identity to obtain $A(x)=\frac{x-B(x)(1-x)^{n-1}}{1-x-x^2}$, which we can compute in $O(n\log n)$ time with one FFT (note that you don't even need to compute the reciprocal of a polynomial, as you can just divide using long division since $A(x)$ is a polynomial).

It remains to compute $[x^m]\frac{A(x)}{(1-x)^{n-1}}+\frac{B(x)}{1-x-x^2}$, which is a significantly easier task. For the second term, we can partition it into $\frac{Mx}{1-x-x^2}$ and $\frac{C}{1-x-x^2}$ and note that both are generating functions for the Fibonacci numbers and thus we can just compute the coefficient of x^m as a Fibonacci number in $O(\log m)$ time (using any divide-and-conquer method you like).

For the first term, we have $A(x)(1-x)^{-(n-1)}$ and we know how to compute both $[x^i]A(x)$ and $[x^j]\left(1-x\right)^{-(n-1)}$ (it is a binomial coefficient) for all i,j. Since $A(x)$ is of degree $\leq n-2$, we can iterate through all i and compute the sum of $[x^i]A(x)\cdot[x^{m-i}]\left(1-x\right)^{-(n-1)}$ (the latter requires large binomial coefficients which should be computed in a similar manner as the previous problem).

Thus, we obtain an $O(n\log n+\log m)$ solution.

I believe you can generalize this solution to solve other recurrences of a similar form.

To end this section, we conclude with a problem that heavily relies on linear recurrences. Actually, it might be a stretch to call this a generating function problem but I just want to demonstrate the trick of using generating functions to compute linear recurrences which is basically the same as the one shown here.

Problem. Sum Modulo You have a number x which is initially K. Every second, for $1\leq i\leq n$, there is a probability p_i that you will replace x with $(x-i)\pmod{M}$. Find the expected number of moves before the counter goes to 0. p_i are given as $\frac{A_i}{\sum A_i}$ for some positive integers $A_1,A_2,\dots,A_n$ and your output should be modulo 998244353 (and it is guaranteed that you don't have to worry about division by 0).

Constraints: $1\leq n\leq\min(500,M-1)$, $2\leq M\leq 10^{18}$, $1\leq A_i\leq 100$

Solution

Let us derive a simple recurrence first. Let $E(i)$ denote the expected number of moves to reach 0 from i. Clearly, $E(0)=0$, and we have $E(i)=p_1E(i-1)+p_2E(i-2)+...+p_nE(i-n)+1$. We use the convention that $E(-r)=E(M-r)$.

First, we look at the high-level idea of the solution. The idea is that for $i\geq n$, we can always write $E(i)$ in terms of $c_0E(0)+c_1E(1)+c_2E(2)+...+c_{n-1}E(n-1)+C$ for some constants c_i and C by repeatedly using the recurrence relation. In particular, $E(M+1)=E(1)$, $E(M+2)=E(2)$, ..., $E(M+n-1)=E(n-1)$ can all be represented in this form in a non-trivial manner using the recurrence (note that the recurrence doesn't hold for multiples of M, but $E(M+i)$ can still be represented non-trivially in this form for $1\leq i\leq n-1$).

Hence, moving everything unknown to one side and the constants to the other, we have $n-1$ nontrivial equations of the form $c_{i,1}E(1)+c_{i,2}E(2)+...+c_{i,n-1}E(n-1)=C_i$ for $1\leq i\leq n-1$. We can solve this system of equations using Gaussian Elimination in $O(n^3)$ time.

Once we obtained the values of $E(1),E(2),\dots,E(n-1)$, we just need to represent $E(k)$ in terms of $E(0),E(1),..,E(n-1)$ and a constant and we are done.

The remaining problem here is how do we get the representation of $E(m)$ in terms of $E(0),E(1),..,E(n-1)$ and a constant C for any m in an efficient manner.

Let's assume for the time being that $E(M), E(2M), E(3M), \ldots$ also satisfy the linear recurrence $E(i) = p_1E(i-1) + p_2E(i-2) + \ldots + p_nE(i-n) + 1$. Thus, the values of $E(M)$, $E(M+1)$, ... might not be what we want now but we will deal with this issue later.

Now, we use the same trick as in this blog. For a polynomial $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_kx^k$ define its valuation $val(f)$ as $a_0 + a_1E(1) + \ldots + a_kE(k)$. Since $E(i) - p_1E_(i-1) - p_2E(i-2) - \ldots - p_nE(i-n) = 1$ for all $i \geq n$, we have $val(x^i - p_1x^{i-1} - p_2x^{i-2} - \ldots - p_nx^{i-n}) = 1$ for all $i \geq n$. Let $P(x) = x^n - p_1x^{n-1} - p_2x^{n-2} - \ldots - p_0x^0$ for convenience. Then, $val(x^kP(x)) = 1$ for all $k \geq 0$. Since val is additive, for any polynomial $Q(x)$, we have $val(Q(x)P(x)) = Q(1)$ (sum of coefficients, since x^k corresponds to 1. If this is unclear, try writing $Q(x)$ as $q_0x^0 + q_1x^1 + \ldots$).

