Rethinking the Calkin-Wilf Tree Horizontally

Jiachen Xu, Max Yu

August 2017

Abstract

The Calkin-Wilf tree is a binary tree containing all rational numbers, first proposed in 2000 by N. Calkin and H.S. Wilf. Though any term in the tree can be solved, all existing algorithms in the past 17 years used to solve the tree are recursive. While the basic method according to the definition is top-down recursive, M. Newman in 2003 proved that the recursive function $q_{i+1} = \frac{1}{2\lfloor q_i \rfloor - q_i + 1}$ also generates terms on the tree. We however, in hope of finding a non-recursive algorithm, approached the Calkin-Wilf tree with a brand-new method, considering the tree starting on the first term of the nth row. Through this horizontal approach, we found two new algorithms, one better than the classical method under specific circumstances, while the other is almost non-recursive, dependent only on a recursive A_n sequence and solving a Linear Diophantine equation. Once the closed-form expression for the A_n sequence is found, the algorithm will be completely non-recursive.

Contents

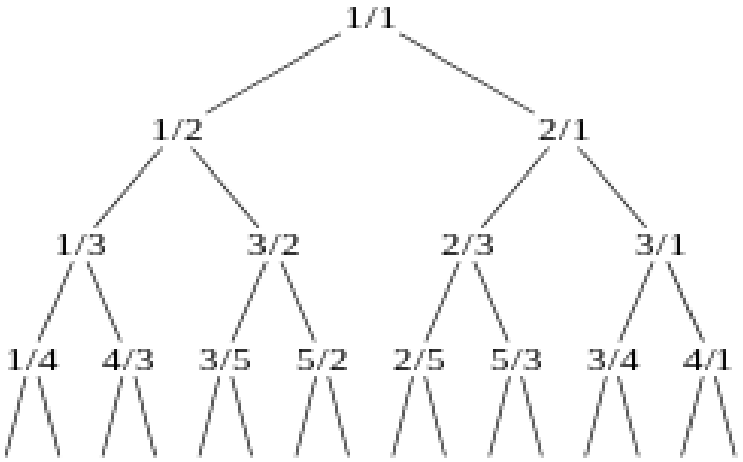
1 Introduction

The Calkin-Wilf Tree is a binary tree with vertices corresponding 1-to-1 to the positive rational numbers, first proposed by Calkin and Wilf [1]. The Calkin-Wilf Tree has a recursive formula proved by Newman, a very counter-intuitive function $q_{i+1} = \frac{1}{2\lfloor q_i \rfloor - q_i + 1}$ [3]. The classic algorithm generates terms starting from the 1st term $q_1 = \frac{1}{1}$, going from top to bottom instead. We explored the Calkin-Wilf Tree through a different approach, starting from the first term on every row, going from left to right without Newman's recursive function. We found two new algorithms through this approach.

1.1 Definitions

Definition 1.1. Root: the tree is rooted at the number $\frac{1}{1}$. In other words the first term is $\frac{1}{1}$.

Definition 1.2. Children: any node on the tree $\frac{a}{b}$ has two children. The left child $\frac{a}{a+b}$ and the right child $\frac{a+b}{b}$. *LC* is short for left child and *RC* is short for right child.



Definition 1.3. Numeration: The tree is labeled row by row from left to right. The root,$\frac{1}{1}$, is defined as the first term. For example, the second number(counted from left to right) of the second row is the $3rd$ term. Similarly, on the third row we have $4th$ to $7th$ term, and the rest follows.

Definition 1.4. $T(x)$: We use function $T(x)$ to represent the xth term of the tree.

2 Basic Properties

Here are some well known and already proven basic properties of the Calkin-Wilf tree we find useful. They all have existing proofs and are relatively trivial to prove with mathematical induction.

