Adventures with Pascal's triangle and Binary Numbers

Daniel Mathews

November 10, 2004

Some of us have probably seen the following array of numbers before:

$$\begin{array}{c} 1 \\ 1\ 1 \\ 1\ 2\ 1 \\ 1\ 3\ 3\ 1 \\ 1\ 4\ 6\ 4\ 1 \\ 1\ 5\ 10\ 10\ 5\ 1 \\ \vdots \end{array}$$

Where, as one can see, each entry is obtained by adding the two entries directly above it – Pascal's triangle as it's known.

All sorts of fun stuff occurs when one plays around with Pascal's triangle. For instance, let's look at the parity of the numbers in this triangle (ie, whether they're even or odd) – as one does, of course! Let's count how many odd and even numbers there are in each row.

	Number of Odd Entries	Number of Even Entries
1	1	0
1 1	2	0
1 2 1	2	1
1 3 3 1	4	0
1 4 6 4 1	2	3
1 5 10 10 5 1	4	2
1 6 15 20 15 6 1	4	3
⋮	⋮	⋮

I'm particularly interested in the number of odd entries in each row, because they turn out to be quite odd. (Sorry about that one, I just couldn't help it). We obtain the following sequence of numbers:

$$1, 2, 2, 4, 2, 4, 4, 8, 2, 4, 4, 8, 4, 8, 8, 16, 2, 4, 4, \ldots$$

If you look at that one for a while, it turns out to be quite interesting. But we'll delay that for a bit. Let's turn our mind to something equally obscure...

remember binary numbers from school? Yes that's right, those numbers which only used the digits 0 and 1 (not 2 through 9), and where the place value of each digit was not 1, 10, 100, 1000 and so on, but instead 1, 2, 4, 8, etc.

Let's write out the first few numbers in binary, starting from zero, just to get the hang of it:

$$0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, \ldots$$

Hmmmm, well writing out binary numbers is one way to spend a rainy day. But now let's consider the number of 1's in the binary representation of each number.

Number in decimal	0	1	2	3	4	5	6	7	8	9	10
Number in binary	0	1	10	11	100	101	110	111	1000	1001	1010
Number of 1's	0	1	1	2	1	2	2	3	1	2	2

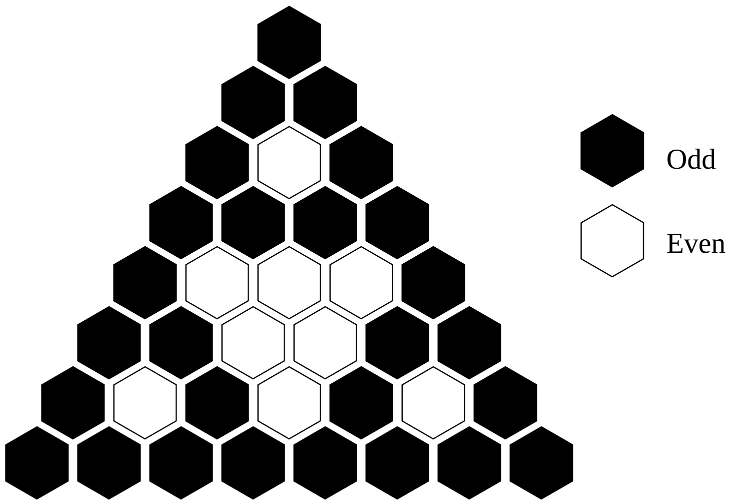
Well what the heck, after all that experience I bet we're feeling a bet wild so let's do something crazy – let's take all these numbers we just got (that is, the number of 1's in each number's binary representation), and let's take 2 to the power of it. We get:

$$1, 2, 2, 4, 2, 4, 4, 8, 2, 4, 4, 8, 4, 8, 8, 16, 2, 4, 4, \ldots$$

which looks entirely familiar... yes, it looks like these two sequences, the number of odd entries in each row of Pascal's triangles, and the number of 1's in the binary representation of each number, are in cahoots! Yes they're identical! This leads us to the remarkable equation, generalising what we just saw:

$$\{\text{\#Odd entries in n'th row of Pascal's triangle}\} = 2^{\{\text{\#1's in binary representation of n}\}}$$

