Generating Functions II - Partitions, Pentagons, and Power Series

Nic Ford

1 Introduction

This article is a followup to an earlier one on a technique in combinatorics called *generating functions*. We'll be exploring the fascinating relationship between counting *partitions* — basically the number of ways of writing a positive integer as a sum of smaller positive integers — and a sequence of integers called the *pentagonal numbers*.

The result we'll be proving, called the *Pentagonal Number Theorem*, is one of my favorite applications of generating functions. It uses generating functions in a somewhat unexpected way: we will find a generating function for the thing we're interested in counting, but rather than use it to find a formula like we did in our previous examples, we will use it to produce a different generating function and ask what *it* counts, leading to an answer that both delivers the proof of the theorem and is very pretty in its own right.

If you're familiar with generating functions, you should be fine to read this article without the first one. If you're not but you're excited to dive into the problem we'll be exploring here, you could probably get away with just reading the first half of the first article, skipping the sections on derangements and the Fibonacci numbers that appear at the end.

I'm grateful to Jake Levinson for his helpful comments on this article.

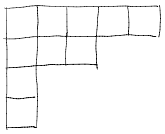
2 Setting Up the Problem

2.1 What Is a Partition?

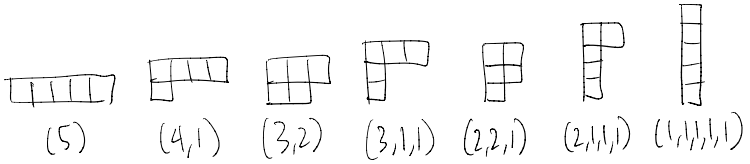
The objects we'll be counting are called *partitions*. Given a positive integer n, a **partition of** n is a way of writing n as a sum of positive integers. Order doesn't matter, and repeats are allowed: (10), $(7,3)$, $(6,3,1)$, $(4,4,2)$, and $(2,2,2,2,2)$ are some of the partitions of 10.

The numbers that show up in a partition are called its **parts**. Since the order of the parts doesn't matter, $(7,3)$ and $(3,7)$ are considered the same partition. The common convention, which we'll follow from now on, is to always write the parts in decreasing order.

You can specify a partition by listing its parts, but it'll also be useful to have a more visual representation. The **Young diagram** of a partition is a picture where each part is represented by a row of boxes. For example, the Young diagram of the partition $(5,3,1,1)$ looks like this:



There are seven partitions of 5. Here they are in both Young diagram and list-of-parts form:



If you'd like some practice, try to draw Young diagrams for all the partitions of 6. You should find that there are 11 of them.

2.2 The Partition Function

The problem we'll be investigating is how to determine, for each n, the number of partitions of n. We'll denote this number by $p(n)$, and p is called the **partition function**. The examples we just discussed tell us that $p(5) = 7$ and $p(6) = 11$. Here are the values of $p(n)$ for the first few n's:

n	0	1	2	3	4	5	6	7	8	9	10	11	12
$p(n)$	1	1	2	3	5	7	11	15	22	30	42	56	77

If you read the previous article on generating functions, you might have a guess about where we're going from here: we'll find a generation function for $p(n)$ — that is, find a nice way to write the power series $\sum_{n=0}^{\infty} p(n)x^n$ — and do some fancy-looking algebra to extract a way to write $p(n)$ in closed form.

If this was your guess, then I have some surprising news. While we will indeed be finding a generating function for $p(n)$, we won't use it to write $p(n)$ in closed form. In fact, *no one knows* how to write $p(n)$ in closed form! While a lot is known about $p(n)$, finding an explicit formula for it is a major unsolved problem in combinatorics, and one that hardly anyone in the field expects to be solved any time soon.

So our goal is going to have to be more modest. We're going to uncover a *recursive relationship* between the values of $p(n)$ for different n's. While the resulting rule doesn't give a formula, it is possible to compute $p(n)$ for n by applying the rule repeatedly, as we'll see once we have the rule in hand.

