Counting Through the Looking Glass

Preface

This essay will use sigma notation in the following ways:

$$\sum_{k=s}^{n} a_k = a_s + a_{s+1} + a_{s+2} + \dots + a_{n-1} + a_n$$

$$\sum_{n \geq s} a_n = \sum_{n=s}^{\infty} a_n = a_s + a_{s+1} + a_{s+2} + \dots$$

$$\sum_{k+r=n} a_k b_r = \sum_{k=0}^{n} a_k b_{n-k} = a_0 b_n + a_1 b_{n-1} + \dots + a_{n-1} b_1 + a_n b_0$$

$$\sum_{kr=n} a_k b_r = \sum_{d|n} a_d b_{n/d} = \text{sum of } a_k b_r \text{ over all } k, r \in \mathbb{Z}^+ \text{ where } kr = n$$

In one section of the essay that does not involve summations of any kind, $\sum$ will also be used to denote an alphabet, or a set of letters. In this case $\sum$ is not to be confused with summation.

Sequences will be denoted as:

$$\{a_k\}_s^n = a_s,\ a_{s+1},\ a_{s+2},\ \dots,\ a_{n-1},\ a_n$$

$$\{a_{k+r}\}_s^n = a_{s+r},\ a_{s+r+1},\ a_{s+r+2},\ \dots,\ a_{n+r-1},\ a_{n+r}$$

$$\{a_n\}_s^{\infty} = a_s,\ a_{s+1},\ a_{s+2},\ \dots$$

Introduction

The maths we know and love today evolved from a need to count and quantify sets of items and measurements. Most people's first encounter with maths as a subject is when counting with their fingers, but later our intuition is built upon towards deeper and more abstract mathematical thinking. Personally, I find beauty in this abstract thinking when it allows me to solve creative problems, and also in the rare instances when I am able to make make connections between seemingly unrelated areas of maths, providing a deeper insight on the nature of these areas.

In this essay we will look at some problems that revolve around counting certain sets of items or arrangements. Many of these problems will involve abstractions or links to surprising fields, including set theory and calculus. The greatest common denominator is the use of generating functions as an intermediary between the counting setup of the problem and an algebraic framework that is used to analyse it. In his video on such a problem, Grant Sanderson (3blue1brown) says that "There's a time in your life before you understand generating functions, and a time after, and I can't think of anything that connects them other than a leap of faith". We will set out to make this leap, and to to, in turn, hopefully gain a deeper understanding of the type of combinatorics with which we are concerned.

1. Taking a Look at the Looking Glass

Our journey into the topic of generating functions begins when a friend asked me a simple question about counting arrangements of letters, or strings. A few weeks ago, during an outing, this friend asked me and another friend how many possible strings could be made from an alphabet of n different letters (here an alphabet is a set of possible letters which strings can contain). Yes, this is a true occurrence, maths students are very boring people. A string is a list of letters, so the question is about possible combinations of n letters. Since it is standard in such problems, we will say that the empty string, "", also counts as a possible string, and we will say that the length of a string is the number of letters it contains. Each of us jumped to different conclusions: I claimed there were 2^n different arrangements, and my two friends each claimed there were $n!$ and $1 + n + n^2 + \ldots$ different possible strings. We soon realised the ambiguity came from the question, which didn't specify how to distinguish between different strings, and so we each had different interpretations of how to count them.

The answer $1 + n + n^2 + \ldots$ came from the assumption that strings could have repeating letters and that the order of the letters mattered (strings made up of the same letters but in different orders are counted as different strings). If we let s_k be the number of possible strings of length k, then the idea is that there are ns_k possible strings of length $k+1$. Our recurrence relation comes from the fact that we can add any one of n letters onto the end of any sting of length k to form a new unique string of length $k+1$. Therefore $s_{k+1} = n \times s_k$. Although this is simply a geometric sequence, we will use it as an introduction to generating functions. Lets say we have a function $S(x)$

$$S(x) = \sum_{k \geq 0} s_k x^k$$

Here $S(x)$ is simply a never ending polynomial, or a power series, where the coefficient of each power of x is a term from the sequence $\{s_k\}_0^\infty$. We call $S(x)$ the (ordinary power series) generating function of $\{s_k\}_0^\infty$. In the words of Herbert Wilf, "A generating function is a clothesline on which we hang up a sequence of numbers for display", and that is how one is used here. While the jump from the sequence to this generating function may not seem intuitive, we will discover how, in more complicated problems, the ideas we use here are very useful for finding a solution or other interesting facts about the problem.

$$s_{k+1} = ns_k \qquad s_0 = 1$$

$$s_{k+1}x^{k+1} = nxs_kx^k \quad (k \geq 0)$$

$$\sum_{k \geq 1} s_k x^k = nx \sum_{k \geq 0} s_k x^k$$

$$S(x) - s_0 = nxS(x)$$

$$S(x) = \frac{1}{1 - nx}$$

The Taylor series expansion of $S(x) = 1 + nx + (nx)^2 + \ldots$ tells us that the coefficient of x^k, which is equal to s_k, is n^k. We can also see that the sum of the terms in the series $\{s_k\}_0^\infty$ is the sum of all the coefficients. Setting $x = 1$ leaves us with only a sum of coefficients, so the total number of possible strings is $S(1)$, or $1/(1-n)$.

But wait a second, this is suggesting that the number of ordered strings made up from an alphabet of n letters is somewhere in between -1 and 0? Clearly we made a mistake somewhere, and any A Level maths student can tell you that the mistake was when we substituted the value $x = 1$ into $S(x)$. Our generating function is a binomial expansion with a negative index, and these series are, and intuitively so, only convergent for $|nx| < 1$ or $|x| < \frac{1}{|n|}$. When using $S(x)$ as a generating function, we are setting it up and treating it as a formal power series, and the operation of evaluating at $x = 1$ "doesn't exist in the ring of formal power series" (Wilf 35). Much of the power of generating functions lies in the ability to analyse them over the domain of complex numbers, which we allow ourselves to do by finding a region within which the power series converges to an analytic function. This region $|x| < \frac{1}{|n|}$ for $S(x)$ is shown by the grey cylinder in Figure 1 below. Substituting complex numbers into generating functions is beyond the scope of this essay, but what it essentially allows us to do is sum over some select terms in a sequence.