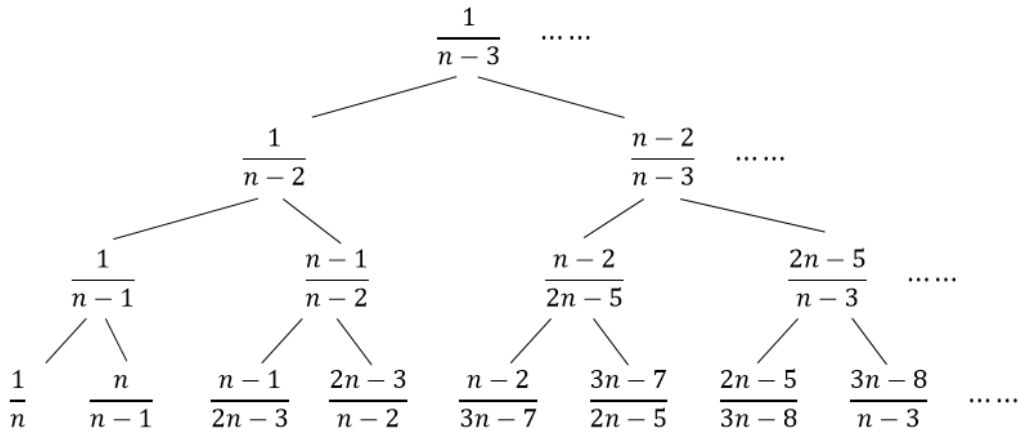
1. There are exactly 2^{n-1} fractions on the nth row.
2. (Position Lemma): $\forall x \in \mathbb{N}$, $\exists! i, j \in \mathbb{Z}_{\geq 0}$, where $x = 2^i + j$, $x < 2^{i+1}$, and $j < 2^n$. $n = i + 1$ would be the row number and $j + 1$ the position of the $P(x)$ from left to right.
3. For any node $T(x)$, its LC will be $T(2x)$ and its RC $T(2x + 1)$
4. For every node $T(x)$ on the tree, its LC: $T(2x) < 1$ and its RC: $T(2x + 1) > 1$.
5. 1 to 1: Every positive rational number only appears once in the tree. In other words, $\nexists a, b \in \mathbb{N}$, $a \neq b$, such that $T(a) = T(b)$
6. Pseudo-Symmetry: The reciprocal of any fraction (except the root) of the tree will be on the same row and are symmetrical about the middle axis in position. In other words, $\forall T(2^{n-1} + m - 1) = \frac{p}{q}$, $T(2^{n-1} + 2^{n-1} + 1 - m - 1) = T(2^{n-1} + (2^{n-1} - m)) = T(2^n - m) = \frac{q}{p}$
7. The fractions generated by the tree are all in lowest terms.
8. The first node on every row, $T(2^{n-1}) = \frac{1}{n}$

3 Rethinking the Calkin-Wilf tree

3.1 Motivations

Inspired by Candace Chiang's idea in PROMYS 2015 of starting the tree at the $\frac{1}{n}$,[2](slightly mentioned in their unpublished paper), we started our following explorations.

We start plotting the tree starting on the first term of the nth row, with the fraction $\frac{1}{n}$, according to lemma 8. For other terms, each time we move up along the tree one term, such as $\frac{1}{n-2}$ to $\frac{1}{n-3}$, and then moving down, giving us 4 new terms.



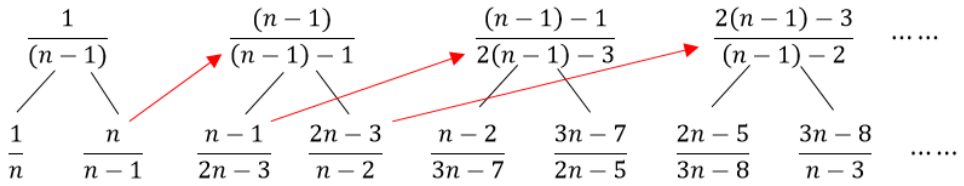
Since all the fractions are in the form of $\frac{An-B}{Cn-D}$, we extract the coefficients on the nth row as sequences of m.

Definition 3.1. $A_n(m)$, $B_n(m)$, $C_n(m)$, $D_n(m)$: Writing the node $T(2^{n-1}+m-1)$ in the form $\frac{An-B}{Cn-D}$, $A_n(m)$, $B_n(m)$, $C_n(m)$, and $D_n(m)$ are the sequences of coefficients for the mth node on the nth row.

Definition 3.2. $t_n(m)$: We define $t_n(m)$ to be the mth node starting from the left on the nth row.

3.2 Pseudo Self-Similar Nodes

We discovered that the previous method of generating $\frac{An-B}{Cn-D}$ terms was rather tedious. The tree could be extended vertically by using only 2 rows rather than the $log_2(m)$.



There is a pseudo self-similar property of the tree when taking the horizontal approach.

