Trees and Meta-Fibonacci Sequences

Abraham Isgur, David Reiss, and Stephen Tanny

Department of Mathematics
University of Toronto, Ontario, Canada

abraham.isgur@utoronto.ca, david.reiss@utoronto.ca, tanny@math.toronto.edu

Submitted: Apr 3, 2009; Accepted: Oct 21, 2009; Published: Oct 31, 2009
Mathematics Subject Classification: 05A15, 11B37, 11B39, 05C05

Abstract

For $k > 1$ and nonnegative integer parameters a_p, b_p, $p = 1..k$, we analyze the solutions to the meta-Fibonacci recursion $C(n) = \sum_{p=1}^{k} C(n - a_p - C(n - b_p))$, where the parameters a_p, b_p, $p = 1..k$ satisfy a specific constraint. For $k = 2$ we present compelling empirical evidence that solutions exist only for two particular families of parameters; special cases of the recursions so defined include the Conolly recursion and all of its generalizations that have been studied to date. We show that the solutions for all the recursions defined by the parameters in these families have a natural combinatorial interpretation: they count the number of labels on the leaves of certain infinite labeled trees, where the number of labels on each node in the tree is determined by the parameters. This combinatorial interpretation enables us to determine various new results concerning these sequences, including a closed form, and to derive asymptotic estimates. Our results broadly generalize and unify recent findings of this type relating to certain of these meta-Fibonacci sequences. At the same time they indicate the potential for developing an analogous counting interpretation for many other meta-Fibonacci recursions specified by the same recursion for $C(n)$ with other sets of parameters.

1 Introduction

In this paper all values are integers. For $k > 1$ and nonnegative parameters a_p, b_p, $p = 1..k$, consider the general meta-Fibonacci (also called "self-referencing" or "nested") homogeneous recursion

$$C(n) = \sum_{p=1}^{k} C(n - a_p - C(n - b_p)) \tag{1.1}$$

Many well-known meta-Fibonacci recursions, with appropriate initial conditions, are special cases of (1.1), which we often write as $(a_1, b_1 : a_2, b_2 : \cdots : a_k, b_k)$. For example, the

Hofstadter recursion [12] is $(0, 1 : 0, 2)$ while the Conolly recursion [6] is $(0, 1 : 1, 2)$. We call the sequences that appear as solutions to meta-Fibonacci recursions *meta-Fibonacci sequences*.

In recent years various special cases of (1.1), together with alternative sets of initial conditions, have been analyzed. See, for example, [1], [2], [3], [4], [5], [6], [7], [11], [14], [15], [18], [20]. These contributions illustrate the very wide range of behavior that can be exhibited by meta-Fibonacci sequences that derive from (1.1). Some sequences, like the one defined by Conolly, are very well behaved, with discernable and provable structure. Others, including the Hofstadter sequence, appear to be quite chaotic, but nonetheless display some evidence of structural regularities [15]. Still others are wild with no hint of any structure but nonetheless appear to remain well defined for all n (for example, the W-sequence originally defined by Hofstadter that is discussed in [2]).

For a given set of initial conditions (1.1) may not have a solution, that is, for some index n one or more of the arguments of C on the right hand side of (1.1) is no longer a positive integer so the recursion cannot be evaluated at this point. In this case we say that the sequence "dies" at n_0 if n_0 is the smallest index for which $n_0 - a_p - C(n_0 - b_p) \leqslant 0$ for one or more values of p.

In this paper we focus on those values of the parameters a_p, b_p in (1.1) and the initial conditions for which the resulting meta-Fibonacci sequence is "slow growing" (or simply slow), by which we mean that the sequence is monotone non-decreasing, and successive terms differ by 0 or 1.[1] Clearly such sequences are entirely determined by their frequency function, that is, the number of times that they hit each positive integer.