Our goal is to find $val(x^m)$ for some integer m. By the division algorithm, we can write $x^m = P(x)Q(x) + R(x)$ where $R(x)$ is a polynomial of degree $\leq n-1$. Hence, $val(x^m) = val(P(x)Q(x) + R(x)) = val(P(x)Q(x)) + val(R(x)) = Q(1) + val(R(x))$. Notice that $val(R(x))$ is already a linear combination of $E(1), E(2), \ldots, E(n-1)$, while $Q(1)$ is a constant. Thus, if we can somehow find $Q(x)$ and $R(x)$, we can represent $E(m)$ as a linear combination of $E(1)$, $E(2)$, ..., $E(n-1)$ and a constant.

To find $Q(1)$ and $R(x)$, we can use a divide-and-conquer algorithm similar to binary exponentiation. Consider the function $solve(m)$ that returns a pair denoting $R(x)$ and $Q(1)$ (a polynomial and a constant). If m is even, let $R_1(x)$, $Q_1(1)$ be the return value of $solve\left(\frac{m}{2}\right)$. Let $x^{\frac{m}{2}} = P(x)Q_1(x) + R_1(x)$. We have

$$x^m = (P(x)Q_1(x) + R_1(x))(P(x)Q_1(x) + R_1(x)) = P(x)^2Q_1(x)^2 + P(x)[2R_1(x)Q_1(x)] + R_1(x)^2 = P(x)[P(x)Q_1(x)^2 + 2R_1(x)Q_1(x)] +$$
.

Let $R_1(x)^2 = Q_2(x)P(x) + R_2(x)$ (just do long division). It follows from the equation above that we can take $R(x) = R_2(x)$ and $Q(1) = P(1)Q_1(1)^2 + 2R_1(1)Q_1(1) + Q_2(1)$. The case where m is odd is similar. Thus, we can compute $val(x^m)$ in $O(n^2 \log m)$ time.

Also, note that if we have computed $val(x^m)$, we can compute $val(x^{m+1})$ in $O(n^2)$ time using the same trick. Thus, we can use this method to compute a representation of $E(M-n+1)$, $E(M-n+2)$, ..., $E(M-1)$ in terms of $E(1), E(2), \ldots, E(n-1)$ and a constant in $O(n^2 \log m + n^3)$ time. The reason we cannot compute $E(M+1)$, $E(M+2)$, ..., $E(M+n-1)$ directly using val is because the recurrence doesn't hold for $E(M)$ as stated before. However, once we have the representation for $E(M-n+1)$, $E(M-n+2)$, ..., $E(M-1)$, we can now plug those into the original recurrence $E(i) = p_1E(i-1) + p_2E(i-2) + \ldots + p_nE(i-n) + 1$ and obtain the representations for $E(M+1)$, $E(M+2)$, ..., $E(M+n-1)$ in $O(n^2)$ time each.

This gives us a $O(n^2(n + \log m))$ solution.

Lagrange Inversion Formula

Finally, inspired by this Div. 1 F problem, I looked up on some applications on Lagrange Inversion Formula and found a few examples on it. I would like to end this article by demonstrating a few applications of it.

Some examples here are from Enumerative Combinatorics Volume 2 and some are from jcvb's Chinese paper on generating functions.

The idea of the Lagrange Inversion Formula is that sometimes we want to find the compositional inverse of a function but it is difficult to find. However, the coefficients of this inverse function might have a simpler formula, which we can obtain from Lagrange Inversion Formula.

There are many variants of stating the Lagrange Inversion Formula, so I will show what I think is the most helpful version of it (also given in this comment).

Theorem. Let $F(x), G(x)$ be formal power series which are compositional inverses (i.e. $F(G(x)) = x$). Suppose $F(0) = G(0) = 0$, $[x^1]F(x) \neq 0$, $[x^1]G(x) \neq 0$, then

$$[x^n]G(x) = \frac{1}{n}[x^{-1}]\frac{1}{F(x)^n}$$

Also, for any power (or Laurent) series $H(x)$, we have

$$[x^n]H(G(x)) = \frac{1}{n}[x^{-1}]H'(x)\frac{1}{F(x)^n}$$

Note: Laurent Series can be intuitively seen as the generalization of power series where the powers can go negative.

Intuitively, if you "know" how to compute $F(x)$, then you can also get the coefficients of the compositional inverse of $F(x)$. Let's go through a few examples.