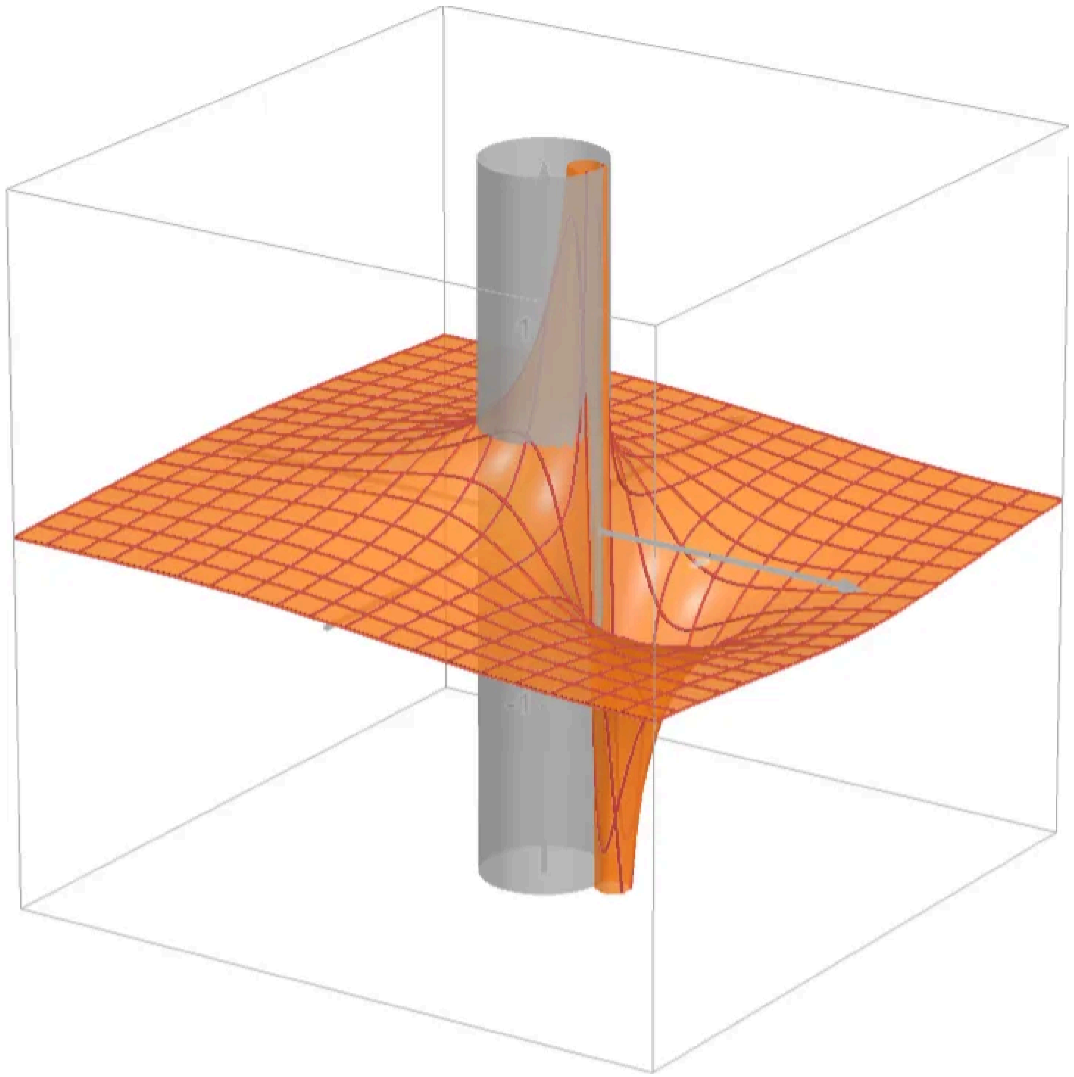


Figure 1: z∈ℂ on the xy-plane against the real component of S(z) on the z-axis, with a grey cylinder showing the domain of S(z). Interact with this graph at https://www.desmos.com/3d/5efsuya669, and turn on translucent mode to see through the grey cylinder

To get over the issue of convergence, we can place an upper limit l on the length of the possible strings. This allows $S(x)$ to converge at $x = 1$, as it would be a polynomial of degree l rather than a power series.

$$s_l = n^l \qquad s_m = 0 \text{ for all } m > l$$

$$S(x) - s_0 = nx(S(x) - s_l x^l)$$

$$S(x) = \frac{1 - (nx)^{l+1}}{1 - nx}$$

$$\sum_{k=0}^{l} s_k = S(1) = \frac{1 - n^{l+1}}{1 - n}$$

This is simply the formula for the sum of $l+1$ (the +1 is because we are including the string of length 0 in our count) terms in a geometric series, but the power of the method resides in the fact that it can be generalised and applied to much more difficult problems.

2. Down the Rabbit Hole

In between thinking about the question my friend asked and researching related areas of maths for this essay, the Alice in Wonderland theme stuck, in part for the feeling of chasing a rabbit or a problem into a wonderland of strange maths. Charles Dodgson, who wrote under the pen name Lewis Carroll, was a mathematician and logician who taught at Christ Church, Oxford, and many have interpreted his books Alice in Wonderland and Through the Looking Glass as being heavily related to and inspired by mathematics.

Alice in Wonderland is littered with allegories to new ideas at the time from logic and geometry, and also with critiques of the transition algebra was making at the time from concrete numbers to abstract representations. For example, Alice's interaction with the caterpillar is a parody of the first purely symbolic system of algebra (the algebra of generating functions is an example of a purely symbolic one) proposed by Augustus De Morgan. According to the New York Times, "Dodgson found the radical new math illogical and lacking in intellectual rigor". I find the metaphor of a rabbit to be apt because I believe there are curiosities waiting for mathematicians chase into rabbit holes all over the field, leading to strange new worlds. I also think the term "wonderland" accurately conveys the feeling of awe or wonder that this maths can inspire. Dodgson likely would have disagreed with the use of generating functions to solve counting problems, but hindsight tells us that the wonderland can more useful and beautiful than the riverbank Alice fell asleep on.

To resolve the ambiguity in his question, my friend rephrased it to ask how many ordered strings with no repeating letters could be made from a set of n letters. Here the term ordered refers to the fact that strings containing the same letters but in different orders are again counted as different strings.

We can solve by again, finding a recurrence relation, forming and simplifying a generating function, and analysing the result. First of all, we note that the number s_{k+1} of possible strings of length $k+1$ is equal to $(n-k)s_k$, since any unique string of length $k+1$ can be formed by adding a letter onto a string of length k. Since strings cannot have repeated letters, this letter must be one of the $n-k$ letters not already in our chosen string of length k.

Every pairing of:

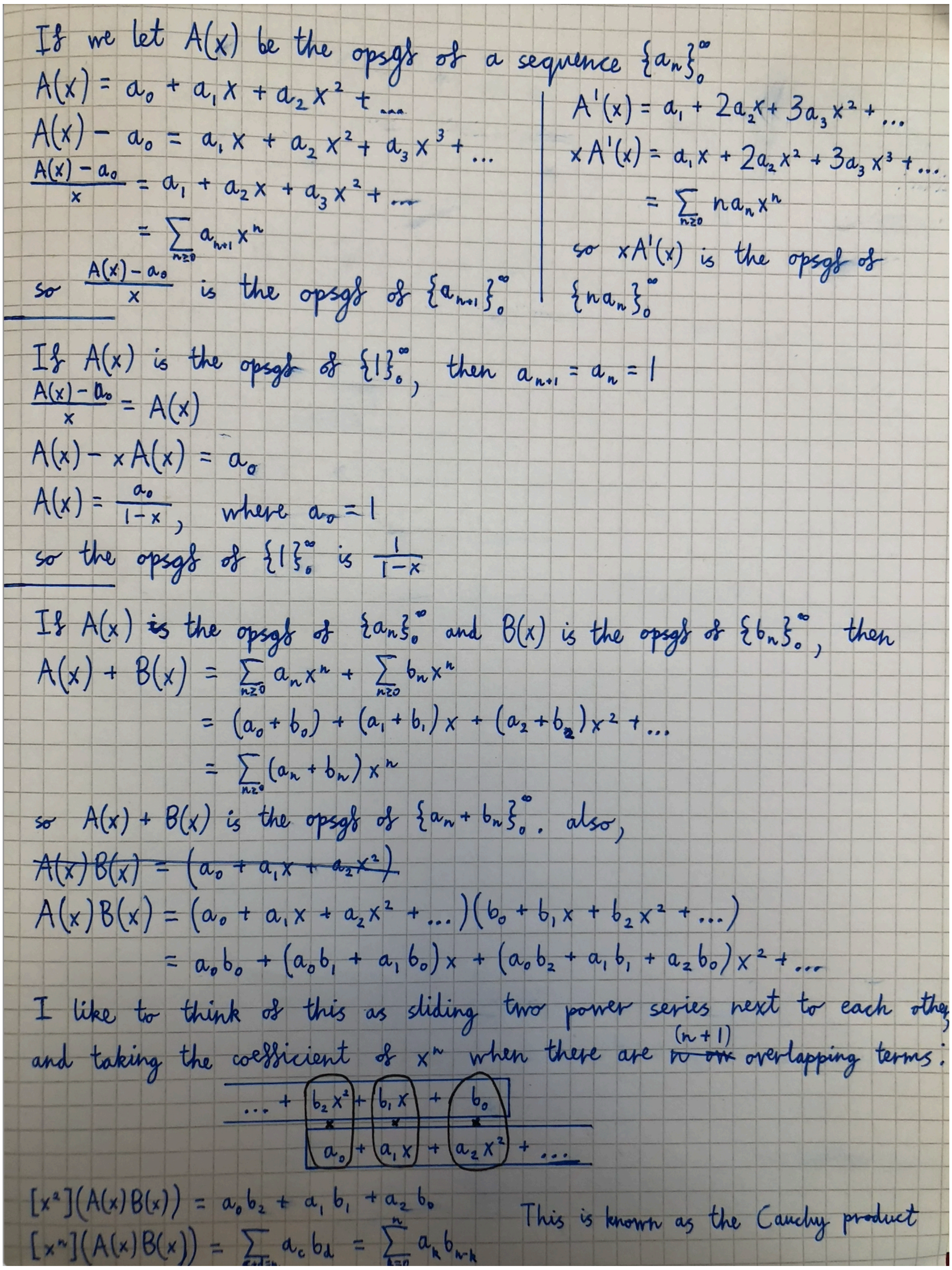
1. a string of length k from the s_k possible strings of this length
2. and a letter from the $n-k$ leftover letters

forms a unique string of length $k+1$. This means that s_{k+1} is equal to the number of pairings of the items from the above sets, which is by definition the product of the number of items in each set, or $s_k \times (n-k)$.

$$s_{k+1} = ns_k - ks_k \qquad s_0 = 1$$

$$S(x) = \sum_{k\geq 0} s_k x^k$$

Before converting our recurrence relation to an equation involving its ordinary power series generating function (opsgf), we will look at some general ways to manipulate these power series. To do so, we will introduce the notation that $[x^n]A(x)$ is equal to the coefficient of x^n in $A(x)$, where $A(x)$ is the opsgf of some sequence $\{a_n\}_0^\infty$. Hence $[x^n]A(x) = a_n$, and $[cx^n]A(x) = a_n/c$.



