LESSON 8 : SEQUENCES

1

A *sequence* is a function that takes non-negative integers as inputs. Many sequences count the number of ways of doing things.

A *hyperbinary* number is like a binary number except that we are allowed to use the digit 2 in the expression. In normal binary numbers, we can take only none or 1 of each power of 2 to build up our number in position notation. So $6 = 4+2$ and $6_{10} = 110_2$. In normal binary, there is always exactly one way to write a number. For most purposes, this is ideal. But mathematicians like to play with possibilities.

If we extend our options to taking either 0, 1, or 2 of each power of 2, then we end up with more than one way to write many familiar numbers. Note that some numbers can still only be written in one way, even in hyperbinary.

Note that $6 = 4 + 2 = 4 + 1 + 1$, and in both cases we use no more than 2 of any power of 2. So we can write 6 at least in 2 ways using hyperbinary. In symbols, we have $6_{10} = 110_2 = 102_2$. Are there any other ways to write 6 in hyperbinary ?

We define the sequence $\beta(n)$ to be the number of ways of writing n in hyperbinary. We can pronounce this "beta of n" or "beta n".

It turns out that $\beta(2n+1) = \beta(n)$. Also $\beta(2n+2) = \beta(n) + \beta(n+1)$. Finally we have the fact that $\beta(0) = 1$, since there is only one way to write 0, even in hyperbinary.

This makes our sequence a *recursive* sequence. We can compute later values using only earlier values. But we can also compute β the hard way, by counting ways of writing n in hyperbinary.

Sequences can be written in an abbreviated way, as simply a list of numbers. This list is usually infinite, but some sequences are finite. Our sequence is infinite.

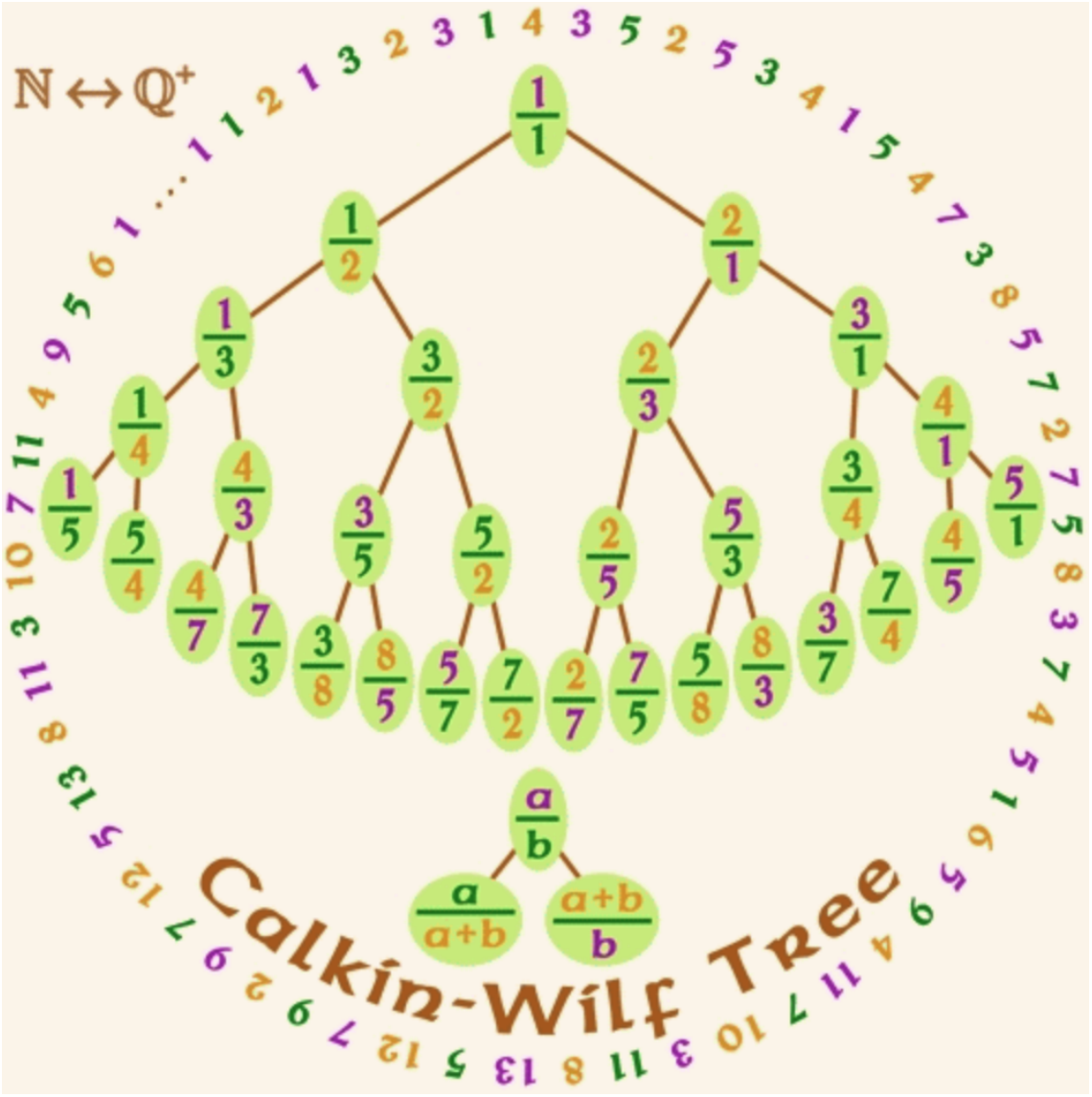
Here's the list: $1, 1, 2, 1, 3, 2, 3, 1, 4, 3, 5, 2, 5, 3, 4, 1, \ldots$.