Tree Enumeration

Problem. Count the number of labelled trees on n vertices (number of trees where vertices are labelled).

Solution

If you have heard of Cayley's Formula, you know that the answer is n^{n-2}.

Let us count the number of **rooted** trees on n vertices. Call this number $t(n)$. If we remove the root from a rooted tree, we get a collection of rooted subtrees. This allows us to get the recurrence

$t(n+1) = (n+1)\sum_{k\geq 0}\sum_{i_1+i_2+\ldots+i_k=n, i_j\geq 1}\frac{n!}{i_1!i_2!\ldots i_k!}t(i_1)t(i_2)\ldots t(i_k)\cdot\frac{1}{k!}$, as we have $n+1$ ways to choose the root, $\frac{n!}{i_1!i_2!\ldots i_k!}$ ways to assign the non-root nodes to a subtree (we divide by $k!$ because each set of subtrees is counted $k!$ times for each permutation), and $t(i_1)t(i_2)\ldots t(i_k)$ is the number of ways to form each rooted subtree.

Rearranging and multiplying by x^{n+1}, we obtain

$$\frac{t(n+1)}{(n+1)!}x^{n+1} = x\sum_{k\geq 0}\frac{1}{k!}\cdot\sum_{i_1+i_2+\ldots+i_k=n, i_j\geq 1}\frac{t(i_1)x^{i_1}}{i_1!}\cdot\frac{t(i_2)x^{i_2}}{i_2!}\cdot\ldots\cdot\frac{t(i_k)x^{i_k}}{i_k!}.$$

Hence, letting $T(x)$ be the EGF of $t(n)$ (and define $t(0)=0$ for simplicity), we have

$$T(x) = \sum_{n\geq 0}\frac{t(n+1)}{(n+1)!}x^{n+1} = x\sum_{k\geq 0}\frac{1}{k!}\cdot\sum_{n\geq 0}\sum_{i_1+i_2+\ldots+i_k=n, i_j\geq 1}\frac{t(i_1)x^{i_1}}{i_1!}\cdot\frac{t(i_2)x^{i_2}}{i_2!}\cdot\ldots\cdot\frac{t(i_k)x^{i_k}}{i_k!}$$

$= x\sum_{k\geq 0}\frac{T(x)^k}{k!}$ (verify that we get the previous line by expanding this)

$= xe^{T(x)}$

Hence, we have the functional equation $T(x) = xe^{T(x)}$. It is not easy to solve this equation directly, however. However, we can see that we have a function in $T(x)$ which is equal to x, which motivates us to write

$T(x)e^{-T(x)} = x$

and let $F(x) = xe^{-x}$, $G(x) = T(x)$ in Lagrange Inversion Formula, we obtain

$[x^n]T(x) = \frac{1}{n}[x^{-1}]\frac{1}{(xe^{-x})^n} = \frac{1}{n}[x^{-1}]x^{-n}e^{nx} = \frac{1}{n}[x^{n-1}]e^{nx} = \frac{1}{n}\cdot\frac{n^{n-1}}{(n-1)!} = \frac{n^{n-1}}{n!}$.

Thus, $t(n) = n^{n-1}$.

Finally, to count the number of unrooted labelled trees, simply divide $t(n)$ by n as each unrooted tree is counted n times in $t(n)$ by the n choices of root. Hence, the answer is n^{n-2}.

Number of 2-edge connected graphs

Problem. Find the number of labelled 2-edge connected graphs on n vertices. A graph is 2-edge connected graphs if it has no bridges, i.e. removing any edge does not disconnect the graph.

Constraints: $n \leq 3\cdot 10^5$

Solution

Let's warmup with an easier problem. Suppose we want to count the number of labelled connected graphs on n vertices. There are different ways to compute this, but let's show a method using EGFs as it will be useful later. Let $C(x)$ be the EGF of the number of connected labelled graphs.

Connected graphs on n vertices are hard to count, but labelled graphs on n vertices are trivial: there are exactly $2^{\binom{n}{2}}$ labelled graphs on n vertices since we can either choose each edge or not. Let $G(x)$ denote the EGF of the number of labelled graphs. The nice thing is that labelled graphs are made up of several connected components, so we can use a similar argument as above (I will skip this step) to obtain $G(x) = \exp(C(x))$, which gives $C(x) = \ln(G(x))$. Since we know how to find $G(x)$, we can find $C(x)$ in $O(n\log n)$ time using polynomial operations.

Ok, now let's return to our problem. Let $b(n)$ denote the number of 2-edge connected graphs on n vertices and $B(x)$ be its EGF. Our goal is to find $B(x)$.