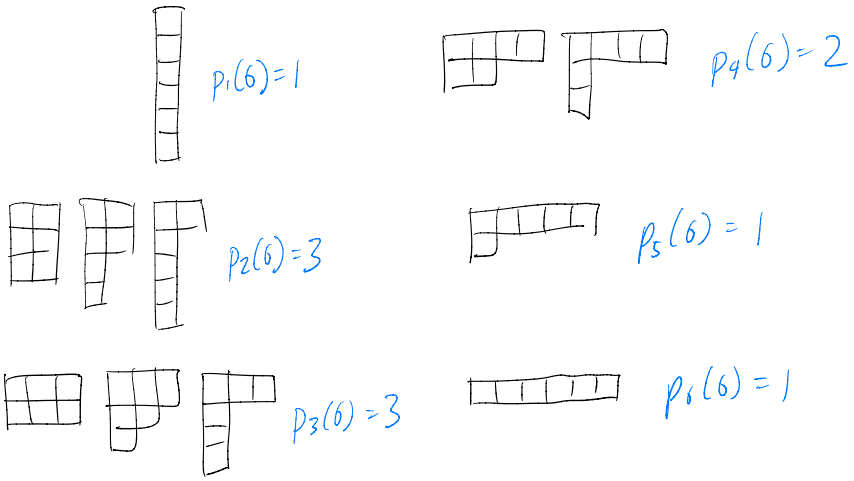
2.3 Uncovering the Pattern

We're going to start by trying to guess the pattern without worrying about the proof. This part of the story is pretty down-to-earth: it won't require generating functions at all, just playing around with pictures of partitions. As we go through this part of the story, I encourage you to pause every once in a while and see what patterns you can uncover on your own before moving on! After we've seen roughly what the pattern looks like, we'll see how to use generating functions to prove it.

We'll start by looking for some recursive patterns in the values of $p(n)$. In other words, we're looking for some way to count partitions of n using information about partitions of numbers

smaller than n. There are tons of patterns to be found here — you might find some others yourself — but we'll focus on the one that leads to the rule we'll be discussing later on.

If we start with a partition of n, one way to get a smaller partition is to delete the first part, that is, the top row of the Young diagram. If you start with a partition whose first part is k, deleting the first part gives you a partition of $n-k$. Since the answer depends on what the first part is, it will be helpful to split up the partitions of n accordingly, so let's introduce some notation for this: we'll write $p_k(n)$ for the number of partitions of n whose first part is k. Here are all the partitions of 6 split up in this way:



Because every partition of n will appear in exactly one of these groups, we can write $p(n)$ as the sum of all the $p_k(n)$'s, that is,

$$p(n) = p_1(n) + p_2(n) + \cdots p_n(n).$$

Now, there are some partitions of $n-k$ that you can't get by starting with a partition of n with first part k and deleting the first part: since the parts appear in decreasing order, the first part of the result has to be less than or equal to k. I encourage you to convince yourself that this amounts to the following rule: if $k < n$, then

$$p_k(n) = p_1(n-k) + p_2(n-k) + \cdots + p_k(n-k).$$

For example, this means that $p_2(6) = p_1(4) + p_2(4)$. Take a moment to check this!

Let's write this last equation in a slightly different way. We said that the result of our deletion has first part less than or equal to k, but this is the same as saying that the first part can't be greater than k. That is, we can get any partition of $n-k$ *except* the ones whose first part is $k+1$ or bigger. In other words,

$$p_k(n) = p(n-k) - (p_{k+1}(n-k) + \cdots + p_{n-k}(n-k)).$$

The recursive relationship between the $p(n)$'s that we're looking for will come from applying this rule over and over again. Let's see how it works for $p(6)$ and it should be reasonably clear that the method will work in general.

We'll start by writing

$$p(6) = p_1(6) + p_2(6) + p_3(6) + p_4(6) + p_5(6) + p_6(6).$$

If we rewrite each term using our rule for $p_k(n)$, we get

$$\begin{aligned} p(6) = {} & p(5) + p(4) + p(3) + p(2) + p(1) + p(0) \\ & - (p_2(5) + p_3(5) + p_4(5) + p_5(5)) \\ & - (p_3(4) + p_4(4)). \end{aligned}$$

We get to stop with the subtractions here, because $p_3(6) = p_1(3)+p_2(3)+p_3(3)$, which is actually *all* the partitions counted by $p(3)$, so there is nothing to subtract, and the same is true for all the subsequent terms.

From here, we can apply the rule again to those six terms at the end and get:

$$\begin{aligned} p(6) = {} & p(5) + p(4) + p(3) + p(2) + p(1) + p(0) \\ & - (p(3) - p_3(3)) - p(2) - p(1) - p(0) \\ & - p(1) - p(0). \end{aligned}$$

The only term left that isn't a $p(n)$ is the $p_3(3)$, which we can replace with $p(0)$. If we make that substitution and then cancel all the terms that cancel, we are left with

$$p(6) = p(5) + p(4) - p(1).$$

You can check this directly using our table from earlier: this equation becomes $11 = 7 + 5 - 1$.