In Table 1.1 we illustrate the initial 19 terms of three meta-Fibonacci sequences derived from special cases of (1.1). The first sequence is generated by the Conolly recursion (0,1:1,2) with the two initial conditions 1,2 (the initial conditions highlighted in bold). The second is generated by a close relative of the Conolly recursion defined in [20], together with the three initial conditions 1,1,2.[2] Both of these sequences are slow. Notice that the second sequence is the same as the first except for the extra repetition of the powers of 2. This kind of kinship among the solutions to certain closely related meta-Fibonacci recursions derived from (1.1) is well-known (see, for example, [4], [5], and [14]); it will also be a feature of some of the new results for other special cases of (1.1) that we introduce later in this paper.

The third meta-Fibonacci sequence in Table 1.1 is not slow. It is also a solution to the same recursion (1,1 : 2,2) as the second, but this time the sequence satisfies a different set of three initial conditions, namely, 2,1,1.

Recently it has been shown in [3], [14] and [7] that for any nonnegative s and $k > 1$ there is a fascinating connection between certain labeled infinite trees and slow growing meta-Fibonacci sequences that arise as solutions to the recursions in the families $(s, 1 : 1 + s, 2 : 2 + s, 3 : \cdots : k - 1 + s, k)$ for $k > 1$ and $(s, 1 : 2 + s, 3)$ respectively, each of which is a special case of (1.1). In each case the meta-Fibonacci sequence counts the

[1] This terminology is due to Frank Ruskey.

[2] It is readily seen that three initial conditions are required to define the recursion (1,1:2,2) whereas only two initial conditions are required for the Conolly recursion (0,1 : 1,2).

Table 1.1: Examples of meta-Fibonacci sequences.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
(0,1:1,2)(n)	**1**	**2**	2	3	4	4	4	5	6	6	7	8	8	8	8	9	10	10	11
(1,1:2,2)(n)	**1**	**1**	**2**	2	2	3	4	4	4	4	5	6	6	7	8	8	8	8	8
(1,1:2,2)(n)	**2**	**1**	**1**	3	3	3	2	4	6	4	4	5	4	8	9	6	7	8	8

number of leaves in particular subtrees of the infinite tree that is related to the sequence. This graphical interpretation for these meta-Fibonacci sequences provides an elegant, intuitive basis for proving certain of their properties, such as that they are slow growing, and for deriving their generating functions and asymptotic behavior. It also provides a clear understanding for the very close relationship among the sequences in each family for different values of the parameter s, where we find that the sequences in each family are identical except for the number of repetitions of powers of k or 2 respectively. [3]

Other meta-Fibonacci sequences also can be related to infinite trees in an analogous manner to the one developed in [14] and [3]. For example, for s nonnegative, consider the sequence $g_s(n)$ defined by the (non-homogeneous) meta-Fibonacci recursion

$$g_s(n) = g_s(n - s - g_s(n-1)) + 1 \tag{1.2}$$

with initial conditions $g_s(n) = 1$ for $n = 1, 2, .., s+1$. See Table 1.2 for the first few values of $g_s(n)$ for $s = 0, 1, 2$. The initial data suggests that each of these sequences is slow, which in fact turns out to be the case for all values of s. The special case of (1.2) with $s = 0$ is one of the earliest known meta-Fibonacci recursions, appearing in [8]; it is an example of the rare instance when a meta-Fibonacci recursion has a simple closed form solution, namely, $g_0(n) = \lfloor \frac{\lfloor\sqrt{8n}\rfloor+1}{2} \rfloor$.