Lemma 3.0.1. (Pseudo Self-Similarity):

By substituting the ns of $\frac{A_n n-B_n}{C_n n-D_n}$ with $(n-1)$, we get the equation

$$\frac{A_n(m)\times(n-1)-B_n(m)}{C_n(m)\times(n-1)-D_n(m)}=\frac{A_{n-1}(m)\times n-B_{n-1}(m)}{C_{n-1}(m)\times n-D_{n-1}(m)}$$

$$\Rightarrow \begin{cases} A_{n-1}(m) = A_n(m), \\ C_{n-1}(m) = C_n(m) \\ B_{n-1}(m) = A_n(m) + B_n(m), \\ D_{n-1}(m) = C_n(m) + D_n(m) \end{cases}$$

Lemma 3.0.2. (LC and RCs:)

And the nth row is made out of LCs and RCs of the $(n-1)$th row.

For LCs:

$$\begin{cases} A_n(2m-1) = A_{n-1}(m), \\ C_n(2m-1) = C_{n-1}(m) + A_{n-1}(m) \\ B_n(2m-1) = B_{n-1}(m), \\ D_n(2m-1) = D_{n-1}(m) + B_{n-1}(m) \end{cases}$$

For RCs:

$$\begin{cases} A_n(2m) = A_{n-1}(m) + C_{n-1}(m), \\ C_n(2m) = C_{n-1}(m) \\ B_n(2m) = B_{n-1}(m) + D_{n-1}(m), \\ D_n(2m) = D_{n-1}(m) \end{cases}$$

3.3 Recursive Algorithm Based on Pseudo Self-Similarity

Combining lemma 3.0.1 and 3.0.2, we get two set of equations.

The first set, for LCs, is

$$\begin{cases} A_n(2m-1) = A_n(m), \\ C_n(2m-1) = C_n(m) + A_n(m) \\ B_n(2m-1) = A_n(m) + B_n(m), \\ D_n(2m-1) = C_n(m) + D_n(m) + A_n(m) + B_n(m) \end{cases}$$

The second set, for RCs, is

$$\begin{cases} A_n(2m) = A_n(m) + C_n(m), \\ C_n(2m) = C_n(m) \\ B_n(2m) = A_n(m) + B_n(m) + C_n(m) + D_n(m), \\ D_n(2m) = C_n(m) + D_n(m) \end{cases}$$

It is trivial to prove that the operations $2m$ and $2m-1$ are sufficient to bring $m = 1$ to any Natural Number. Thus for any wanted m we can generate $t_n(m)$ using these two sets of equations recursively.

One may question the complexity of this algorithm compared to the traditional recursive algorithm starting at the root. The exact complexity is unclear, but for small m's, meaning those near the left side of the tree compared to the total number of nodes on that layer, this algorithm will definitely be more efficient.

Also, applying the Pseudo-Symmetry lemma, basic property 6, the reciprocal of any fraction of the tree will be on the same row and are symmetrical about the middle axis in position. Thus we can find any term near the right side of the tree with equivalent simplicity.

To get an idea of when to use this algorithm, we give a crude approximation of calculating efforts.

Assume the calculation of each addition is 1 unit time, $1u$.

Then for the traditional algorithm, the LC and RC of $\frac{p}{q}$ both take $1u$, since they both need to calculate $p+q$. So in order to find $T(2^{n-1}+m-1)$, we need approximately $(n-1)u$'s.

With our horizontal Algorithm, we need $3u$'s to generate $t_n(2x)$ and $t_n(2x-1)$ from $t_n(x)$. In other words, for a $t_n(m)$, $2^i < m \leq 2^{i+1}$, we need $3(i+1) = 3\lceil \log_2 m \rceil u$'s.

In conclusion, the traditional algorithm is only dependent on n and our algorithm is only dependent on m.

Let $n - 1 > 3\lceil \log_2 m \rceil u$,

$$\frac{n-1}{3} > \log_2 m$$

$$m < 2^{\frac{n-1}{3}}$$

So when $m < 2^{\frac{n-1}{3}}$ our horizontal algorithm should be more efficient.

4 Solving the Calkin-Wilf Tree Non-Recursively

4.1 Horizontal Properties

Taking the horizontal representation of $\frac{An-B}{Cn-D}$, we find some interesting properties that help us to generate any node non-recursively (compared to all existing methods are recursive algorithms).