If we ran this procedure for any $p(n)$, it would always terminate: every time we apply our rule for $p_k(n)$ we decrease the number inside the parentheses, so we eventually hit the end.

It's worth reflecting on how simple our final expression is. Most of the terms cancelled, and the ones that didn't all ended up with a coefficient of 1 or -1. If you tried this for larger n's, you would find that this remains true. For example, after a lot more steps, we'd get that

$$p(24) = p(23) + p(22) - p(19) - p(17) + p(12) + p(9) - p(2).$$

It's far from clear why this should happen! Why do almost all the terms cancel, and what's special about the ones that don't? These are exactly the questions that will be answered by looking at the generating function for $p(n)$. (It's also possible to answer these questions by analyzing our procedure directly, but, at least in my opinion, the resulting argument isn't nearly as nice. If you're interested in a version of the story that *does* proceed that way, using a slightly different way of splitting up the partitions, there is a nice explanation in this MathPages article.)

3 Generating Functions

3.1 The Generating Function for the Partition Function

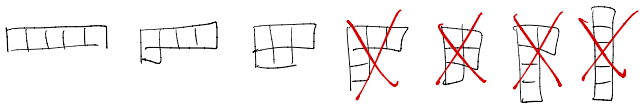
We'll now turn to writing down a generating function for $p(n)$. We'll approach this problem in a couple of stages, both because this should make the final answer easier to understand and because one of the intermediate steps will be useful for a different reason later on.

Our first example in the previous article was the generating function for $\binom{n}{k}$, which was

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

One way to see this is to imagine multiplying n copies of $(1+x)$ and counting how many times we'll get x^k. Each term of the product comes from picking either the 1 or the x from each of the $(1+x)$'s, and you'll get x^k if you picked k x's and $n-k$ 1's. Therefore, the number of times we get x^k is equal to the number of ways of selecting, from among the n factors of $(1+x)$, k of them to give you an x. And this is exactly what $\binom{n}{k}$ counts.

We can use a similar argument to produce a generating function for $p(n)$. As a first step, let's consider a related object called the **distinct partition function**, often written $q(n)$. This is the number of partitions of n with no repeated parts. For example, of the seven partitions of 5, all but three have repeated parts, so $q(5) = 3$:



The generating function for $q(n)$ is then

$$\sum_{n=0}^{\infty} q(n)x^n = (1+x)(1+x^2)(1+x^3)\cdots.$$

We can see this by asking, like we did for the $\binom{n}{k}$'s, how you might get an x^n to pop out when you expand this product. You get a term in the product by selecting either the 1 or the x from the first factor, either the 1 or the x^2 from the second factor, and so on; you'll get x^n exactly when the powers of x that you picked add up to n. In other words, each x^n comes from picking a set of distinct positive integers that add up to n. This is exactly what $q(n)$ counts!

(You might be a bit queasy about the fact that we've written our generating function as an infinite product. Is this well-defined? It may help to notice that, while we are multiplying infinitely many factors together, the contribution to any particular coefficient can only come from finitely many of them. If, for example, you wanted to know the coefficient of x^{10}, then when you expand out the product you can't be selecting the x^{11} from $(1+x^{11})$, since this would already push the power of x past 10, and the same is true for all the higher powers of x.)

Since each factor we're multiplying looks like $(1+x^k)$, when we're expanding out the product, we can either take the x^k or the 1, which corresponds to either including k in our partition or not. If we instead want a generating function for $p(n)$, we want to find a way to allow k to be included more than once. We can accomplish this by replacing $(1+x^k)$ with $(1+x^k+x^{2k}+x^{3k}+\cdots)$: this way, when it comes time to select a term from this factor, we can pick x^{mk}, which corresponds to including k in our partition m times.