The idea is to relate $b(n)$ with $c(n)$. Suppose we have a labelled connected graph on n vertices, say G. Any connected graph G can be decomposed into a bridge tree, where each vertex is a 2-edge connected component and the edges of the tree are the bridges of the graph. Let s be the size of the 2-edge connected component containing vertex 1, and fix the 2-edge connected component containing vertex 1 as the root of the bridge tree. There are $\binom{n-1}{s-1}$ ways to choose the other elements in the component and $b(s)$ ways to connect edges within the component. Now, in the bridge tree, let $a_1, a_2, \ldots, a_k$ be the total weight of subtrees of the children of the root (we define the weight of a vertex in the bridge tree as the size of the 2-edge connected component represented by it and the weight of a subtree as the sum of weights of all vertices in the subtree). Then, there are $\frac{(n-s)!}{a_1!a_2!\ldots a_k!}$ ways to assign the remaining $n-s$ vertices to each subtree. Each subtree represented a general connected graph on a_i vertices, so there are $c(a_i)$ ways to connect edges in the i-th subtree. Finally, there are sa_i ways to choose the "bridge" between subtree i and the root, because we need to pick exactly one vertex from the subtree and the root component to connect.

Summing over all tuples $(a_1, a_2, \ldots, a_k)$ with sum $n-s$, and dividing by $k!$ to account for the fact that each set of subtrees is counting $k!$ times, we obtain the recurrence

$$c(n) = \sum_{s=1}^{n} b(s) \cdot \binom{n-1}{s-1} \cdot (n-s)! \cdot \sum_{k\geq 0} \frac{1}{k!} \sum_{a_1+a_2+\ldots+a_k=n-s} \prod_{j=1}^{k} \frac{c(a_j)\cdot a_j \cdot s}{a_j!}$$

$$= \sum_{s=1}^{n} b(s) \cdot \frac{(n-1)!}{(s-1)!} \cdot \sum_{k\geq 0} \frac{s^k}{k!} \sum_{a_1+a_2+\ldots+a_k=n-s} \prod_{j=1}^{k} \frac{c(a_j)\cdot a_j}{a_j!}$$

Hence,

$$\tfrac{nc(n)}{n!} = \sum_{s=1}^{n} \frac{b(s)}{(s-1)!} \cdot \sum_{k\geq 0} \frac{s^k}{k!} \sum_{a_1+a_2+\ldots+a_k=n-s} \prod_{j=1}^{k} \frac{c(a_j)\cdot a_j}{a_j!}$$

Note that we have $nc(n)$ and $a_j \cdot c(a_j)$ appearing in the summand. It seems like it is easier to consider the EGF of the sequence nc_n, say $C_1(x)$. Note that $C_1(x) = xC'(x)$ by the "multiplication by n" rule. In any case, we work in generating functions to obtain

$$C_1(x) = \sum_{n\geq 0} \frac{nc(n)}{n!} x^n = \sum_{n\geq 0} x^n \sum_{s=1}^{n} \frac{b(s)}{(s-1)!} \cdot \sum_{k\geq 0} \frac{s^k}{k!} \sum_{a_1+a_2+\ldots+a_k=n-s} \prod_{j=1}^{k} \frac{c(a_j)\cdot a_j}{a_j!}$$

$$= \sum_{n\geq 0} \sum_{s=1}^{n} \frac{b(s)x^s}{(s-1)!} \cdot \sum_{k\geq 0} \frac{s^k}{k!} \sum_{a_1+a_2+\ldots+a_k=n-s} \prod_{j=1}^{k} \frac{c(a_j)\cdot a_j \cdot x^{a_j}}{a_j!}$$

Simplifying the interior sum and product by noting its relevance to the coefficients of $C_1(x)$, we obtain

$$= \sum_{n\geq 0} \sum_{s=1}^{n} \frac{b(s)x^s}{(s-1)!} \cdot \sum_{k\geq 0} \frac{s^k x^{n-s}}{k!} [x^{n-s}] C_1(x)^k$$

$$= \sum_{n\geq 0} \sum_{s=1}^{n} \frac{b(s)x^s}{(s-1)!} \cdot x^{n-s} [x^{n-s}] \sum_{k\geq 0} \frac{s^k}{k!} C_1(x)^k$$

$$= \sum_{n\geq 0} \sum_{s=1}^{n} \frac{b(s)x^s}{(s-1)!} \cdot x^{n-s} [x^{n-s}] \exp(sC_1(x))$$

Swapping the order of summation, we get

$$= \sum_{s\geq 0} \frac{b(s)x^s}{(s-1)!} \sum_{n\geq s} x^{n-s} [x^{n-s}] \exp(sC_1(x))$$

$$= \sum_{s\geq 0} \frac{b(s)x^s}{(s-1)!} \sum_{n\geq 0} x^{n} [x^{n}] \exp(sC_1(x))$$

$$= \sum_{s\geq 0} \frac{b(s)x^s}{(s-1)!} \exp(sC_1(x)).$$

$$= \sum_{s\geq 0} \frac{sb(s)(x\exp(C_1(x))^s}{s!}.$$

Let $B_1(x) = xB'(x)$ be the EGF of $sb(s)$. Thus, we obtain $C_1(x) = B_1(x\exp(C_1(x)))$.