The properties:

1. $\begin{cases} A_n(m+1) = C_n(m) \\ B_n(m+1) = D_n(m) \end{cases}$
2. $\begin{cases} A_n(2m-1) = A_n(m) \\ B_n(2m-1) = A_n(m) + B_n(m) \end{cases}$
3. $\begin{cases} A_n(2m) = A_n(2m-1) + A_n(2m+1) \\ B_n(2m) = B_n(2m-1) + B_n(2m+1) \end{cases}$
4. $A_n(m)D_n(m) - B_n(m)C_n(m) = 1$

Now proving the properties:

Proposition 4.1. Property 1: $\begin{cases} A_n(m+1) = C_n(m) \\ B_n(m+1) = D_n(m) \end{cases}$

Proof. We give our proof by strong induction.

Base Case $i = 1$.

For all $m \leq 2^1$

$$C_n(1) = 1 = A_n(2)$$

$$D_n(1) = 0 = B_n(2)$$

The lemma holds true for the base case.

Induction Assume the lemma holds true for all $t_n(m)$, $m \leq 2^i$, $i \in \mathbb{N}$

We first extend our lemma to the $(n-1)$th row. According to lemma 3.0.1,

$$\begin{cases} A_{n-1}(m) = A_n(m), \\ C_{n-1}(m) = C_n(m) \\ B_{n-1}(m) = A_n(m) + B_n(m), \\ D_{n-1}(m) = C_n(m) + D_n(m) \end{cases}$$

By our induction hypothesis, $A_n(m+1) = C_n(m)$. Substituting in $A_{n-1}(m) = A_n(m)$ and $C_{n-1}(m) = C_n(m)$, we get $A_{n-1}(m+1) = C_{n-1}(m)$.

Then, we consider $B_{n-1}(m+1) = A_n(m+1) + B_n(m+1)$. Substituting in our induction hypothesis $A_n(m+1) = C_n(m)$ and $B_n(m+1) = D_n(m)$, we get $B_{n-1}(m+1) = C_n(m) + D_n(m) = D_{n-1}(m)$

So now the lemma also holds true for all $t_{n-1}(m)$, $m \leq 2^i$.

Now then choose a x, $\forall x \in \mathbb{N}$, such that $2^i \leq x < 2^{i+1}$.

Consider $t_n(x)$ and $t_n(x+1)$. There are two cases, $x \equiv 1 \pmod 2$ or $x \equiv 0 \pmod 2$.

Case 1: $x \equiv 1 \pmod 2$

Let $m = \frac{x+1}{2}$

Since x is odd, $t_n(x)$ is a LC of the $(n-1)$ row, namely LC of the $t_{n-1}(\frac{x+1}{2} = m)$.

So the $t_n(x+1)$ is the RC of $t_{n-1}(m)$.

According to the LC and RC lemma 3.0.2, $A_n(x+1) = A_{n-1}(m) + C_{n-1}(m) = C_n(x)$, and $B_n(x+1) = B_{n-1}(m) + D_{n-1}(m) = D_n(x)$. So the lemma holds true for this case.

Case 2: $x \equiv 0 \pmod 2$

Let $m = \frac{x}{2}$

Since x is even, $t_n(x)$ is a RC of the $(n-1)$ row, namely RC of the $t_{n-1}(\frac{x}{2} = m)$

So the $t_n(x+1)$ is the LC of $t_{n-1}(m+1)$.

We already proved that the lemma also holds true for the first 2^i terms of the $n-1$ row. So $A_{n-1}(m+1) = C_{n-1}(m)$ and $B_{n-1}(m+1) = D_{n-1}(m)$.

According to the LC and RC lemma 3.0.2,

$$A_n(x+1) = A_{n-1}(m+1)$$

$$B_n(x+1) = B_{n-1}(m+1)$$

$$C_n(x) = C_{n-1}(m)$$

$$D_n(x) = D_{n-1}(m)$$

$$\Rightarrow \begin{cases} A_n(x+1) = C_n(x), \\ B_n(x+1) = D_n(x) \end{cases}$$

The lemma also holds true for this case.

So if the lemma holds true for all $t_n(m)$, $m \leq 2^i$, $i \in \mathbb{N}$, it also holds true for all $2^i \leq x < 2^{i+1}$.

Thus applying strong induction, the lemma is valid for all $m \in \mathbb{N}$

□

Proposition 4.2. Property 2: $\begin{cases} A_n(2m-1) = A_n(m) \\ B_n(2m-1) = A_n(m) + B_n(m) \end{cases}$

Proof. By the Pseudo Self-Similarity property, 3.0.1, we have $\begin{cases} A_{n-1}(m) = A_n(m) \\ B_{n-1}(m) = A_n(m) + B_n(m) \end{cases}$

And by the LC and RC lemma, 3.0.2, we have $\begin{cases} A_{n-1}(m) = A_n(2m-1) \\ B_{n-1}(m) = B_n(2m-1) \end{cases}$

Combining the two set of equations, we get $\begin{cases} A_n(2m-1) = A_n(m) \\ B_n(2m-1) = A_n(m) + B_n(m) \end{cases}$

Which is our property 2.