As we saw a couple of times the previous article, this is a geometric series:

$$1+x^k+x^{2k}+\cdots = \frac{1}{1-x^k}.$$

If we make this replacement in our generating function for $q(n)$ — that is, replace each $(1+x^k)$ with $1/(1-x^k)$ — we get the following generating function for $p(n)$:

$$\sum_{n=0}^{\infty} p(n)x^n = \frac{1}{(1-x)(1-x^2)(1-x^3)\cdots}.$$

3.2 A Recursive Relationship

Our formula for $p(n)$'s generating function implies that

$$\left(\sum_{n=0}^{\infty} p(n)x^n\right)(1-x)(1-x^2)(1-x^3)\cdots = 1.$$

If we write

$$(1-x)(1-x^2)(1-x^3)\cdots = \sum_{n=0}^{\infty} c(n)x^n,$$

then using this notation, we have that $(\sum_{n=0}^{\infty} p(n)x^n)(\sum_{n=0}^{\infty} c(n)x^n) = 1$. I encourage you to check that, if you expand this out and compare the coefficient of x^n on each side, we get that

$$\sum_{k=0}^{n} c(k)p(n-k) = \begin{cases} 0 & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

For any particular n, we can see what this formula looks like by just expanding the product $(1-x)(1-x^2)(1-x^3)\cdots$ by hand and stopping once we get past x^n. For $n = 6$, you'd get

$$(1-x)(1-x^2)(1-x^3)\cdots = 1 - x - x^2 + x^5 + \cdots,$$

which means $c(0) = 1$, $c(1) = c(2) = -1$, $c(3) = c(4) = 0$, $c(5) = 1$, and $c(6) = 0$. Plugging this into the formula above gives

$$p(6) - p(5) - p(4) + p(1) = 0.$$

This is exactly the same thing we found in the last section! This is a pretty good sign that we're on the right track. If we find a way to explicitly compute the $c(k)$'s, we'll be done.

4 What Does This Count?

4.1 Evens Minus Odds

The first thing to notice is that the generating function for $c(n)$ is very similar to the generating function for $q(n)$ — that is, partitions with no repeated parts — that we considered earlier:

$$\sum_{n=0}^{\infty} q(n)x^n = (1+x)(1+x^2)(1+x^3)\cdots$$
$$\sum_{n=0}^{\infty} c(n)x^n = (1-x)(1-x^2)(1-x^3)\cdots.$$

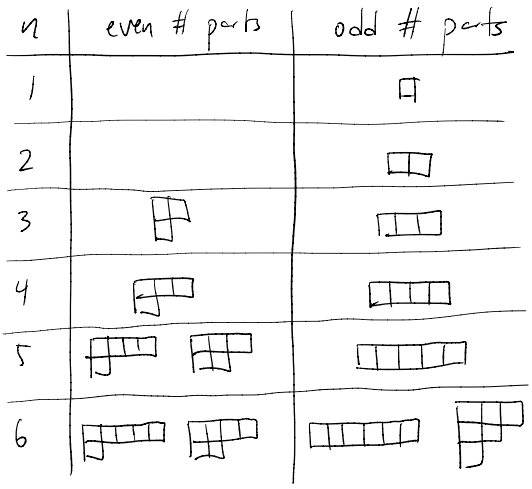
The only difference is the minus sign inside each of the factors being multiplied.

How do these minus signs change what we're counting? If we again imagine expanding the product and counting how many times we get x^n, we see that we still get one such term for each partition of n with no repeated parts, because it's still the case that we're either picking the 1 or the x^k from each $(1-x^k)$, and we still get x^n exactly when the powers of x add up to n. But

now, every time we pick x^k — that is, every time we stick a row of k onto our partition — we also multiply the term by -1.

So our term will be x^n if the partition has an even number of rows and $-x^n$ if it has an odd number of rows. For example, there are three ways to get x^5: it can come from $(-x^5)$, $(-x^4)(-x)$, or $(-x^3)(-x^2)$. The first of these ends up with a minus sign and the others each end up with a plus sign, so the final coefficient on x^5 is 1, as we saw near the end of the last section. Putting this all together, you can compute $c(n)$ by finding, among all partitions of n with no repeated parts, the *difference* between the number of partitions with an even number or parts and the number with an odd number of parts.

We can make this a bit more concrete by looking at a couple examples. Here are all the partitions of 1 through 6 with no repeated parts split up based on whether they have an even or odd number of parts:



From the diagram, we see once again that when $n = 5$, the left column has one more partition than the right, giving another way to see that $c(5) = 1$. On the other hand, when $n = 6$, we get the same number of partitions on each side, so $c(6) = 0$.