This means $\beta(0) = 1$, $\beta(1) = 1$, $\beta(2) = 2$, $\beta(3) = 1$, $\beta(4) = 3$, and so on.

After this lesson and some practice, you should be able to calculate $\beta(n)$ for various values of n. The quickest way to do this is to use the recursion. But first we'll focus on the hyperbinary numbers, as a way to review positional notation. We will compute all the way to $\beta(10)$.

2

Now we'll look at the Calkin-Wilf tree. We start at the top with $\frac{1}{1}$. Then, from each "parent" node, we create two "children". If the parent is $\frac{a}{b}$, then the left child is $\frac{a}{a+b}$ and the right child is $\frac{a+b}{b}$.



This tree is a way to *enumerate* or "print out" each of the positive rational numbers. We just traverse the tree, one row at a time, from left to right. As we traverse the tree in this way, we see that the denominator becomes the numerator.

So we get $\frac{f(0)}{f(1)}, \frac{f(1)}{f(2)}, \frac{f(2)}{f(1)}, \frac{f(3)}{f(2)}, \ldots$

The tree implicitly defines this strange sequence $f(n)$.

It also defines a sequence of fractions, which we have mentioned but not named. Let $\phi(n)$ be the fraction in the n-th position. So $\phi(n) = \frac{f(n)}{f(n+1)}$.

Then $\phi(n)$ is a sequence, and it is a function that "maps" or "transforms" the non-negative integers to the positive rational numbers. This means that the positive rational numbers have the same "power" of infinity as the non-negative integers.

We can easily print out *all* rational numbers this way: $0, \phi(0), -\phi(0), \phi(1), -\phi(1), \phi(2), -\phi(2), ...$

Now to join these two ideas together. The sequence $f(n)$ is exactly the sequence $\beta(n)$. The sequence that counts the ways we can write a number in hyperbinary strangely also gives us the numerators and denominators of this tree.

Now the picture might make more sense. The numbers wrapping around the tree are the β sequence.

So we can read the positive rationals, all in reduced terms, and without repetition, right off the tree that we create. *Or* we can use the β sequence to print them out. We will also use f for this same sequence, since they are the same. The names of mathematical objects are basically inessential. They are like pronouns, dependent on their context.

To sum up, we have $\phi(n) = \frac{f(n)}{f(n+1)} = \frac{\beta(n)}{\beta(n+1)}$, where $n = 0, 1, 2, 3, 4, ...$

The recursion can be written

$f(0) = 1$

$f(2n+1) = f(n)$

$f(2n+2) = f(n) + f(n+1).$

Then the n-th positive rational number, in this particular but ideal enumeration, is $\phi(n)$. Note that the zero-th rational number is $\frac{1}{1} = 1$.

Here again, in a little more detail, is the way the sequence starts:

$f = 1, 1, 2, 1, 3, 2, 3, 1, 4, 3, 5, 2, 5, 3, 4, 1, 5, 4, 7, 3, 8, 5, 7, 2, 7, 5, 8, 3, 7, 4, 5, 1, 6, 5, 9, 4, 11, 7, 10, ...$

It goes on forever, because we can keep counting how many ways we can write n in hyberbinary. This is something that a computer program will do for us, but the program will take longer and longer as n gets bigger and bigger. Does infinity "really" exist ? In-finite. The negation of having a definite ending. It exists in our seeing that there's no obvious place to stop the game. We just run out of time, memory, etc.

We'll build the tree on the board, compute the number of hyperbinary representations the hard way, and also practice using the recursion.