□

Proposition 4.3. Property 3: $\begin{cases} A_n(2m) = A_n(2m-1) + A_n(2m+1) \\ B_n(2m) = B_n(2m-1) + B_n(2m+1) \end{cases}$

Proof. By $2m$ we know that $t_n(2m)$ is a RC.

By the LC and RC lemma, 3.0.2, we have

$$\begin{cases} A_n(2m) = A_{n-1}(m) + C_{n-1}(m) \\ B_n(2m) = B_{n-1}(m) + D_{n-1}(m) \\ C_n(2m) = C_{n-1}(m) \\ D_n(2m) = D_{n-1}(m) \end{cases}$$

Plug equations 3 and 4 into 1 and 2, we get

$$\textcircled{1} \begin{cases} A_n(2m) = A_{n-1}(m) + C_n(2m) \\ B_n(2m) = B_{n-1}(m) + D_n(2m) \end{cases}$$

And by the Pseudo Self-Similarity Property, 3.0.1, combined with Property 2, 4.2, we have

$$\textcircled{2} \begin{cases} A_{n-1}(m) = A_n(m) = A_n(2m-1) \\ B_{n-1}(m) = A_n(m) + B_n(m) = B_n(2m-1) \end{cases}$$

Combining $\textcircled{1}$ with $\textcircled{2}$ and also Property 1, 4.1, we have

$$\begin{cases} A_n(2m) = A_n(2m-1) + C_n(2m) = A_n(2m-1) + A_n(2m+1) \\ B_n(2m) = B_n(2m-1) + D_n(2m) = B_n(2m-1) + B_n(2m+1) \end{cases}$$

Which is our Property 3.

□

Proposition 4.4. Property 4: $A_n(m)D_n(m) - B_n(m)C_n(m) = 1$

Proof. We give our proof by strong induction.

Base Case $i = 1$.

For all $m \leq 2^0$

$$A_n(1)D_n(1) - B_n(1)C_n(1) = 0 \times 0 - (-1) \times 1 = 1$$

The lemma holds true for the base case.

Induction Assume the lemma holds true for all $t_n(m)$, $m \leq 2^i$, $i \in \mathbb{N}$

We first extend our lemma to the $(n-1)$th row. According to lemma 3.0.1, $\begin{cases} A_{n-1}(m) = A_n(m), \\ C_{n-1}(m) = C_n(m) \\ B_{n-1}(m) = A_n(m) + B_n(m), \\ D_{n-1}(m) = C_n(m) + D_n(m) \end{cases}$

$$\begin{aligned} &A_{n-1}(m)D_{n-1}(m) - B_{n-1}(m)C_{n-1}(m) \\ =&A_n(m)(C_n(m) + D_n(m)) - (A_n(m) + B_n(m))C_n(m) \\ =&A_n(m)D_n(m) - B_n(m)C_n(m) \\ =&1 \end{aligned}$$

So the lemma also holds true for all $t_{n-1}(m)$, $m \leq 2^i$.

Now then choose a x, $\forall x \in \mathbb{N}$, such that $2^i \leq x < 2^{i+1}$.

Consider $t_n(x)$. There are two cases, $x \equiv 1 \pmod 2$ or $x \equiv 0 \mod 2$.

Case 1: $x \equiv 1 \pmod 2$

Let $m = \frac{x+1}{2}$

Since x is odd, $t_n(x)$ is a LC of the $(n-1)$ row, namely LC of the $t_{n-1}(\frac{x+1}{2} = m)$.

According to the LC and RC lemma 3.0.2, $\begin{cases} A_n(x) = A_{n-1}(m), \\ C_n(x) = C_{n-1}(m) + A_{n-1}(m) \\ B_n(x) = B_{n-1}(m), \\ D_n(x) = D_{n-1}(m) + B_{n-1}(m) \end{cases}$

$$\begin{aligned} &A_n(x)D_n(x) - B_n(x)C_n(x) \\ =&A_{n-1}(m)(C_{n-1}(m) + D_{n-1}(m)) - (A_{n-1}(m) + B_{n-1}(m))C_{n-1}(m) \\ =&A_{n-1}(m)D_{n-1}(m) - B_{n-1}(m)C_{n-1}(m) \\ =&1 \end{aligned}$$

(According to our generalization to the $(n-1)$th row.)