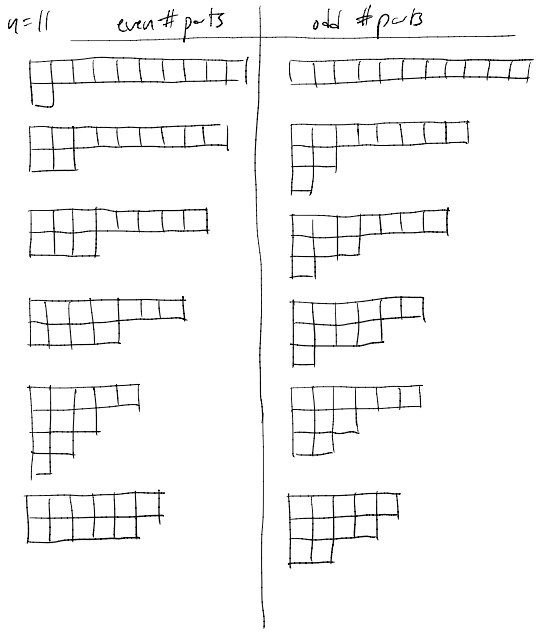
4.2 Pairing Off Partitions

This gives us a way to determine the values of $c(n)$, and therefore the coefficients in our relationship between the $p(n)$'s. It's a nice pattern, but if we left things here it wouldn't really be a satisfying answer to our original question. For one thing, we'd be "answering" our question about how to count partitions by just counting some different partitions. But more substantively, there was something quite striking about the original recursive relationship that would go totally unexplained: why did most of the coefficients end up being 0? And were the few that weren't always 1 or -1?

We can get to the bottom of this mystery by spending a bit more time analyzing what $c(n)$ is counting. This last step will finally deliver on the promise from the introduction about pentagonal numbers, and in my opinion it's the most beautiful part of the whole story.

Our examples from earlier suggest that most of the $c(n)$'s are going to end up being zero. If some $c(n) = 0$, our earlier description implies that, among the partitions of n with no repeated parts, there are the *same* number with an odd number of parts as with an even number of parts.

One way to show that two sets have the same number of elements is to find some way of pairing off elements of the first set with elements of the second, so let's start by looking at an example n and see if we can manage to do this for the partitions we're counting. Here are all the partitions of 11 with no repeated parts, split up based on whether they have an even or odd number of parts:



We can maybe see the beginning of a way to pair these off with each other: we can either break off a box from the end of the first row and stick it on the bottom, or, if the bottom row has just one box, we can do the reverse. This operation will always either add or remove one row, so it will take a partition with an odd number of parts to one with an even number and vice versa.

The only pair where this doesn't quite work is the bottom one in the diagram. In that one, the second row has just one fewer box than the first, so if we took a box off the top row we would introduce a repeated part. But we can fix this with a slight modification to the rule. We can peel off the entire last diagonal and stick it on the bottom, and that does the job:

The complete rule for pairing off our partitions is as follows:

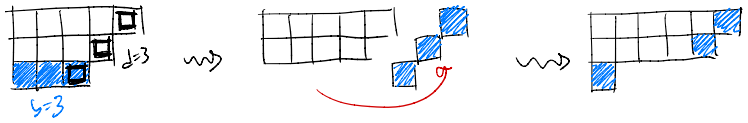
- Let b be the length of the bottom row, and let d be the length of the last diagonal (that is, the longest diagonal of squares you can find by starting at the top right and going down and to the left at a 45 degree angle).
- If $b > d$, remove all the boxes in the diagonal and place them on the bottom row.
- If $b \leq d$, instead remove all the boxes from the bottom row and place them along the diagonal.

If you'd like, try using this rule to pair off the partitions of 13 with each other. (We'll see in a minute that 12 is special!)

4.3 Pentagonal Numbers

Now, we are trying to show that *most* of the $c(n)$'s are zero, not all of them. That means that our pairing-off procedure must fail for some n's.

There are two possible ways for this to happen, and they both involve the cases where the diagonal we're counting reaches all the way to the bottom row. The first possibility is that $b = d$, and the partition has d rows. In this case, the bottom row and the diagonal share a box, so when we remove the bottom row as our rule tells us to, we've also removed the end of the diagonal:

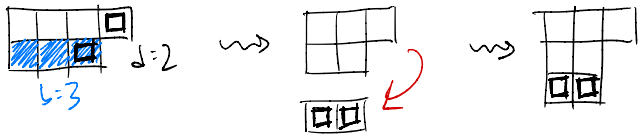


This causes a couple of problems. First, the output partition has the same number of parts as the input, whereas we were trying to switch from an odd to an even number of parts or vice versa. Second, this process isn't reversible: I encourage you to check that if you tried applying our rule to the partition at the end in this example, you wouldn't get the partition at the beginning.