If we let $A(x)$ be the opsgf of a sequence $\{a_n\}_0^\infty$

$A(x) = a_0 + a_1 x + a_2 x^2 + \dots$

$A(x) - a_0 = a_1 x + a_2 x^2 + a_3 x^3 + \dots$

$\frac{A(x) - a_0}{x} = a_1 + a_2 x + a_3 x^2 + \dots$

$= \sum_{n \geq 0} a_{n+1} x^n$

so $\frac{A(x) - a_0}{x}$ is the opsgf of $\{a_{n+1}\}_0^\infty$

$A'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \dots$

$xA'(x) = a_1 x + 2a_2 x^2 + 3a_3 x^3 + \dots$

$= \sum_{n \geq 0} n a_n x^n$

so $xA'(x)$ is the opsgf of $\{na_n\}_0^\infty$

If $A(x)$ is the opsgf of $\{1\}_0^\infty$, then $a_{n+1} = a_n = 1$

$\frac{A(x) - a_0}{x} = A(x)$

$A(x) - xA(x) = a_0$

$A(x) = \frac{a_0}{1-x}$, where $a_0 = 1$

so the opsgf of $\{1\}_0^\infty$ is $\frac{1}{1-x}$

If $A(x)$ ~~is~~ the opsgf of $\{a_n\}_0^\infty$ and $B(x)$ is the opsgf of $\{b_n\}_0^\infty$, then

$A(x) + B(x) = \sum_{n \geq 0} a_n x^n + \sum_{n \geq 0} b_n x^n$

$= (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + \dots$

$= \sum_{n \geq 0} (a_n + b_n) x^n$

so $A(x) + B(x)$ is the opsgf of $\{a_n + b_n\}_0^\infty$. also,

~~$A(x)B(x) = (a_0 + a_1 x + a_2 x^2)$~~

$A(x)B(x) = (a_0 + a_1 x + a_2 x^2 + \dots)(b_0 + b_1 x + b_2 x^2 + \dots)$

$= a_0 b_0 + (a_0 b_1 + a_1 b_0)x + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 + \dots$

I like to think of this as sliding two power series next to each other and taking the coefficient of x^n when there are (n+1) ~~no on~~ overlapping terms:

$\dots + b_2 x^2 + b_1 x + b_0$

$a_0 + a_1 x + a_2 x^2 + \dots$

$[x^2](A(x)B(x)) = a_0 b_2 + a_1 b_1 + a_2 b_0$

$[x^n](A(x)B(x)) = \sum_{c+d=n} a_c b_d = \sum_{k=0}^{n} a_k b_{n-k}$

This is known as the Cauchy product

Figure 2: some ways to manipulate ordinary power series generating functions

Some of the key ways shown in Figure 2 to manipulate ordinary power series generating functions are written below:

- The opsgf of $\{1\}_0^\infty$ is $1/(1-x)$
- If $A(x)$ is the opsgf of $\{a_n\}_0^\infty$, then the opsgf of $\{a_{n+1}\}_0^\infty$ is given by $(A(x) - a_0)/x$
- If $A(x)$ is the opsgf of $\{a_n\}_0^\infty$ and $B(x)$ is the opsgf of $\{b_n\}_0^\infty$, then $A(x) + B(x)$ is the opsgf of $\{a_n + b_n\}_0^\infty$
- If $A(x)$ is the opsgf of $\{a_n\}_0^\infty$, then $xA'(x)$ is the opsgf of $\{na_n\}_0^\infty$

- If $A(x)$ is the opsgf of $\{a_n\}_0^\infty$ and $B(x)$ is the opsgf of $\{b_n\}_0^\infty$, then $A(x)B(x)$ is the opsgf of $\sum_{c+d=n} a_c b_d$. In other words, $[x^n](A(x)B(x)) = \sum_{c=0}^{n} a_c b_{n-c}$

The above methods give us the following differential equation for $S(x)$:

$$\frac{S(x) - s_0}{x} = nS(x) - xS'(x)$$

The reader is welcome to try and find $S(1)$ by solving this differential equation, but it is actually quite complicated and results in a messy answer, so we will instead try a new kind of generating function - an exponential generating function.

Exponential generating functions (egfs) are of the form $A(x) = \sum_{n\geq 0} a_n \frac{x^n}{n!}$, where $A(x)$ is the egf of $\{a_n\}_0^\infty$. They are useful for many of the same reasons as opsgfs, but are more useful in certain counting problems than opsgfs and less useful in others, for reasons we will soon cover.

If $A(x)$ is the egf of $\{a_n\}_0^\infty$, then

$$A(x) = \frac{a_0}{0!} + \frac{a_1 x}{1!} + \frac{a_2 x^2}{2!} + ...$$

$$A'(x) = \frac{a_1}{1!} + \frac{2a_2 x}{2!} + \frac{3a_3 x^2}{3!} + ...$$

$$A'(x) = \frac{a_1}{0!} + \frac{a_2 x}{1!} + \frac{a_3 x^2}{2!} + ...$$

which means that $A'(x)$ is the egf of $\{a_{n+1}\}_0^\infty$.

I believe it is easier to understand maths when you work through some of it yourself, so I leave it as an exercise to the reader to derive the remaining ways to manipulate exponential generating functions written below:

- The egf of $\{1\}_0^\infty$ is e^x
- If $A(x)$ is the egf of $\{a_n\}_0^\infty$, then the egf of $\{a_{n+1}\}_0^\infty$ is given by $A'(x)$
- If $A(x)$ is the egf of $\{a_n\}_0^\infty$ and $B(x)$ is the egf of $\{b_n\}_0^\infty$, then $A(x) + B(x)$ is the egf of $\{a_n + b_n\}_0^\infty$
- If $A(x)$ is the egf of $\{a_n\}_0^\infty$, then $xA'(x)$ is the opsgf of $\{na_n\}_0^\infty$
- If $A(x)$ is the egf of $\{a_n\}_0^\infty$ and $B(x)$ is the egf of $\{b_n\}_0^\infty$, then $A(x)B(x)$ is the egf of $\sum_{c+d=n} \frac{n!}{c!d!} a_c b_d$. In other words, $[x^n/n!](A(x)B(x)) = \sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k}$

I have a proof of each of these facts I promise, but they are too large to fit in the margin of this essay.