Table 1.2: The sequences $g_s(n)$, s=0,1 and 2.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$s = 0$	**1**	2	2	3	3	3	4	4	4	4	5	5	5	5	5	6	6	6	6	6
$s = 1$	**1**	**1**	2	2	2	3	3	3	3	4	4	4	4	4	5	5	5	5	5	5
$s = 2$	**1**	**1**	**1**	2	2	2	2	3	3	3	3	3	4	4	4	4	4	4	5	5

To see how $g_s(n)$ relates to a labeled tree, first we define a tree structure G_s that consists of an infinite number of rooted trees joined together at their respective roots. To create G_s first we join an infinite chain of nodes $\{v_i\}$ with node v_{i+1} connected to node v_i for $i = 1, 2, 3, \ldots$. We distinguish these nodes, calling them s-nodes; in Figure 1.1 these are indicated by the square boxes. The i^{th} s-node is the root of a chain with i (ordinary) nodes below it. In addition, there is an extra ordinary node connected to the first s-node at the very beginning of G_s.

[3]For example, the two slow sequences in Table 1.1 are solutions to the recursion $(s, 1 : 1+s, 2)$, which is $(s, 1 : 1+s, 2 : 2+s, 3 : \cdots : k-1+s, k)$ for $k = 2$, for $s = 0$ and $s = 1$ respectively. The graphical interpretation of these sequences explains why they are essentially the same except for the occurrence of one additional repetition at every power of 2 in the sequence for $s = 1$.

We label the nodes of G_s with the positive integers $1, 2, 3, ..$ in the following way: the initial (ordinary) node receives the label 1, then in turn each s-node, starting with the first, receives the smallest s consecutive labels not yet used (note that if $s = 0$ then the s-nodes do not receive any labels). Once an s-node receives its labels, then all of its descendants are labeled, each with a single label, in order from top to bottom, with the smallest available labels. Then the next s-node is labeled, together with its descendants, and so on. Figure 1.1 shows the initial portion of G_s for $s = 2$.

The leaves of G_s consist of the initial node in G_s and the last node in each of the chains that descend from the s-nodes. It is readily verified that $g_s(n)$ counts the number of leaves in G_s that have a label that is less than or equal to n.

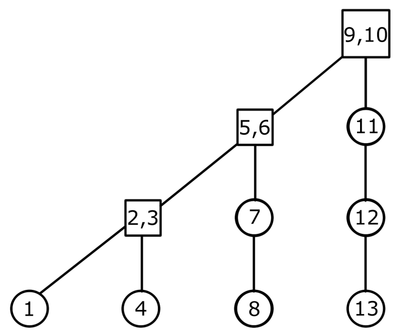

Figure 1.1: The tree G_2 up to label 13. The s-nodes are drawn as squares and the j-nodes as circles.

It is natural to ask if there is some way to identify other meta-Fibonacci sequences that might be related to trees in this way, and how to determine the tree structure that would apply in such instances; that is, we look for some unifying method that would help to identify how to relate the solutions of certain classes of meta-Fibonacci recursions to different labeled infinite trees. We turn our attention to this question, focusing on slow growing sequences since these appear to be the most likely candidates for such a relationship.

In the next section we experiment with the recursion (1.1) for $k = 2$. Furthermore, we impose a constraint on the parameters, discussed below, that appears a priori to be potentially interesting. Through this process we identify two new families of meta-Fibonacci sequences that are promising candidates for a combinatorial interpretation and on which we focus in the balance of the paper. In Sections 3 and 4 we show that indeed these meta-Fibonacci sequences are related to infinite trees; in fact, it turns out that they are related to the same infinite trees identified in [18] and [3] but with modified labeling schemes. As such, our findings generalize and unify earlier known results relating meta-Fibonacci sequences and infinite trees. We apply these counting interpretations in Section 5 to derive a closed form for the solution to one of the recursions and an asymptotic estimate to the solution for the other. In Section 6 we comment on the role of the initial conditions in determining the properties of the meta-Fibonacci sequences that derive from the recursions. We conclude in Section 7 with a brief discussion of some potential directions for further inquiry in this area.

2 A Brief Empirical Interlude

Recall from Section 1 that for any nonnegative s and appropriate initial conditions both the recursions $(s, 1 : 1 + s, 2)$ and $(s, 1 : 2 + s, 3)$ have a solution that is a slow growing sequence that counts the number of leaves in certain infinite labeled trees. Observe that each of these recursions has four parameters that, in the notation of (1.1), satisfy the relation $a_1 + b_2 = a_2 + b_1$.