We know how to find C_1 and our aim now is to find the coefficients of B_1.

Let $P(x) = C_1(x)$ and $Q(x) = x\exp(C_1(x))$. We have the relation $P(x) = B_1(Q(x))$ and want to find $[x^n]B_1(x)$. To make $B_1(x)$ appear, substitute x as $Q^{-1}(x)$ (the compositional inverse exist because $Q(x)$ is a power series with a nonzero x term and no constant term). Thus, $B_1(x) = P(Q^{-1}(x))$. This looks very similar to Lagrange Inversion Formula!

Indeed, we let $P = H$ and $Q^{-1} = G$. Then, $Q = F$, so we have

$$[x^n]B_1(x) = [x^n]P(Q^{-1}(x)) = \tfrac{1}{n}[x^{-1}]P'(x)\tfrac{1}{Q(x)^n} = \tfrac{1}{n}[x^{-1}]C_1'(x)\tfrac{1}{x^n\exp(C_1(x))^n} = \tfrac{1}{n}[x^{n-1}]\tfrac{C_1'(x)}{\exp(C_1(x))^n}.$$

This can be computed via standard polynomial operations in $O(n\log n)$ time.

Coefficient of fixed x^k in $f(x)^i$

This is a more of a trick than a specific problem. Let $f(x)$ be a power series with a compositional inverse ($[x^0]f(x) = 0$, $[x^1]f(x) \neq 0$). We can find the coefficient of x^k (assume $k \geq 1$) in $f(x)^i$ for all $1 \leq i \leq n$ in $O(n \log n)$ time (assume $k = O(n)$).

Let $ans(i)$ denote the answer for fixed i. Instead of looking at $ans(i)$ as a sequence, let's introduce a new variable u and consider the OGF

$$A(u) = ans(0) + ans(1)u + ans(2)u^2 + \ldots = \sum_{n\geq 0}[x^k]f(x)^n u^n = [x^k]\sum_{n\geq 0}(f(x)u)^n = [x^k]\frac{1}{1-uf(x)}.$$

Since $f(x)$ has a compositional inverse (say $g(f(x)) = x$), by Lagrange Inversion formula (with $H(x) = \frac{1}{1-ux}$), we obtain

$$[x^k]\frac{1}{1-uf(x)} = \frac{1}{k}[x^{-1}]\left(\frac{1}{1-ux}\right)'\left(\frac{1}{g(x)^k}\right) = \frac{1}{k}[x^{k-1}]\left(\frac{1}{1-ux}\right)'\left(\frac{1}{\left(\frac{g(x)}{x}\right)^k}\right).$$

Note that by Quotient Rule, $\left(\frac{1}{1-ux}\right)' = \frac{u}{(1-ux)^2}$.

Our goal is to rewrite our sum in a clear manner so that we can "read off" the coefficients of u^i. We try to change the problem of finding the coefficients of u^i into a problem about purely finding the coefficients of x^j of some function.

The idea is to expand the series

$\frac{u}{(1-ux)^2} = u(1-ux)^{-2} = u\sum_{i\geq 0}\binom{i+1}{1}(ux)^i$ (recall how to expand $(1-x)^{-2}$)

$= u^{i+1}\sum_{i\geq 0}(n+1)x^i$, thus

$$[x^k]\frac{1}{1-uf(x)} = \frac{1}{k}[x^{k-1}]\sum_{i\geq 0}(i+1)x^i u^{i+1}\left(\frac{1}{\left(\frac{g(x)}{x}\right)^k}\right)$$

Let's look at $ans(i+1)$, the coefficient of u^{i+1}. We have

$$ans(i+1) = [u^{i+1}]\frac{1}{k}[x^{k-1}]\sum_{i\geq 0}(i+1)x^i u^{i+1}\left(\frac{1}{\left(\frac{g(x)}{x}\right)^k}\right) = \frac{i+1}{k}[x^{k-i-1}]\frac{1}{\left(\frac{g(x)}{x}\right)^k}.$$

Now, our problem reduces to computing the coefficients of one **fixed** function $P(x) = \frac{1}{\left(\frac{g(x)}{x}\right)^k}$, which we can compute the first n terms of using the usual polynomial operations! Thus, we can compute $ans(i)$ for all i in $O(n \log n)$ time!