The differences between these methods and the ones for opsgfs, in particular the difference between the product rules for both, are the key factor that make egfs more useful for certain problems and vice versa. Both are used when "stitching together" the structures counted by the sequences being convolved, but the egf product rule is more useful when the objects in the structures (in this case the letters in the strings) are relabelled, as there will be $\binom{n}{k}$ relabelling's, and this is represented by the egf product rule. These rules and the definition of egfs allow them to more easily "soak up" factorials and binomial coefficients, and in general make them more useful, as we will see, for counting labelled or ordered structures.

$$S(x) = \sum_{k \geq 0} s_k \frac{x^k}{k!}$$

$$s_{k+1} = ns_k - ks_k \qquad s_0 = 1$$

$$S'(x) = nS(x) - xS'(x)$$

$$(1+x)S'(x) = nS(x)$$

$$\frac{S'(x)}{S(x)} = \frac{n}{1+x}$$

$$\int \frac{1}{S(x)} \frac{dS(x)}{dx} dx = \int \frac{n}{1+x} dx$$

$$\ln(S(x)) = n \ln(1+x) + C$$

$$S(x) = A(1+x)^n$$

$$S(0) = s_0 = A = 1$$

$$[x^k]S(x) = \binom{n}{k}$$

$$s_k = \left[\frac{x^k}{k!}\right] S(x) = \binom{n}{k} k! = \frac{n!}{(n-k)!}$$

We have reached a closed form solution for s_k that was already apparent from the definition of s_k if we had prior understanding of permutations. However, this solution was reached using no knowledge other than a recurrence relation, and the generating function tells us something important about the counting setup itself. First, we will look at the sum of s_k over k, which is the number of strings we are looking for.

$$\sum_{k \geq 0} s_k = \sum_{k=0}^{n} \frac{n!}{(n-k)!} = n! \sum_{k=0}^{n} \frac{1}{k!}$$

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

$$\sum_{k \geq 0} s_k = n! \left(e - \sum_{k > n} \frac{1}{k!} \right)$$

$$\sum_{k > n} \frac{1}{k!} = \frac{1}{(n+1)(n)...(1)} + \frac{1}{(n+2)(n+1)...(1)} + ... < \sum_{k \geq 1} \left(\frac{1}{n}\right)^k$$

$$n!e - \sum_{k \geq 0} s_k < \frac{1}{1 - 1/n} - 1 = \frac{1}{n-1}$$

$$\sum_{k \geq 0} s_k \approx n!e$$

We have shown that $n!e$ rounds down to the number of ordered strings with no repetitions made up from n possible letters when $n > 1$, and we can check at $n = 1$ and $n = 0$ to find that $\sum_{k \geq 0} s_k = \lfloor n!e \rfloor$ for all $n > 0$.

The guest appearance of e makes for a very surprising approximation to a seemingly simple question, but due to e^x's role as the egf of $\{1\}_0^\infty$, e actually shows up quite often in problems involving order dependant structures and egfs, especially in approximations for these.

3. The Madhatter, the Dormouse and the March Hare

Now we will take a look at the third interpretation of the question, the one that led to the answer 2^n. With all three generating functions before us, we can compare them and try to gain some insight on why generating functions are so effective.

If we assume that the order of strings does not matter (i.e. strings made up of the same letters but in different orders are counted as the same string) and strings cannot have repeating letters, then we can let $s(n, k)$ be the number of such strings. Here k is the length of each string counted by $s(n, k)$, and n is the number of letters in the alphabet.

We can separate the strings counted by $s(n, k)$ into two piles:

1. one in which each string contains the last letter in the alphabet (or the nth letter), which we will label as "n"
2. another pile where none of the strings contain "n"

We find that each string in the first pile can be made by adding the letter "n" to a string of length $k - 1$ which does not contain the letter "n". All of these strings are counted by $s(n - 1, k - 1)$, so there are $s(n - 1, k - 1)$ strings in the first pile.

Each string in the second pile is of length k and can contain any letter other than "n", so the strings in the second pile are counted by $s(n - 1, k)$.

Our resulting recurrence is $s(n, k) = s(n - 1, k - 1) + s(n - 1, k)$, which we note describes the recurrence relationship within the pascal triangle (shown in Figure 3 below). We also know that $s(n, 0) = 1$ since the empty string is counted, and $s(n, -1) = s(n, n + 1) = 0$, as strings cannot contain less than 0 or more than n letters.

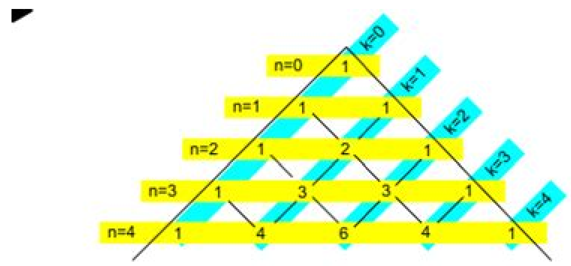


Figure 3: A google image showing the recurrence relation forming pascal's triangle

$$S_n(x) = \sum_{k \geq 0} s(n, k) x^k \qquad S_0(x) = 1$$

$$S_n(x) = x S_{n-1}(x) + S_{n-1}(x)$$

$$S_n(x) = (1 + x) S_{n-1}(x)$$

$$S_n(x) = (1 + x)^n$$

If this generating function looks familiar, its because this is the generating function of the second interpretation of my friend's question, although critically that was an egf and this is an opsgf. We can find $s(n,k) = [x^k]S_n(x)$ using the Taylor expansion of $S_n(x)$.

$$[x^k]S_n(x) = \frac{S^{(k)}(0)}{k!}, \text{ where } S^{(k)}(x) \text{ is the } k\text{th derivative of } S(x)$$

$$S^{(k)}(x) = n(n-1)...(n-k+1)(1+x)^{n-k} = \frac{n!}{(n-k)!}(1+x)^{n-k}$$

$$[x^k]S_n(x) = \frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Because this is an opsgf, we can find the sum of the coefficients in $S_n(x)$, or the number of unordered strings with no repetitions made up from n letters, as $S_n(1)$.

$$\sum_{k\geq 0} s(n,k) = \sum_{k=0}^{n} \binom{n}{k} = S_n(1) = (1+1)^n = 2^n$$

We can also reach the answer 2^n by choosing whether or not to include each of our n letters in a string. A different string is formed from each unordered combination of these letters, which means that the combination of each choice to either include or not include a letter uniquely determines a string. There are n of these choices, each with 2 options, giving us $2 \times 2 \times ... \times 2$, or 2^n possible strings.

The reason why the two interpretations of the question, where in both, strings cannot have repeated letters, but where in the first, order does matter, and where in the second it does not, have such the same generating functions (but one is an egf and the other an opsgf) is because the similarity in the counting setups is maintained when converting both to generating functions. The first generating function is the egf of $\{k!\binom{n}{k}\}_0^\infty$, and the second is the opsgf of $\{\binom{n}{k}\}_0^\infty$. This is because we can say that the number of ordered strings s_k of length k is equal to the number of permutations of the unordered strings s'_k of length k. Therefore $s_k = k!s'_k$, and the use of an egf allows us to deal with and manipulate the $k!$ term algebraically. In general, the different types of generating functions encode sequences in different ways that can be analysed and also counted, with properties like the $x^n/n!$ in exponential generating function encoding information such as structure being ordered. Generating functions form a language that unifies different problems in combinatorics by representing them in a common framework while maintaining their key properties, which is what makes them such a powerful tool.

6. A Mad Tea-Party

To quote the New York Times, "In the mid-19th century, mathematics was rapidly blossoming into what it is today: a finely honed language for describing the conceptual relations between things ... In 'Alice' [Dodgson] attacked some of the new ideas as nonsense using ... reductio ad absurdum". Reductio ad absurdum is a technique where an idea is pushed to its logical extreme to find a contradiction or absurd result. We will do the same to our three counting setups by considering what happens as the sets we are looking at become infinitely large.

In our first counting setup, where the order of strings matters and we can have repeated letters from an alphabet of n letters, we let strings have a maximum length of l in order to get a convergent count for the number of possible strings. If we let l approach ∞, we essentially remove this maximum length, which tells us that

$$\frac{1}{1-n} = \sum_{k\geq 0} n^k = \lim_{l\to\infty}\left(\frac{1-n^{l+1}}{1-n}\right)$$

$$\lim_{l\to\infty}(1-n^{l+1}) = 1$$

Which is true if and only if $|n| < 1$, showing that $1/(1-n)$ is only convergent in this domain. To more carefully discuss the infinite possible strings of infinite possible lengths, we can talk about the set $\mathcal{S}$ of such strings, and look at the number of elements it contains, or its cardinality, $|\mathcal{S}|$. We note that strings in $\mathcal{S}$ are of finite length.

We can organise $\mathcal{S}$ like a dictionary by partitioning it into n pairwise disjoint subsets (split it into n smaller sets which contain no repeated elements) whose union is $\mathcal{S}$, and label each with a different one of the n letters from our alphabet. We can then put each of our strings into the subset that corresponds to its first letter, and since we already know its first letter from the label of its subset, we can remove this first letter. This thought experiment was proposed by Dr Ian Stewart in his book From Here to Infinity using the example of a dictionary called the Hyperwebster as the set $\mathcal{S}$, and is a thought-provoking demonstration of the strangeness of infinite sets. The concept is an analogy for the Banach-Tarski paradox, which is a strange problem related to splitting a spheres into 2 identical copies of itself.