Motivated by this observation we investigate more generally the nature of the solutions, if any, to the recursion (1.1) with $k = 2$ and $a_1 + b_2 = a_2 + b_1$. Without loss of generality assume that $a_1 \leqslant a_2$. To begin we set $a_1 = 0$ and test all the different possible recursions for $b_2 \leqslant 20$.

To generate the solution for each such recursion we need to provide at least b_2 initial conditions. The initial conditions we adopt are inspired by the previous work in [14] and [3] for the recursions $(s, 1 : 1 + s, 2)$ and $(s, 1 : 2 + s, 3)$ respectively. In both cases the initial conditions that yielded a solution with a combinatorial interpretation consisted of a string of consecutive 1s followed by a single 2. For the recursion $(s, 1 : 1 + s, 2)$ there were $s + 1$ 1s; for the recursion $(s, 1 : 2 + s, 3)$ there were $s + 2$ 1s.

For the present more general situation we adapt the above pattern by taking as initial conditions a string of $(b_2 - 1)$ 1s followed by a single 2. Using these initial conditions we attempt to generate a solution sequence for each recursion up to 1 million terms, to test if the sequence appears to live and to examine its properties.

Our findings from these calculations are summarized in Table 2.1. We indicate by the letter S that the recursion with that particular set of parameters generates a sequence to 1 million terms, so appears to have a solution; a * indicates that there is no solution (the sequence dies). The results are striking: the only recursions for which a solution appears to exist have the parameters $(0, j : j, 2j)$ and $(0, j : 2j, 3j), j > 0$. Further, from the data, all of the solution sequences appear to be slow growing. Note that for $j = 1$ these recursions are precisely those already known to us from [14] and [3] for the special case $s = 0$.

Table 2.1: Recursions with solutions (S): $(0, b_1 : a_2, b_2)$, $b_2 = a_2 + b_1$ with initial conditions $1, \ldots, 1, 2$

b_1/b_2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	*	S	S	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
2		*	*	S	*	S	*	*	*	*	*	*	*	*	*	*	*	*	*	*
3			*	*	*	S	*	*	S	*	*	*	*	*	*	*	*	*	*	*
4				*	*	*	*	S	*	*	*	S	*	*	*	*	*	*	*	*
5					*	*	*	*	*	S	*	*	*	*	S	*	*	*	*	*
6						*	*	*	*	*	*	S	*	*	*	*	*	S	*	*
7							*	*	*	*	*	*	*	S	*	*	*	*	*	*
8								*	*	*	*	*	*	*	*	S	*	*	*	*
9									*	*	*	*	*	*	*	*	*	S	*	*
10										*	*	*	*	*	*	*	*	*	*	S

We repeat the above exercise allowing non-zero values for the parameter a_1. This time we test all possible values of the parameters a_1, b_1, and b_2 up to 5, with $a_2 =$

$a_1 + b_2 - b_1$. For the initial conditions once again we take a string of $1s$ followed by a single 2. The precise number of $1s$ depends on the relations among the parameters; we take the minimum number required to allow the recursion to begin to compute the solution. For each recursion we attempt to generate 1 million terms of the solution sequence.

Table 2.2: Parameters for recursion $(a_1, b_1 : a_2, b_2)$: $a_1 + b_2 = a_2 + b_1, a_1 > 0$. Recursions with solutions indicated by S.