For any given n, there can be at most one partition that looks like this: the bottom row has to be d, the next row up has to be $d+1$, and so on up to $d+(d-1)$, since there are d rows in total. If there *is* a partition of n like this, we can add up these numbers to learn what n has to be: we get

$$n = d + (d+1) + (d+2) + \cdots + (d+(d-1)) = d^2 + \frac{1}{2}d(d-1) = \frac{d(3d-1)}{2}.$$

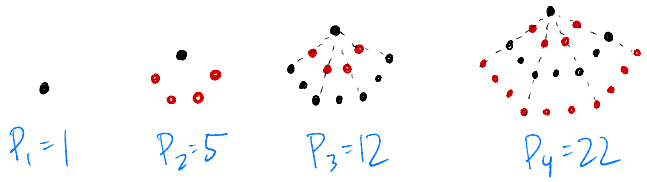
There's one other way our procedure could fail. If $b = d+1$ and the partition has d rows, our rule tells us to remove the diagonal and stick it on the bottom. But when we remove the diagonal we're also removing a box from the bottom row, so when we try to add it to the bottom we end up with a repeated part:



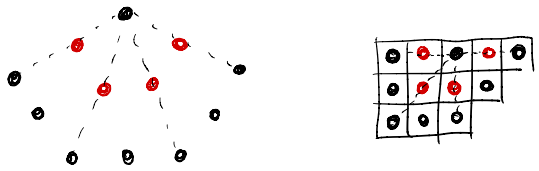
For the same reasons as last time, any given n can have at most one partition like this, and you can once again add up the rows to see which n's this can happen for. I encourage you to check that you get $n = \frac{d(3d-1)}{2} + d$.

I also encourage you to check that these are the *only* two cases where our rule can fail: for any partition that isn't one of these two types, we succeed in pairing it off with exactly one partner.

The number $\frac{d(3d-1)}{2}$ has a name: it's called the ***d*'th pentagonal number**, and we'll write it P_d. The name comes from a way of arranging P_d dots into a pentagon, kind of similar to the square or triangular numbers you might be more familiar with:



You can actually see the same pentagonal shape in disguise in the first type of partitions that cause our rule to break:



The second type just look like these with an extra column on the left.

So what does this tell us about the $c(n)$'s and our recursive relationship between the $p(n)$'s? If n is not of the form P_d or $P_d + d$, then our rule perfectly pairs of our partitions, so $c(n) = 0$. If n *is* equal to P_d or $P_d + d$, then there will be exactly one partition that doesn't get a partner. The number d is also the number of parts in the partition, so that's what determines whether the extra partition has an even or odd number of parts, and therefore whether $c(n)$ ends up as 1 or -1. Summing up, we have that, if $n > 0$,

$$c(n) = \begin{cases} 1 & \text{if } n = P_d \text{ or } P_d + d, \text{ and } d \text{ is even} \\ -1 & \text{if } n = P_d \text{ or } P_d + d, \text{ and } d \text{ is odd} \\ 0 & \text{otherwise.} \end{cases}$$

(The way we've set things up, we need to treat $n = 0$ as a special case and also declare that $c(0) = 1$. Do you see why $n = 0$ doesn't exactly fit into this template?)

The $c(n)$'s appeared in our recursive formula for the $p(n)$'s via the fact that, if $n > 0$, then

$$\sum_{k=0}^{n} c(k)p(n-k) = 0.$$

With our new characterization of the $c(n)$'s, we can pull the $k = 0$ term to the other side of the equation and write

$$\begin{aligned} p(n) &= p(n - P_1) + p(n - (P_1 + 1)) - p(n - P_2) - p(n - (P_2 + 2)) + \cdots \\ &= p(n-1) + p(n-2) - p(n-5) - p(n-7) + \cdots, \end{aligned}$$

where we stop adding terms once the pentagonal numbers get bigger than n. I encourage you to compare this to the cases we worked out for $n = 6$ and $n = 24$ back at the beginning and see that they match!

This is the **Pentagonal Number Theorem**, and it's one of my favorite applications of generating functions. The generating function gets used in such a counterintuitive way: rather than leading directly to a formula for $p(n)$, the generating function instead led us to a *different* counting problem, for $c(n)$, which turned out to be easier to solve, and solving it is what gave us our recursive formula. There is of course a huge amount more to be said about partitions — they're some of the most well-studied objects in combinatorics — but I hope you've enjoyed this little taste, and I'm looking forward to exploring more applications of generating functions as this series continues.