If $f(x)$ does not have a compositional inverse, it is possible to "adjust" our function f (create a new function related to f) so that it has a compositional inverse. I leave this as an exercise.

Final Boss: Div 1 F — Slime and Sequences

As the grand finale of this 2-part article, I would like to discuss the recent very difficult Div. 1 F problem which was the inspiration of the entire blog in the first place.

Problem. Slime and Sequences A sequence of positive integers s is called good if for each Error: Misplaced & that is present in s, the first occurrence of $k-1$ (which must exist) must occur before the last occurrence of k. Count the number of times each integer $1 \leq i \leq n$ appears over all good sequences of length n.

Constraints: $1 \leq n \leq 100000$

Solution

The official editorial has a solution using PIE which ends up with an application of Lagrange Inversion Formula very similar to the example just shown. You can try to see the connection between the previous example and the trick used in the official editorial (for the Lagrange Inversion part). Here, I will demonstrate Elegia's solution described here which to me is a more intuitive way to approach the problem (thanks to Elegia for helping me with some parts I didn't understand and showing that this appraoch can lead to a full solution!).

The first step is to reduce the problem into something simpler. If we look at the samples, we see that curiously the sum of all the answers in the output is $n \cdot n!$, which suggest that there are only $n!$ good sequences. This cannot be coincidence, and a bijection between permutations and good sequences should exist.

In fact, this is IMO 2002 Shortlist C3. Since the last occurrence of k must occur after the first occurrence of $k-1$, we can consider iterating through the good sequence from right to left several times. In the first run, we record the positions of the number 1s, from right to left. On the second run, we record the positions of the number 2s, from right to left and so on. At the end of the process, n distinct numbers from 1 to n are recorded. This will be our permutation.

We claim that this is the bijection we seek. The idea is that if we read the values in our final permutation p from left to right, every time we meet an ascent (Error: Misplaced &), it indicates that we have finished recording all occurrences of the current number and is now going from right to left again. The condition of good sequences guarantees that every time we record the occurrences of a new number, there will be a new ascent in our permutation. Thus, we can easily recover the good sequence by marking the ascents of the permutation.

This also gives us a nice way to formulate the problem. Let $ans(i)$ denote the i-th answer for our problem. For any permutation p, divide it into a minimum number of decreasing blocks. Let's say the block sizes are $b_1, b_2, \ldots, b_k$. Then, this permutation contributes b_1 to $ans(1)$, b_2 to $ans(2)$ and so on. For example, if $p = (5,3,2,7,1,4,6)$, then we divide it into blocks $[5,3,2]$, $[7,1]$, $[4]$, $[6]$. p increases b_1, b_2, b_3 and b_4 by 3, 2, 1, 1 respectively.

Now, let's do some double counting. Fix a position $1 \le j \le n$. Can we calculate how many times this position contributes to $ans(k)$ over all $n!$ permutations (for a fixed k)?

Note that for position j to contribute to the answer, the prefix p_1, p_2, ..., p_j must contain exactly $k-1$ ascents. Additionally, the last $n-j$ elements can be permuted in any manner. Finally, there are $\binom{n}{j}$ ways to choose which elements occur in the prefix. Thus, if we let $A(n,k)$ denote the number of permutations of length n with exactly $k-1$ ascents, position j contributes $(n-j)!\binom{n}{j}A(j,k) = \frac{n!}{j!}A(j,k)$ to the answer.

Summing over all j, we get the nice-looking formula $ans(k) = n!\displaystyle\sum_{j=1}^{n}\frac{A(j,k)}{j!}$. From here, you can derive a simple $O(n^2)$ solution, since there is a simple recurrence for $A(n,k)$. However, we are looking for more here.

For convenience, now we ignore the $n!$ term and just let $ans(k)$ be $\displaystyle\sum_{j=1}^{n}\frac{A(j,k)}{j!}$. We can multiply each answer by $n!$ at the end.

What's nicer is that $A(n,k)$ is actually a well-known sequence called the Eulerian numbers! If you use Wikipedia or Enumerative Combinatorics 1, you can find the formulas related to generating functions involving $A(n,k)$.

Here, I use the notation $A(n,k)$ in Enumerative Combinatorics 1, which differs a bit from Wikipedia k is shifted and $A(0,0)=1$. You can either derive or google the following formula (first thing that comes up when you look for generating functions of Eulerian numbers, though it may look slightly different due to variable shifting):

$$F(x,y) = \sum_{k\ge 0}\sum_{n\ge 0}A(n,k)\frac{x^n}{n!}y^k = \frac{1-y}{1-ye^{(1-y)x}}$$

What we want to find is $\displaystyle\sum_{j=1}^{n}\frac{A(j,k)}{j!}$, so let us rewrite

$F(x,y) = \displaystyle\sum_{k\ge 0}y^k\sum_{n\ge 0}A(n,k)\frac{x^n}{n!}$ and focus on $G_k(x) = \displaystyle\sum_{n\ge 0}A(n,k)\frac{x^n}{n!}$ for a fixed k.