This is indeed an obscure connection. But we can prove it. Well, I'm not going to prove it rigorously but I'll give you the idea. Firstly, look at how the odds and evens appear in Pascal's triangle:



This looks quite remarkable and, for those who know something about fractals, it has exactly the same pattern as the Sierpinski gasket. The rows with all odd numbers are rows number $0, 1, 3, 7, 15, \ldots$. These look like all the numbers which are one less than a power of 2, and in fact this is true – all rows numbered $2^n - 1$ for some n have all entries odd. You see, when we have an all-odd row, the next row is all-even, so the odds wipe themselves out completely (since each entry in the next row is the sum of the two entries above it – odd + odd = even). But not quite completely – the two extremities of the next row are odd, as each of them only has one entry above it. So now we have two odd entries in the next row (row number 2^n), at the very extremities.

But what happens now? The whole process starts again, TWICE! That is, the two odd entries at either side start entirely new versions of the original triangle, in exactly the same pattern! And they don't intersect for a while because they are so far apart – they can only spread down like the original triangle. In fact, they are just so far apart (you can check it if you like) so that, at the next power-of-two-minus-one, $2^{n+1} - 1$, they are just about to intersect and the row is all odd again! So the odds wipe themselves out again and the doubling process starts all over again, so on and so forth.

What has this got to do with binary numbers? Well the power of 2 gives us a clue. We can actually write an equation out of my last two paragraphs of rambling, believe it or not. First, let the number of odd entries in the n'th row be $f(n)$. We already know because of our wipe-out theory that at every power of 2 there are only 2 odd entries, so $f(2^k) = 2$ for all possible k. Because of the way that the triangle replicates itself twice, if we go, say x steps beyond the "doubling point" 2^n, then the point we get to is a point x steps into the original triangle, copied twice. So the number of odd entries in row number $2^n + x$ is twice the number of odd entries in row number x. That is,

$$f(2^n + x) = 2f(x).$$

Excellent! But we have to remember that this doesn't work for all x, because if we go too far past 2^n the triangle will have wiped itself out again. We can check that it only works for x between 0 and $2^n - 1$. Phew!

Now to bring the binary numbers into it. When a number is written in binary form, we are basically writing it as a sum of powers of 2. For example,

$$
\begin{aligned}
10_2 &= 2^1 \\
1101_2 &= 2^3 + 2^2 + 2^0
\end{aligned}
$$

So, if we think about it, we can use the f formula on these powers of 2 because, for instance,

$$
\begin{aligned}
f(2^3 + 2^2 + 2^0) &= f(2^3 + \text{other stuff}) \\
&= 2f(\text{other stuff})
\end{aligned}
$$

(You can check for yourself that the other stuff is always within the right limits). We can now use this idea to figure out f of some numbers.

$$
\begin{aligned}
f(10_2) &= f(2^1) = 2 \text{ (as f of a power of 2 gives 2)} \\
f(1101_2) &= f(2^3 + 2^2 + 2^0) \\
&= f(2^3 + (2^2 + 2^0)) \\
&= 2f(2^2 + 2^0) \\
&= 4f(2^0) \\
&= 8
\end{aligned}
$$

And so, we can see, for every power of 2 in the sum, we have a factor of 2! But the number of powers of 2 we write out is just the number of digits in the binary representation. So we have a connection! To see this more clearly, take any old binary number with, say, m 1's in it.

$$
\begin{aligned}
\overbrace{1101011101\cdots101}^{m\text{ 1's}} &\longrightarrow \overbrace{2^{\text{stuff}} + 2^{\text{other stuff}} + \cdots + 2^{\text{more stuff}}}^{m\text{ terms}} \\
&\longrightarrow f(2^{\text{stuff}} + 2^{\text{other stuff}} + \cdots + 2^{\text{more stuff}}) \\
&\longrightarrow 2f(2^{\text{other stuff}} + \cdots + 2^{\text{more stuff}}) \\
&\longrightarrow \text{ (m times)} \\
&\longrightarrow 2^m
\end{aligned}
$$

And so we have it, O intrepid mathematical adventurers! This proves the result.