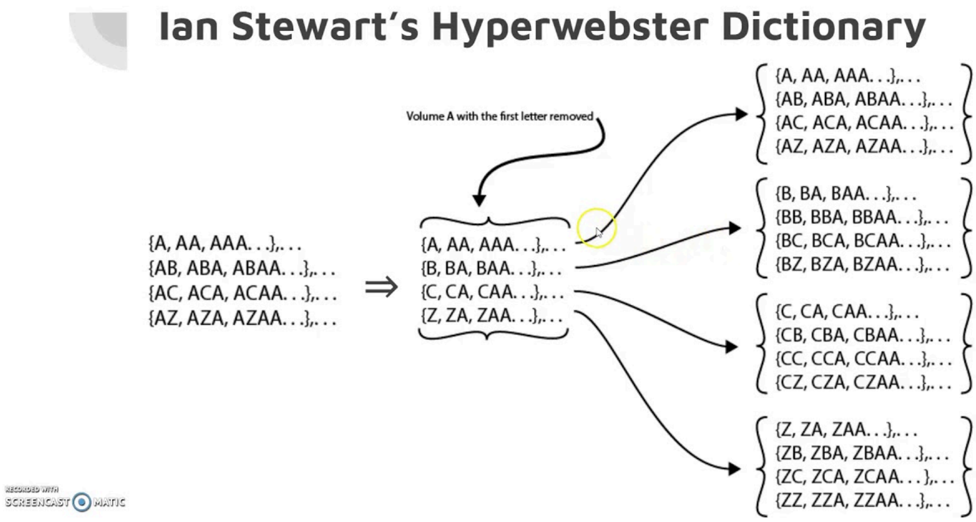


Figure 4: A google image showing what the subset labelled "A" in the Hyperwebster might look like

We can find the cardinality of $\mathcal{S}$ by comparing $\mathcal{S}$ with the the set of natural numbers $\mathbb{N}$. By laying every possible string out in ascending order of length (so the empty string will be laid out, followed by all strings containing exactly 1 letter, all strings of containing exactly 2 letters, and so on), with strings of the same length being arranged in alphabetical order, we can label each with a natural number based on its position in our ordering. This ordering can be used to map any string in $\mathcal{S}$ to a specific natural number, and the reverse can be applied (we can find the string at a certain position) to find the corresponding string for any natural number. Thus every string can be mapped to a unique natural number, and all natural numbers have been mapped to, so we have found a "bijection" between the two sets, showing that $|\mathcal{S}| = |\mathbb{N}|$. We will call $|\mathbb{N}|$ countable infinity, since elements in sets with this cardinality can be counted.

Essentially, bijections are one-to-one mappings between elements in two sets, often using an abstract kind of function that goes from elements in one set to elements in another, with an inverse that can go back from the second set to the first. Georg Cantor, the founder of modern set theory, posited that if we can connect each element in the first set to a unique element in the second, with all elements in both sets being involved, then the two sets must have the same cardinality.

Now we will instead consider what happens as n approaches ∞. n is the number of possible letters which strings can draw from, or the cardinality $|\sum|$ of our alphabet $\sum$. So, to make n infinitely large we can label each letter in $\sum$ with a positive integer, making n countably infinite. In our third counting setup, where strings are unordered and cannot have repeated letters, we can allow strings to be infinitely long and count them using our generating function $(1+x)^n$, which gives us a count of 2^n. This comes from the idea that strings are essentially different subsets of $\sum$, which means that $\mathcal{S}$ is the set of subsets of $\sum$, also referred to as the power set $\mathcal{P}(\sum)$ of $\sum$. Each combination of choices of whether or not to include each letter in $\sum$ encodes a unique string in $\mathcal{S}$. Since in each choice we have two options to either include or not include a certain letter, we have 2^n possible strings. This relationship still holds when the set of letters $\sum$ is countably infinitely large, since the bijection between strings in $\mathcal{S}$ and a combination of n binary choices (choices with two options) can still be formed. We can represent our combination of binary choices as an infinite string of binary digits (or bits) by denoting the choice to include a letter or not to include a letter as "1" or "0" respectively, and concatenating all our choices together. For example, "10011..." is formed by choosing to include the first letter, the fourth, the fifth, and so on. This approach is only possible when our strings can be infinitely long, since the power set $\mathcal{P}(\sum)$ of an infinite set will contain infinitely large subsets.

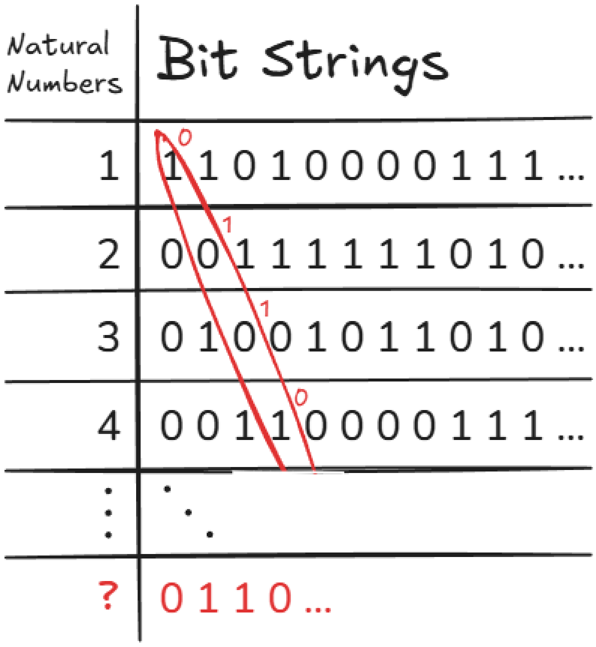


Figure 5: A visualization of Cantor's Diagonal Argument

We can now prove by contradiction that $|S|$ is uncountably infinite by attempting to form a bijection between $\mathcal{S}$ and $\mathbb{N}$. If we place our binary sequence representations of the strings in any order, and assign a natural number to each based on this order (as shown in the table in Figure 3), then all the binary sequences must have a corresponding natural number for our bijection to hold. However, if we take the ith bit of each binary sequence, where i is the natural number assigned to that sequence, and place the complement of our bit (the complement of "1" is "0", and the complement of "0" is "1") as the ith element of a new binary sequence, then the sequence formed by doing this operation over all i will differ from each sequence in our bijection by at least one bit. This means that this sequence cannot occur anywhere in the list of sequences associated with a natural number, and it therefore cannot have a corresponding natural number. Since this process, also shown in Figure 3, can be done for any ordering of the binary sequences, and thus for any bijection between these two sets, $|\mathcal{S}|$ must be strictly greater than $|\mathbb{N}|$. This proof is known as Cantor's Diagonal Argument, and was used to prove the following for any set X:

$$\mathcal{P}(X) = 2^{|X|} > |X|$$

This proves that there are multiple infinities of different sizes, such as uncountable infinity, which is given by $|\mathcal{P}(\mathbb{N})|$. This idea was met with great controversy at the time, but had awesome (by which I mean awe-inspiring, though also I'd say cool) consequences.

If $\mathcal{S}$ is the set of all ordered infinite strings which can contain repeating letters from a countable (finite or infinite) alphabet $\sum$, then we can use the same diagonalization argument to show that $\mathcal{S}$ is uncountably infinite. Just as before we have infinite strings, with letters this time represented as digits in base n rather than as bits in a bit string.

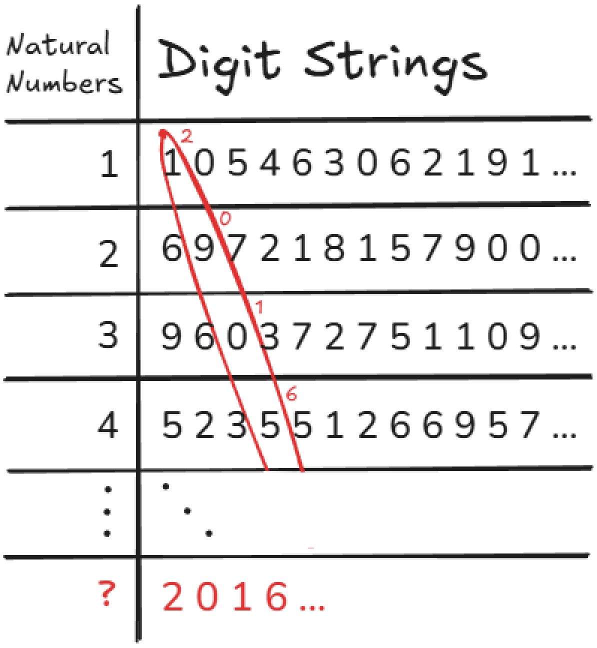


Figure 6: A visualization of Cantor's diagonal argument for digit strings in base 10. I foolishly deleted the last visualization from my library after exporting it so had to make this one from scratch, hence any inconsistencies

Notably, real numbers in the interval $[0, 1]$ (between 1 and 0) are essentially just infinite strings from the alphabet $\{0, 1, 2, ..., 9\}$ (the digits base 10) after the decimal point. All real numbers can be thought of as a pairing of a natural number before the decimal point and such a string after the decimal point, so there are $|\mathbb{N}| \times |\mathcal{P}(\mathbb{N})|$ real numbers. Multiplications between infinite quantities give the largest infinity, so there are $|\mathcal{P}(\mathbb{N})|$, or uncountably infinite, real numbers.