Observe that we want the **prefix sum** of the coefficients of $G_k(x)$. By the prefix sum trick, we get $ans(k) = [x^n]\frac{1}{1-x}\cdot G_k(x)$.

Thus, $ans(k) = [x^n y^k]\frac{1}{1-x}F(x,y) = [x^n y^k]\frac{1-y}{(1-x)(1-ye^{(1-y)x})}$, which is exactly the same expression quoted in Elegia's post.

The hard part is finding this fast. I got to this point on my own but didn't really know how to proceed (I wasn't even sure if it was doable) before reading Elegia's post so huge thanks to him!

Let us focus on the expression $(1-y)[x^n]\frac{1}{(1-x)(1-ye^{(1-y)x})}$ and keep in mind that we want to find the answer as a polynomial in y, which we can read the answer from.

The first major concern is that we have $e^{(1-y)x}$, which is an exponential of a multivariable function. This seems painfully awful to deal with especially if we need to expand it in power series form in the end. A nice trick here is to eliminate this nonsenes by making the substitution $z=(1-y)x$. Then, $x=\frac{z}{1-y}$ and our expression becomes:

$$(1-y)[x^n]\frac{1}{(1-x)(1-ye^{(1-y)x})} = (1-y)[x^n]\frac{1}{\left(1-\frac{z}{1-y}\right)(1-ye^z)} = (1-y)^{n+1}[z^n]\frac{1}{\left(1-\frac{z}{1-y}\right)(1-ye^z)}$$

Let us clear the denominator in $1-\frac{z}{1-y}$ by multiplying $1-y$ to the denominator, so we need to find

$$(1-y)^{n+2}[z^n]\frac{1}{(1-y-z)(1-ye^z)}$$

Ok, this looks simpler, though we still have products of multivariable functions. Life would be simpler if we can separate the two factors in the denominator. As a matter of fact, we can! Let's write the expression in **partial fractions**. Let $\frac{1}{(1-y-z)(1-ye^z)} = \frac{A}{1-y-z} + \frac{B}{1-ye^z}$. We want to find some simple functions A, B (hopefully in one variable) so that $A(1-ye^z) + B(1-y-z) = 1$.

Note that we have a linear function on y on the LHS if z is a treated as a constant. Thus, it motivates us to let A and B be functions in z that annilhates the LHS. We can treat z as a constant and compare the coefficients of y and 1 to obtain $A(z) = \frac{1}{1-e^z(1-z)}$, $B(z) = \frac{-e^z}{1-e^z(1-z)}$.

Substituting back, we need to find

$$(1-y)^{n+2}[z^n]\frac{1}{(1-y-z)(1-ye^z)} = (1-y)^{n+2}[z^n]\left(\frac{1}{(1-e^z(1-z))(1-y-z)} + \frac{-e^z}{(1-e^z(1-z))(1-ye^z)}\right)$$

It remains to compute $[z^n]$ of each fraction fast. Our main goal is to isolate the variable y as much as possible, treating z pretty much like a constant.

For example, let's look at the second fraction. We want to find $[z^n]\frac{-e^z}{(1-e^z(1-z))(1-ye^z)}$. Let $f(z) = \frac{-e^z}{1-e^z(1-z)}$. Thus, we need to compute $[z^n]\frac{f(z)}{1-ye^z}$.

Similarly, the first fraction can be rewritten as

$[z^n]\frac{1}{(1-e^z(1-z))(1-z-y)} = \frac{\frac{1}{1-z}}{(1-e^z(1-z)(1-\frac{y}{1-z})}$ (note that we divide by $1-z$ to make something of the form $\frac{1}{1-h(z)y}$ to make it similar to our second fraction).

Letting $g(z) = \frac{1}{(1-z)(1-e^z(1-z))}$, the first fraction reduces to $[z^n]\frac{g(z)}{1-\frac{y}{1-z}}$.

In both cases, we have to compute something of the form $[z^n]\frac{F(z)}{1-G(z)y}$ for some functions F and G. You can see that this is similar to $[x^k]\frac{1}{1-uf(x)}$ in our previous example.

Here, we have $G(z) = e^z$ and $\frac{1}{1-z}$. There is a subtle detail here which is that $[z^0]G(z) = 1 \neq 0$ in both cases, so G does not have a compositional inverse and we can't apply what we did just now directly. However, $G(z)-1$ does have a compositional inverse in both cases, so let $A(z) = e^z - 1$ and $B(z) = \frac{1}{1-z} - 1$. Thus, we need to compute $[z^n]\frac{f(z)}{1-(A(z)+1)y}$ and $[z^n]\frac{g(z)}{1-(B(z)+1)y}$.