1 1 1 1	*	2 1 6 5	*	3 2 6 5	*	4 4 4 4	*
1 1 2 2	S	2 2 2 2	*	3 3 3 3	*	4 4 5 5	*
1 1 3 3	S	2 2 3 3	*	3 3 4 4	*	4 5 4 5	*
1 1 4 4	*	2 2 4 4	S	3 3 5 5	*	5 1 5 1	*
1 1 5 5	*	2 2 5 5	*	3 4 3 4	*	5 1 6 2	S
1 2 1 2	*	2 3 2 3	*	3 4 4 5	*	5 1 7 3	S
1 2 2 3	*	2 3 3 4	*	3 5 3 5	*	5 1 8 4	S
1 2 3 4	S	2 3 4 5	*	4 1 4 1	*	5 1 9 5	*
1 2 4 5	*	2 4 2 4	*	4 1 5 2	S	5 2 5 2	*
1 3 1 3	*	2 4 3 5	*	4 1 6 3	S	5 2 6 3	*
1 3 2 4	*	2 5 2 5	*	4 1 7 4	*	5 2 7 4	S
1 3 3 5	S	3 1 3 1	*	4 1 8 5	*	5 2 8 5	*
1 4 1 4	*	3 1 4 2	S	4 2 4 2	*	5 3 5 3	*
1 4 2 5	*	3 1 5 3	S	4 2 5 3	*	5 3 6 4	*
1 5 1 5	*	3 1 6 4	*	4 2 6 4	S	5 3 7 5	*
2 1 2 1	*	3 1 7 5	*	4 2 7 5	*	5 4 5 4	*
2 1 3 2	S	3 2 3 2	*	4 3 4 3	*	5 4 6 5	*
2 1 4 3	S	3 2 4 3	*	4 3 5 4	*	5 5 5 5	*
2 1 5 4	*	3 2 5 4	S	4 3 6 5	*		

The results of this second set of calculations are highlighted in Table 2.2. Here too the letter S indicates that for a particular set of the parameters the recursion generates a sequence to 1 million terms while a * indicates that there is no solution. Once again the results are intriguing: with just one exception, namely, $(1, 3 : 3, 5)$[4], the only recursions with an apparent solution have the parameters $(s, j : j + s, 2j)$ or $(s, 1 : 2 + s, 3)$, where $s > 0$ and $j > 0$; further, all of these solutions are slow growing sequences![5]

The recursion $(s, j : s + j, 2j)$ is a natural generalization of the recursion $(0, j : j, 2j)$ identified in the first set of calculations with "shift" parameter s;[6] in addition, it is a

[4]Subsequently, additional investigation over a greater range of values for the parameters has lead to the identification of another exception, the recursion $(2, 5 : 4, 7)$. Both of these exceptional recursions have essentially the same slow solution, namely, the ceiling function for $\frac{n}{2}$. Further discussion of these exceptions in the context of another family of recursions with different initial conditions would take us too far afield here and will appear in a forthcoming communication.

[5]Unfortunately, this result is dependant on the choice of initial conditions. If we do not restrict ourselves to the initial conditions of 1s followed by a single 2, more exceptions arise which we have not yet been able to categorize. In this paper we restrict our focus to the aforementioned sequences and their close relatives.

[6]Unlike the case for $j = 1$, for fixed $j > 1$ there does not appear to be any simple relationship among the solutions for different values of $s > 0$.

natural generalization of the recursion (s,1 : 1+s,2) studied in [14] and for which combinatorial interpretations are known. For ease of reference we describe the family of recursions $(s, j : j + s, 2j), s \geqslant 0$ and $j > 0$, as well as its natural k-term analogue $(s, j : j+s, 2j : 2j+s, 3j : \cdots : (k-1)j+s, kj)$, to be of type $[0, j : j, 2j]$ (note the square brackets to denote the entire family, whereas round brackets denote a specific recursion in the family).