When our alphabet $\sum$ is a countably infinite set $\{a_0, a_1, a_2, \dots\}$, we can denote strings as sequences of natural numbers, including 0. To do this, we replace each letter a_k in each string in $\mathcal{S}$ with the natural number k. By converting each natural number in our sequence to base 9, and concatenating these natural numbers together into a string, we end up with a base-9 digit string. To avoid sequences like 1, 11, 111, ... being mapped to the same digit string as sequences like 111, 11, 1..., we can use the number 9 as a delimiter between natural numbers in our sequence (so we glue natural numbers base 9 together with the digit 9 in between consecutive numbers in the sequence). In doing so, we have mapped each string in $\mathcal{S}$ into a unique string from alphabet $\{0, 1, 2, \dots, 9\}$. The letter a_0 will be mapped to the natural number 0, which when concatenating into a string from the base 10 alphabet, will be represented as an empty string. So “$a_0, a_{13}, a_{21}, a_0, a_5, ...$” will become “914923995...”. All strings from the finite alphabet can be mapped back to their corresponding unique string from $\mathcal{S}$ by converting substrings between 9’s into natural numbers base 9 and finding the letter in $\sum$ corresponding to this natural number. So for example, “991791921...” becomes “$a_0, a_0, a_{16}, a_1, a_{19}, ...$”. Therefore we have found a bijection between the set of ordered infinite strings with repeating letters that come from a countably infinite alphabet, and those that come from the finite alphabet $\{0, 1, 2, ..., 9\}$. We now know both are uncountably infinite.

But if our strings, still ordered, infinitely long, and made up from a countably infinite alphabet $\sum$, are not allowed to contain repeating letters, how do we count them? We can find a bijection from the set $\mathcal{T}$ of strings with repeating letters to the string $\mathcal{S}$ without repeating letters. Both are made up from a common alphabet $\sum$, so we can represent strings from both as sequences of natural numbers $\{a_k\}_1^\infty$ and $\{b_k\}_1^\infty$ respectively, where$\{b_k\}_1^\infty$ is the one that doesn’t contain repeated numbers. Taking a string $a_1, a_2, a_3, \ldots$, we can form a string $b_1, b_2, b_3, \ldots$ by letting b_k be the a_kth smallest number not already in the set $\{b_1, b_2, b_3, \ldots, b_{k-1}\}$ (the set of letters in the string that come before the kth letter). So if $\{a_k\}_1^\infty = 2, 11, 10, 11 \ldots$, then b_1 is the a_1th, or 2nd, natural number not already in the sequence b. This is simply 2, so $b_1 = 2$. b_2 will be the 11th number not already in b which means $b_2 = 12$. b_3 is the 10th number not in b. The set of numbers currently in b is $\{2, 12\}$, so $b_3 = 11$. b_4 is the 11th number not already in b, which currently contains 2, 12 and 11, so $b_4 = 14$. Therefore $\{b_k\}_1^\infty = 2, 12, 11, 14...$. We know this procedure is a bijection because we can apply it to any string in $\mathcal{T}$ to get exactly one string in $\mathcal{S}$, and there is an inverse of the function or procedure that can be applied to any string in $\mathcal{S}$ that clearly gives back the string that was mapped to it from $\mathcal{T}$, and only that string. The inverse of the procedure is to, for each k from 1 to ∞ inclusive, set a_k as the cardinality of the set $\{1, 2, 3, ..., b_k\}$ after taking away from it all numbers also contained in the set $\{b_1, b_2, b_3, ..., b_{k-1}\}$.

Therefore, when strings drawing from a countable alphabet can be of infinite length, no matter the counting setup, there will be uncountably infinite such strings. When strings are finite, they can always be ordered in ascending lengths and then alphabetically, meaning that they will be countable. Our generating functions seem to produce absurd results at these strange extremes of the counting problems they originated from, but this is only because the algebra of infinite cardinals is itself very strange. For example, the generating function of the first counting setup is $1/(1 - nx)$. This indicates that strings of infinite length can be counted by $[x^{\aleph_0}]\frac{1}{1-nx} = n^{\aleph_0}$. Cantor’s diagonal argument was used to prove that $n^{\aleph_0} = 2^{\aleph_0}$ and that the resultant quantity is uncountably infinite. We have shown this to be true, and found using another kind abstract functions, bijections, that it remains true even when $n = \aleph_0$. $2^{\aleph_0}$ is also the sum of coefficients in the generating function for our third counting setup, and for our second counting setup, this value is $n!e$. The ideas that $n^n = n!e = 2^n$ when n becomes infinitely large, that sets of strings can contain copies of themselves, and that some infinities are larger than others seems like something out of wonderland. Yet these curious and absurd ideas, which can be reached simply by counting strings of different kinds, now lay the foundation for the branch of maths known as set theory. David Hilbert said that “No one will drive us from the paradise which Cantor created for us”, and over the years, the language of set theory has indeed become that paradise - one within which the much of the rest of maths is now described and formalized.

4. Splitting up the Playing Cards

Having fully explored the problem my friend posed, it’s time to use the method of generating functions to solve a more difficult (and in my opinion, beautiful) problem. This problem is about counting the number of partitions of a set. The idea is that: given a set S of n elements, a k-partition of this set is the number of ways to split it up into k pairwise disjoint subsets whose union is S. Essentially, this is the number of ways to split S into k classes, where the order of the elements in each class and the order of the classes within S is of no importance: the partition $\{1, 2\}\{3, 4\}$ is the same as the partition $\{4, 3\}\{2, 1\}$. Since the elements within S themselves don't matter, we will let S be the set of natural numbers between 1 and n inclusive. We will denote this set as $[n]$.

For example, all the 2-partitions of $[4]$ are: $\{1,2\}\{3,4\}$, $\{1,3\}\{2,4\}$, $\{1,4\}\{2,3\}$, $\{1,2,3\}\{4\}$, $\{1,2,4\}\{3\}$, $\{1,3,4\}\{2\}$, and $\{1\}\{2,3,4\}$. We will denote the number of k-partitions of $[n]$ as $\left\{ {n \atop k} \right\}$. These are also known as Stirling numbers of the second kind. Our problem is to count all partitions $b(n)$ of $[n]$, or all ways to split $[n]$ into pairwise disjoint subsets whose union is $[n]$.

$$b(n) = \sum_{k \geq 0} \left\{ {n \atop k} \right\}$$

The numbers $\{b(n)\}_0^\infty$ are known as the Bell numbers. We will begin trying to find these numbers by finding a recurrence relation in the Stirling numbers of the second kind $\left\{ {n \atop k} \right\}$ as we did for the binomial coefficients.

Like with the binomial coefficients, we can split the k-partitions of $[n]$ into two piles: one in which the element n is in its own class by itself in every partition, and one where n is in a class with other elements in every partition.

In the first pile, removing the element n from each partition would be the same as removing a full class, since n is the sole occupant of one of the classes in each partition. The remaining classes in each partition contain numbers from 1 to $n-1$ inclusive, and there are $k-1$ classes left in each partition. It is clear to see how each partition in this first pile is a partition from $\left\{ {n-1 \atop k-1} \right\}$ with a class $\{n\}$ added on. Therefore there are $\left\{ {n-1 \atop k-1} \right\}$ partitions in this first pile.