So the lemma holds true for this case.

Case 2: $x \equiv 0 \pmod 2$

Let $m = \frac{x}{2}$

Since x is even, $t_n(x)$ is a RC of the $(n-1)$ row, namely RC of the $t_{n-1}(\frac{x}{2} = m)$

According to the LC and RC lemma 3.0.2, $\begin{cases} A_n(x) = A_{n-1}(m) + C_{n-1}(m), \\ C_n(x) = C_{n-1}(m) \\ B_n(x) = B_{n-1}(m) + D_{n-1}(m), \\ D_n(x) = D_{n-1}(m) \end{cases}$

$$\begin{aligned} &A_n(x)D_n(x) - B_n(x)C_n(x) \\ =&(A_{n-1}(m) + C_{n-1}(m))D_{n-1}(m) - (B_{n-1}(m) + D_{n-1}(m))C_{n-1}(m) \\ =&A_{n-1}(m)D_{n-1}(m) - B_{n-1}(m)C_{n-1}(m) \\ =&1 \end{aligned}$$

(According to our generalization to the $(n-1)$th row.)

So the lemma also holds true for this case.

So if the lemma holds true for all $t_n(m)$, $m \leq 2^i$, $i \in \mathbb{N}$, it also holds true for all $2^i \leq x < 2^{i+1}$.

Thus applying strong induction, the lemma is valid for all $m \in \mathbb{N}$

□

4.2 Finding $A_n(m)$

$A_n(m)$ is rather annoying to find. As we have found in the properties, we can give a set of recursive formulas by properties 2 and 3: $\begin{cases} A_n(2m-1) = A_n(m) \\ A_n(2m) = A_n(2m-1) + A_n(2m+1) = A_n(m) + A_n(m+1) \\ A_n(1) = 0 \\ A_n(2) = 1 \end{cases}$

It is trivial to prove that these set of equations are enough to generate every A_n term. However, **the existence of a closed form is still unknown**. We at least failed to find any.

We solve for the exact value of $A_n(m)$ for a given m by partitioning the $A_n(m)$ into a sequence of

$$A_n(m) = c_1A_n(2^1) + c_2A_n(2^2) + ... + c_iA_n(2^i)$$

The proof for the feasibility of such a partition for any $A_n(m)$ is trivial by strong induction.

And the reason that we partition $A_n(m)$ into $A_n(2^i)$'s is the following property

Lemma 4.0.1. $A_n(2^i) = i, \forall i \in \mathbb{N}$

Proof. First, we should know that $A_n(2^i + 1)$, $\forall i \in \mathbb{N}$. This is trivial to prove by induction and Property 2, 4.2, that $A_n(2m - 1) = A_n(m)$.

Then, applying induction.

Base Case $i = 1$

$$A_n(2^1) = 1$$

Induction Assume $A_n(2^i) = i$

Consider the term $A_n(2^{i+1})$

By Property 3, 4.3, and Property 2, 4.2

$$\begin{aligned} A_n(2^{i+1}) =& A_n(2^{i+1} - 1) + A_n(2^{i+1} + 1) \\ =& A_n(2^i) + 1 \\ =& i + 1 \end{aligned}$$

Thus by induction, finishing the proof. □

So, $A_n(m) = c_1 A_n(2^1) + c_2 A_n(2^2) + ... + c_i A_n(2^i)$ is equvilent to

$$A_n(m) = c_1 \times 1 + c_2 \times 2 + ... + c_i \times i$$

4.3 Non-Recurssive Algorithm based on known $A_n(m)$

Now, by Property 4, 4.4, we have a linear Diophantine equation $A_n(m)D_n(m) - B_n(m)C_n(m) = 1$. Slightly tweaking it with Property 1, 4.1, we have

$$A_n(m)D_n(m) - A_n(m + 1)B_n(m) = 1$$

When we have $A_n(m)$ and $A_n(m + 1)$, we can already produce a unlimited series of possible solutions for $B_n(m)$ and $D_n(m)$ by various well known methods, such as the Magic Box or a variation of Euclidean Algorithm. The solutions will be in the form $A_n(m)(D_n(m)+kA_n(m+1))-A_n(m+1)(B_n(m)+kA_n(m)) = 1$, $\forall k \in \mathbb{Z}$

Definition 4.1. $(X(m), Y(m))$: Denote the least non-negative integer pair of (x, y) solutions for $A_n(m)x - A_n(m+1)y = 1$ as $(X(m), Y(m))$.