Now we use the same approach as the previous example. Let $A^{-1}(z)$ denote the compositional inverse of $A(z)$ (you can find it explicitly by the definition of inverse). We let $H(z) = \frac{f(A^{-1}(z))}{1-(z+1)y}$, $G(z) = A(z)$ and $F(z) = A^{-1}(z)$ in Lagrange Inversion Formula. Then,

$[z^n]\frac{f(z)}{1-(A(z)+1)y} = [z^n]H(G(z)) = \frac{1}{n}[x^{-1}]H'(x)\frac{1}{F(x)^n} = \frac{1}{n}[x^{n-1}]\left(\frac{f(A^{-1}(x))}{1-(x+1)y}\right)'\frac{1}{\left(\frac{A^{-1}(x)}{x}\right)^n}$. Let $C(x) = f(A^{-1}(x))$ and $D(x) = \frac{1}{\left(\frac{A^{-1}(x)}{x}\right)^n}$. Take a moment to check that we can still compute the first n terms of $C(x)$ and $D(x)$ using polynomial operations in $O(n \log n)$ (find $A^{-1}(x)$ and substitute back into their definitions).

Now, our problem reduces to finding $\frac{1}{n}[x^{n-1}]\left(\frac{C(x)}{1-(x+1)y}\right)'D(x)$. Using quotient rule, we have (differentiating with respect to x),

$$\left(\frac{C(x)}{1-(x+1)y}\right)' = \frac{C'(x)[1-(x+1)y]-C(x)(-y)}{(1-(x+1)y)^2} = \frac{C'(x)}{1-(x+1)y} + \frac{C(x)y}{(1-(x+1)y)^2}.$$

Thus,

$$\frac{1}{n}[x^{n-1}]\left(\frac{C(x)}{1-(x+1)y}\right)'D(x) = \frac{1}{n}[x^{n-1}]\frac{C'(x)D(x)}{1-(x+1)y} + [x^{n-1}]\frac{C(x)D(x)y}{(1-(x+1)y)^2}.$$

We have two subproblems, finding $[x^n]\frac{P(x)}{1-(x+1)y}$ and $[x^n]\frac{P(x)y}{(1-(x+1)y)^2}$ for some computable functions P. Both can be dealt with using the geometric series formula. We have

$$[x^n]\frac{P(x)}{1-(x+1)y} = [x^n]P(x)\sum_{i\geq 0}(x+1)^i y^i$$

Let $P(x) = p_0 + p_1x + p_2x^2 + \ldots$ be the series expansion, then

$$[x^n]P(x)\sum_{i\geq 0}(x+1)^i y^i = \sum_{i\geq 0} y^i[x^n]\left(P(x)(x+1)^i\right) = \sum_{i\geq 0} y^i \sum_{j\geq 0} p_{n-j}\binom{i}{j}.$$

Hence, we need to compute $ans(i) = \sum_{j\geq 0} p_{n-j}\binom{i}{j} = i!\sum_{j\geq 0}\frac{p_{n-j}}{j!(i-j)!}$ fast. But this is almost the same thing as what we did in the Atcoder problem mentioned at the very beginning of this article! Define $E(x) = \sum_{i\geq 0}\frac{p_{n-i}}{i!}x^i$ and $F(x) = \sum_{i\geq 0}\frac{1}{i!}x^i$ for some large enough M. Then, our answer is the coefficient of x^i in $E(x)F(x)$. This part can be solved in $O(n \log n)$ time.

Finally, we need to compute $[x^n]\frac{P(x)y}{(1-(x+1)y)^2}$. With a similar expansion,

$$[x^n]\frac{P(x)y}{(1-(x+1)y)^2} = [x^n]P(x)y\sum_{i\geq 0}(i+1)(x+1)^i y^i$$

$$= \sum_{i\geq 0}(i+1)y^{i+1}[x^n]\left(P(x)(x+1)^i\right)$$

$$= \sum_{i\geq 0}(i+1)y^{i+1}[x^n]\left(P(x)(x+1)^i\right)$$

$$= \sum_{i\geq 0}(i+1)y^{i+1}\sum_{j\geq 0}p_{n-j}\binom{i}{j}.$$

Thus, we can compute this in the same way as before in $O(n\log n)$ using FFT.

Putting it altogether, we obtain a (albeit complicated) solution in $O(n\log n)$ time!

I hope this explanation makes Elegia's solution more intuitive to understand. :) Thanks to Elegia for the wonderful solution.

If you have any questions or spotted any errors, please tell me in the comments. There are probably more cool applications of generating functions in CP that I am not aware of, so feel free to share them in the comments too. :)