Removing n from each partition in the second pile again leaves us with partitions of $[n-1]$, however this time there are k classes in each partition, since n was not alone in a class, so removing it didn't change the number of classes in each partition. Thus removing n from partitions in the second pile leaves us with k-partitions of $[n-1]$, however, there are k copies of each partition. This is because adding n onto each of k different classes in each k-partition of $[n-1]$ creates a new, unique k-partition of $[n]$. Therefore, this second pile contains $k\left\{ {n-1 \atop k} \right\}$ partitions.

The method we use to count the number of terms in each pile as Stirling numbers is similar to how we formed bijections in the previous chapter, and it has given us the recurrence relation

$$\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k\left\{ {n-1 \atop k} \right\}$$

To create a base case for the recurrence, we will claim that $\left\{ {0 \atop 0} \right\} = 1$, and $\left\{ {n \atop 0} \right\} = 0$ when $n \neq 0$, since you cannot partition a non-empty string into 0 classes. You also can't partition a set of n elements into less than 0 or more than n classes, and you can't have a set with a negative number of elements, so $\left\{ {n \atop k} \right\} = 0$ when $n < 0$, $k < 0$, or $k > n$.

Now we can try to look for a generating function for our Stirling numbers the same way as we did for the binomial coefficients, and then use the rules for opsgfs to find an expression for the generating function.

$$\begin{aligned} A_n(x) &= \sum_{k \geq 0} \left\{ {n \atop k} \right\} x^k \qquad A_0(x) = 1 \\ &= xA_{n-1}(x) + xA'_{n-1}(x) \end{aligned}$$

However, the presence of the three different functions $B_n(x)$, $B_{n+1}(x)$, and $B'_n(x)$ in this equation makes it very difficult to find the generating function $B_n(x)$, so we will try a different approach. Using the egf instead would still leave us with these three different functions in our equation for $B_n(x)$, so we can try to sum the Stirling numbers over n rather than over k using a different generating function $B_k(x)$.

$$\begin{aligned}
B_k(x) &= \sum_{n\geq 0} \left\{ {n \atop k} \right\} x^n \qquad B_0(x) = 1 \\
&= xB_{k-1}(x) + kxB_k(x) \\
&= \frac{x}{1-kx} B_{k-1}(x) \\
&= \frac{x}{1-kx} \cdot \frac{x}{1-(k-1)x} \cdot \frac{x}{1-(k-2)x} \cdot ... \cdot \frac{x}{1-1x} B_0(x) \\
&= \frac{x^k}{(1-x)(1-2x)(1-3x)...(1-kx)}
\end{aligned}$$

An expression for $\left\{ {n \atop k} \right\}$ can be found using partial fraction decomposition, which is a method that is very common in problems involving generating functions. The idea is that fractions like $\frac{1}{ab}$ can be expressed as sums of fractions like $\frac{\alpha}{a} + \frac{\beta}{b}$ whose denominators multiply to give the original fraction. Interestingly, this same idea is this basis of what makes Dirichlet series such as the Riemann zeta function so powerful. These generating functions are used in number theory to study, among other things, properties of primes.

A simple example of partial fractional decomposition would be $\frac{1}{x(x-1)} = \frac{\alpha}{x} + \frac{\beta}{x+1}$. We can multiply both sides by $x(x+1)$ and compare coefficients of x to find α and β. $\alpha(x+1) + \beta x = 1$. It follows that $\alpha = 1$ and $\beta = -1$. In our case, we can apply the partial fraction expansion

$$\frac{1}{(1-x)(1-2x)...(1-kx)} = \sum_{j=0}^{k} \frac{\alpha_j}{1-jx}$$

We can apply a clever trick here by multiplying both sides by $1 - rx$, where r is some integer $1 \leq r \leq k$.

$$\frac{1}{(1-x)(1-2x)...(1-(r-1)x)(1-(r+1)x)...(1-kx)} = \alpha_r + (1-rx)\left(\sum_{j=0}^{r-1} \frac{\alpha_j}{1-jx} + \sum_{j=r+1}^{k} \frac{\alpha_j}{1-jx} \right)$$

If we let $x = 1/r$, then

$$\begin{aligned}
\alpha_r &= \frac{1}{(1-1/r)(1-2/r)...(1-(r-1)/r)(1-(r+1)/r)...(1-k/r)} \\
&= r^{k-1} \cdot \frac{1}{(r-1)(r-2)...(r-(r-1))} \cdot \frac{1}{(r-(r+1))...(r-k)} \\
&= (-1)^{k-r} \frac{r^{k-1}}{(r-1)!(k-r)!}
\end{aligned}$$

We recall that $[cx^n]A(x)$, where $A(x)$ is the opsgf of $\{a_n\}_0^\infty$, gives us a_n/c

$$\begin{aligned}
\left\{ {n \atop k} \right\} &= [x^n] \frac{x^k}{(1-x)(1-2x)...(1-kx)} \\
&= [x^{n-k}] \frac{1}{(1-x)(1-2x)...(1-ks)} \\
&= [x^{n-k}] \sum_{r=1}^{k} \frac{\alpha_r}{(1-rx)} \\
&= \sum_{r=1}^{k} \alpha_r [x^{n-k}] \frac{1}{(1-rx)}
\end{aligned}$$

From the first chapter, we know that $[x^n]\frac{1}{(1-rx)} = r^n$

$$\begin{aligned}
\left\{ {n \atop k} \right\} &= \sum_{r=1}^{k} \alpha_r r^{n-k} \\
&= \sum_{r=1}^{k} (-1)^{k-r} \frac{r^{n-1}}{(r-1)!(k-r)!} \\
&= \sum_{r=1}^{k} (-1)^{k-r} \frac{r^n}{r!(k-r)!}
\end{aligned}$$

And so we have a formula for the Stirling numbers of the second kind, albeit a quite messy one. We also note that

$$k! \left\{ {n \atop k} \right\} = \sum_{r=1}^{k} \binom{k}{r} (-1)^{k-r} r^n$$

which may look familiar. If we return to the opsgf $A_n(x)$ for $\left\{ {n \atop k} \right\}$ over n and recall the rule for the multiplication of two egfs: $[x^n/n!](A(x)B(x)) = \sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k}$, then we find that

$$\begin{aligned}
A_n(x) &= \sum_{k \geq 0} \left\{ {n \atop k} \right\} x^n = \sum_{k \geq 0} k! \left\{ {n \atop k} \right\} \frac{x^k}{k!} \\
&= \left(\sum_{k \geq 0} k^n \frac{x^k}{k!} \right) \times \left(\sum_{k \geq 0} (-1)^k \frac{x^k}{k!} \right) \\
&= e^{-x} \left(\sum_{k \geq 0} \frac{k^n x^k}{k!} \right)
\end{aligned}$$

Now that we have an opsgf for the Stirling numbers of the second kind over k, we can find their sum, i.e. the Bell numbers, by substituting in $x = 1$

$$\begin{aligned}
b(n) &= \sum_{k \geq 0} \left\{ {n \atop k} \right\} = A_n(1) \\
&= \frac{1}{e} \sum_{k \geq 0} \frac{k^n}{k!}
\end{aligned}$$

There is a more common way to reach the same result from the formula for the Stirling numbers of the second kind without using egf multiplication. I leave finding this approach as an exercise for the reader.

Again, e makes a guest appearance. The fact that this specific irrational number shows up so often in discrete combinatorics is surprising, and frankly, I don't truly understand why e shows up so often, other than from the fact that e^x is an essential egf, and that egf multiplication encodes the combination and relabelling combinatorial structures (in this case classes from partitions).

Our formula for the Bell numbers is itself interesting, but it involves an infinite sum, making it impossible to compute, so we will try to find another form by looking at the exponential generating function of $b(n)$.

$$B(x) = \sum_{n \geq 0} b(n) \frac{x^n}{n!}$$

Since the only way to partition $[0]$ is into the 0-partition $\{\}$, $b(0)x^0/0! = b(0) = 1$. We also note that when $n \geq 1$, $0^n = 0$, so $b(n) = \frac{1}{e}\sum_{k\geq 1}\frac{k^n}{k!}$

$$\begin{aligned} B(x) - 1 &= \frac{1}{e}\sum_{n\geq 1}\frac{x^n}{n!}\sum_{k\geq 1}\frac{k^n}{k!} \\ &= \frac{1}{e}\sum_{k\geq 1}\frac{1}{k!}\sum_{n\geq 1}\frac{(kx)^n}{n!} \\ &= \frac{1}{e}\sum_{k\geq 1}\frac{1}{k!}(e^{kx} - 1) \\ &= \frac{1}{e}\left(\sum_{k\geq 1}\frac{(e^x)^k}{k!} - \sum_{k\geq 1}\frac{1}{k!}\right) \\ &= \frac{1}{e}\left(e^{e^x} - e\right) \\ &= e^{e^x - 1} - 1 \end{aligned}$$

Therefore we have the elegant exponential generating function e^{e^x-1} for the Bell numbers. We have reduced the complexities of partitions of sets to a combination of only exponentiation and subtraction by 1, showing how the idea of a permutation can be represented algebraically similarly to how 2^n represents the idea of a power set. “This result is surely an outstanding example of the power of the generating function approach” (Wilf, 22). We aren’t quite done yet, because we want a simple method of finding $b(n)$. So far we have only used recurrence relations to find generating functions, but now we will do the reverse and use our generating function to find a recurrence relation for $b(n)$.