We found that $D_n(m)$ and $B_n(m)$ are not random solutions to the linear diophantine equation. Instead, $\begin{cases} D_n(m) = X(m) + kC_n(m) \\ B_n(m) = Y(m) + kA_n(m) \end{cases}$, where $k = \lceil \log_2 m \rceil - 1$ and $m \geq 2$. In other words, with $A_n(m)$ and $C_n(m) = A_n(m+1)$, it is enough to solve the entire $t_n(m)$.

Now to prove the Theorem.

Theorem 4.1. $\begin{cases} D_n(m) = X(m) + kC_n(m) \\ B_n(m) = Y(m) + kA_n(m) \end{cases}$, where $k = \lceil \log_2 m \rceil - 1$ and $m \geq 2$

Proof. We give our proof by strong induction. Let $2^i < m \leq 2^{i+1}$

Base Case $i = 0$.

Only $m = 2$ satisfies $2^0 < m \leq 2^1$

$A_n(2) = 1$, $C_n(2) = 1$,

Solving $A_n(2)X(2) - C_n(2)Y(2) = X(2) - Y(2) = 1$

$$(X(2), Y(2)) = (1, 0)$$

And since $D_n(2) = 1$, $B_n(2) = 0$, $k = \lceil \log_2 2 \rceil - 1 = 0$

So $\begin{cases} D_n(2) = 1 = X(2) + kC_n(2) \\ B_n(2) = 0 = Y(2) + kA_n(2) \end{cases}$

The Theorem is valid for $i = 0$

Induction Assume the theorem holds true for all m, $2^i < m \leq 2^{i+1}$

Which $k = \lceil \log_2 m \rceil - 1 = i + 1 - 1 = i$

Now then choose a x, $\forall x \in \mathbb{N}$, such that $2^i < x \leq 2^{i+1}$.

Note that:

$$\lceil \log_2 x \rceil - 1 = i + 1$$

First we want to solve a useful relationship. Say $\alpha X - \beta Y = 1$, (X, Y) the minimum non-negative pair of solutions as the definition 4.1.

We know that $\alpha(X + Y) - (\beta + \alpha)Y = 1$

Since (X, Y) is the minimum non-negative pair, at least one of the two $Y - \alpha < 0$ and $X - \beta < 0$ hold true. And because the signs of any (x,y)solution to $ax - by = 1$,with $a, b \in \mathbb{N}$, must be the same, both $Y - \alpha < 0$ and $X - \beta < 0$.

And thus $(X + Y) - 1 \times (\beta + \alpha) < 0$, the pair $(X + Y, Y)$ is the minimum non-negative solution to the new $\alpha x - (\beta + \alpha)y = 1$ equation.(1)

An analogous relationship would be the pair $(X, Y + X)$ is the minimum non-negative solution to the new $(\alpha + \beta)x - \beta y = 1$ equation.(2)

Now moving back to the subject, consider $t_n(x)$. According to the Recursive Algorithm Based on Pseudo Self-Similarity, 3.3, we can generate two terms with $t_n(x)$, namely $t_n(2x - 1)$ and $t_n(2x)$.

$t_n(2x - 1)$ By 3.3, we have

$$\begin{cases} A_n(2x-1) = A_n(x), \\ C_n(2x-1) = C_n(x) + A_n(x) \\ B_n(2x-1) = A_n(x) + B_n(x), \\ D_n(2x-1) = C_n(x) + D_n(x) + A_n(x) + B_n(x) \end{cases}$$

Plugging it in:

$$\begin{aligned} &A_n(2x-1)D_n(2x-1) - C_n(2x-1)B_n(2x-1) \\ =&A_n(x)(C_n(x) + D_n(x) + A_n(x) + B_n(x)) - (C_n(x) + A_n(x))(A_n(x) + B_n(x)) = 1 \end{aligned}$$

Based on the relationship (1) we proved, $X(2x - 1) = X(x) + Y(x)$, $Y(2x - 1) = Y(x)$

We know by the Induction Hypothesis that $X(x) = D_n(x) - iC_n(x)$ and $Y(x) = B_n(x) - iA_n(x)$.

So verifying the theorem:

$$\begin{aligned}
D_n(2x-1) - X(2x-1) =& (C_n(x) + D_n(x) + A_n(x) + B_n(x)) - (D_n(x) - iC_n(x) + B_n(x) - iA_n(x)) \\
=& (i+1)(A_n(x) + C_n(x)) \\
=& (i+1)C_n(2x-1) \\
=& (\lceil \log_2 x \rceil - 1)C_n(2x-1) \\
B_n(2x-1) - Y(2x-1) =& (A_n(x) + B_n(x)) - (B_n(x) - iA_n(x)) \\
=& (i+1)A_n(x) \\
=& (i+1)A_n(2x-1) \\
=& (\lceil \log_2 x \rceil - 1)A_n(2x-1)
\end{aligned}$$

The theorem holds for $t_n(2x-1)$.

Case 2: $t_n(2x)$ By 3.3, we have

$$\begin{cases}
A_n(2x) = A_n(x) + C_n(x), \\
C_n(2x) = C_n(x) \\
B_n(2x) = A_n(x) + B_n(x) + C_n(x) + D_n(x), \\
D_n(2x) = C_n(x) + D_n(x)
\end{cases}$$

Plugging it in:

$$\begin{aligned}
& A_n(2x)D_n(2x) - C_n(2x)B_n(2x) \\
=& (A_n(m) + C_n(m))(C_n(x) + D_n(x)) - C_n(x)(A_n(x) + B_n(x) + C_n(x) + D_n(x)) = 1
\end{aligned}$$

Based on the relationship (2) we proved, $X(2x) = X(x)$, $Y(2x-1) = Y(x) + X(x)$

We know by the Induction Hypothesis that $X(x) = D_n(x) - iC_n(x)$ and $Y(x) = B_n(x) - iA_n(x)$.

So verifying the theorem:

$$\begin{aligned}
D_n(2x) - X(2x) =& (C_n(x) + D_n(x)) - (D_n(x) - iC_n(x)) \\
=& (i+1)C_n(x) \\
=& (i+1)C_n(2x) \\
=& (\lceil \log_2 x \rceil - 1)C_n(2x) \\
B_n(2x-1) - Y(2x-1) =& (A_n(x) + B_n(x) + C_n(x) + D_n(x)) - (D_n(x) - iC_n(x) + B_n(x) - iA_n(x)) \\
=& (i+1)(A_n(x) + C_n(x)) \\
=& (i+1)A_n(2x) \\
=& (\lceil \log_2 x \rceil - 1)A_n(2x)
\end{aligned}$$

The theorem also holds for $t_n(2x)$.

It is trivial to prove that the two operations $(2x)$ and $(2x-1)$ can cover all numbers of $(2^{i+1}, 2^{i+2}]$ with $(2^i, 2^{i+1}]$.

Thus applying strong induction, the Theorem holds for all i.

□

5 Conjectures and Observations

- (Closed Form Conjecture) There exists a closed form solution to

$$\begin{cases}
A_n(2m-1) = A_n(m) \\
A_n(2m) = A_n(2m-1) + A_n(2m+1) = A_n(m) + A_n(m+1) \\
A_n(1) = 0 \\
A_n(2) = 1
\end{cases}$$

- (The Partition Conjecture) Any term $A_n(m)$ can be partitioned into $xA_n(2^i) + yA_n(2^{i+1})$, where $x = A_n(\alpha)$, $y = A_n(\beta)$, $\alpha, \beta \leq 2^{\lceil \log_2 m \rceil - 2}$. The actual pattern is more orderly yet harder to formulate.

- (An Improved $A_n(m)$ Equation) $A_n(2^i - k) = i \times A_n(k+2) - B_n(k+2)$

6 Acknowledgements

I would like to thank my math teacher for his support on this project. I would also like to thank Candace Chiang and the PROMYS summer program for the inspiration of this topic.

6.1 My Partner's Work

My partner Jiachen Xu's work mainly includes:

- Discovering Property 1, 4.1, along with the proof.
- Discovering Property 2, 4.2.
- Discovering Property 3, 4.3, along with an induction proof. (Not given in this paper due to complexity)
- Noting the Improved $A_n(m)$ Equation.
- And also some exploration on the Binary representation that didn't come into use.

7 References

References

[1] N. Calkin and H. S. Wilf. "Recounting the Rationals". In: *American Math Monthly* 107.4 (2000), pp. 360–363.

[2] Candace Chiang, the Participant, and William Zhang. "Counting Rationals". Unpublished lab paper. 2015.

[3] D. E. Knuth. "Problem 10906". In: *American Math Monthly* 110.7 (2003). Solution by Moshe Newman, pp. 642–643.