To find a recurrence relation, we will use a standard method that, much like other methods related to generating functions, seems unusual but is sensible once you understand why it works, even if it still feels surprising. We will take the natural logarithm, differentiate, and multiply by x for both sides of the equation of our generating function. On the RHS (right hand side, in this case e^{e^x-1}) this makes sense, since functions related to or containing e^x tend to lend themselves naturally to differentiation and natural logarithms. What this does to a power series like $B(x)$ is, as we will see, convert it into a fraction involving $B(x)$ and a related generating function, allowing us to find a recurrence between terms $b(n)$. Natural logarithms reduce complexity from multiplications by converting them to a sum, and differentiation converts the leftover logs of sums to rations. Multiplying by x restores the power of x dropped in differentiation.

$$\begin{gathered} \sum_{n\geq 0} b(n)\frac{x^n}{n!} = e^{e^x-1} \\ \ln\left(\sum_{n\geq 0} b(n)\frac{x^n}{n!}\right) = e^x - 1 \\ \frac{xB'(x)}{B(x)} = \frac{\sum_{n\geq 0}\frac{nb(n)x^n}{n!}}{\sum_{n\geq 0}\frac{b(n)x^n}{n!}} = xe^x \\ \sum_{n\geq 0} nb(n)\frac{x^n}{n!} = xe^x\sum_{n\geq 0} b(n)\frac{x^n}{n!} \end{gathered}$$

We are left with an egf being equal to the multiple of two egfs. We can use the egf multiplication rule to find that

$$
\begin{aligned}
nb(n) &= \sum_{k=0}^{n} \binom{n}{k}(n-k)b(k) \\
b(n) &= \frac{1}{n}\sum_{k=0}^{n} \frac{n!}{k!(n-k)!}(n-k)b(k) \\
b(n) &= \sum_{k=0}^{n} \binom{n-1}{k} b(k)
\end{aligned}
$$

And so we finally have a way to compute the Bell numbers. Definitely a step up from strings, the sequence of Bell numbers is one that has "been rediscovered many times" in the words of Eric Temple Bell, after whom the numbers are named. The idea of set partitions is first recorded to have popped up in medieval Japan, and Bell numbers are now used in clustering algorithms in computer science and in statistics to describe the shape of the Poisson distribution. This problem is fascinating to me for the ingenuity it uses to turn a combinatorial mess into a simple generating function, which has many real world uses. However, it is only a glimpse of the applications and creativity that generating functions approaches can have.

5. What a Wonderful Dream it had been

A final kind of generating function which we will only glance at is a Dirichlet series. The Dirichlet series generating function of $\{a_n\}_1^\infty$ is $f(s) = \sum_{n\geq 1} a_n/n^s$. The Dirichlet series generating function of $\{1\}_1^\infty$ is the Riemann zeta function $\zeta(s)$, which is the subject of one of the most important unsolved problems in maths - the Riemann zeta hypothesis. This problem is one of the Millennium Prize Problems, which means that it is one of the most influential unsolved problems towards the development of maths, and its solver will be awarded 1 million dollars. We will look at what about the function makes it so special, and what this says about generating functions as a whole.

$\zeta(s)$ is of interest for the unique multiplication rule of Dirichlet series, making it similar to the function e^x in a way. This multiplication rule is: If $f(s)$ is the Dirichlet series generating function of $\{a_n\}_1^\infty$, and $g(s)$ is the Dirichlet series generating function of $\{b_n\}_1^\infty$,

$$
\left[\frac{1}{n^s}\right] f(s)g(s) = \sum_{d|n} a_d b_{n/d}
$$

What this means is that

$$
\left[\frac{1}{n^s}\right] \zeta(s)f(s) = \sum_{d|n} a_d
$$

Where $\sum_{d|n} a_d$ is the sum of the factors of a number n. Critically, this means that the Riemann zeta function can be used to analyse prime number, factors of numbers, and number theoretic sequences. A number theoretic sequence $\{a_n\}_1^\infty$ is one whose terms are determined solely by the terms a_p, where p is a power of a prime.

What I meant to demonstrate for the final time, using Dirichlet series, is that generating functions quite literally have infinite potential, since there any type of generating function can be devised for a certain field. These generating functions, alongside other crazy abstract forms of algebra, upended many of the areas of maths which we thought we knew, such as set theory, number theory, and combinatorics. Maths is thousands of years old, originating from a need to count things, and is still constantly changing and developing at an increasing rate. Unfortunately, there is no differential equation to find this rate - you'll just have to take my word for it.

Topics like counting prime numbers continue to engage the imaginations of mathematicians, which is what fuels this development. As a very introspective subject, in maths beauty and imagination often comes first, and this approach tends to be unreasonably effective at predicting and assisting discoveries in the natural sciences. Hopefully this essay has lit your imagination in the same way as it's topic did mine.

Finally we come to the question, what did Alice see through the looking glass? I have cited how Dodgson criticised modern maths many times in Alice, but I enjoy his work nonetheless, because I interpret it as showing how despite (or because of?) much of the strangeness in maths, the subject inspires wonder, and encourages us to keep on dreaming.

Bibliography

3Blue1Brown (2022). *Olympiad level counting: How many subsets of {1,…,2000} have a sum divisible by 5? YouTube*. Available at: https://www.youtube.com/watch?v=bOXCLR3Wric [Accessed 7 Apr. 2025].

Bayley, M. (2010). Opinion | Algebra in Wonderland.
The New York Times. [online] 6 Mar. Available at:
https://www.nytimes.com/2010/03/07/opinion/07bayley.html#.

from (2012).
Finite strings from infinite alphabet. [online] Mathematics Stack Exchange. Available at: https://math.stackexchange.com/questions/107372/finite-strings-from-infinite-alphabet [Accessed 7 Apr. 2025].

numbers (2017).
Showing cardinality of all infinite sequences of natural numbers is the same as the continuum. [online] Mathematics Stack Exchange. Available at: https://math.stackexchange.com/questions/2397108/showing-cardinality-of-all-infinite-sequences-of-natural-numbers-is-the-same-as [Accessed 7 Apr. 2025].

Philippe Flajolet and Sedgewick, R. (2009).
Analytic combinatorics. Cambridge ; New York: Cambridge University Press.

Stewart, I. (1996).
From here to infinity : [a guide to today's mathematics]. Oxford: New York.

the (2015).
What is the need of exponential generating functions on combinatorial problems? [online] Mathematics Stack Exchange. Available at: https://math.stackexchange.com/questions/1133712/what-is-the-need-of-exponential-generating-functions-on-combinatorial-problems [Accessed 7 Apr. 2025].

the, U. (2022).
Understanding the functionness of a generating function. [online] Mathematics Stack Exchange. Available at: https://math.stackexchange.com/questions/4490022/understanding-the-functionness-of-a-generating-function?noredirect=1&lq=1 [Accessed 7 Apr. 2025].

Veritasium (2025).
The Man Who Almost Broke Math (And Himself…). [online] YouTube. Available at: https://www.youtube.com/watch?v=_cr46G2K5Fo [Accessed 7 Apr. 2025].

Vsauce (2015).
The Banach–Tarski Paradox. *YouTube*. Available at: https://www.youtube.com/watch?v=s86-Z-CbaHA [Accessed 7 Apr. 2025].

Wikipedia. (2021a).
Bell number. [online] Available at: https://en.wikipedia.org/wiki/Bell_number.

Wikipedia. (2021b).
Generating function. [online] Available at: https://en.wikipedia.org/wiki/Generating_function.

Wilf, H.S. (2005).
generatingfunctionology. CRC Press.