It will be convenient to be able to refer to individual terms in the solution to the recursion $(s, j : j + s, 2j : 2j + s, 3j : \cdots : (k-1)j + s, kj)$ with particular values of s, j and k. For this reason we write out the recursion explicitly in traditional function notation as follows:

$$A_{s,j,k}(n) = \sum_{p=1}^{k} A_{s,j,k}(n - (p-1)j - s - A_{s,j,k}(n - pj)) \tag{2.1}$$

A combinatorial interpretation is also known for $(s, 1 : 2 + s, 3)$ (see [3]), which generalizes the recursion $(0, j : 2j, 3j)$ identified in the first set of calculations by introducing the shift parameter s, but only for $j = 1$. [7] We say that recursions in the family $(s, j : s + 2j, 3j), s \geqslant 0$ and $j > 0$ are of type $[0, j : 2j, 3j]$. In this case too it will be convenient to introduce a traditional notation for the generic recursion of this type. Anticipating our results we adopt the following:

$$B_{s,j}(n) = B_{s,j}(n - s - B_{s,j}(n-j)) + B_{s,j}(n - 2j - s - B_{s,j}(n-3j)) \tag{2.2}$$

Our empirical evidence suggests that there are no solutions for the analogue of (2.2) with $k > 2$. For both (2.1) and (2.2), we drop the subscripts when the meaning is clear.

The data described above places a clear focus on recursions that are natural generalizations of those for which a combinatorial interpretation is already known. In so doing it points strongly to the possibility that there might be some combinatorial interpretation involving trees for these more general recursions. As we shall see in the next two sections, this turns out to be the case.

3 Combinatorial Interpretation for the Family [0,j : j,2j]

The structure of (2.1) is the same for all values of the parameter j. Since a combinatorial interpretation is known for the case $j = 1$ in terms of infinite labeled binary and k-ary trees it seems reasonable to expect that the sequences derived from the more general recursion (2.1) for $j > 1$ might have an analogous interpretation in terms of similar infinite trees. Whatever tree works for general j must reduce to the k-ary tree for the special case $j = 1$.

[7] For $j > 1$ and $s > 0$ the recursion $(s, j : s + 2j, 3j)$ does not appear to ever have a solution for the initial conditions that we assumed. In Section 4 we will have more to say about this recursion with different initial conditions.

This observation led us to focus on introducing an alternate labeling scheme on the infinite k-ary tree used in [18] that incorporates the inclusion of the general parameter j but reduces when $j = 1$ to the usual labeled k-ary tree. But this means that we can no longer count nodes of the labeled tree, since the number and position of the nodes in the tree will not vary with alternate values of j. So we must count something else: that something is the labels on the nodes. With this by way of motivation, we now describe the construction and labeling process that we have discovered for the k-ary tree that provides the basis for the counting interpretation that we seek for sequences from (2.1).

Let $T_{s,j,k}$ with $s \geqslant 0, j \geqslant 1, k \geqslant 2$ denote an infinite k-ary tree. All the nodes on the absolute left (except for the bottom leftmost node) are *s-nodes* containing s positive integer labels; all other nodes are *j-nodes* containing j positive integer labels. (Note that in [14], [18], nodes analogous to the s-nodes are referred to as *super nodes*). We refer to *levels* in the tree as follows: the bottom level consists of the j-nodes with no *children*; we also call these j-nodes leaves. The parents of the leaves are at the penultimate level and are called penultimate nodes. We refer to the successive levels above the penultimate level as the *third level*, *fourth level*, and so on.

Label each of the nodes of $T_{s,j,k}$ in pre-order, starting from 1. Each j-node (respectively s-node) receives j (respectively s) consecutive numbers and no number is used more than once. Define the finite tree $T_{s,j,k}(n)$ to be that portion of $T_{s,j,k}$ consisting of only those nodes (both s-nodes and j-nodes) and labels up to the label n and the node containing it (this last node containing n may be only partially filled in). See Figure 3.1 where $s = 2, j = 3, k = 2$ and $n = 89$.

For $m > 1$ we define the m^{th} k-ary subtree of $T_{s,j,k}$ or $T_{s,j,k}(n)$ to be the subtree consisting of the $(m-1)^{st}$ s-node together with all of its descendants on lower levels of the tree; for $m = 1$ the first k-ary subtree is the initial j-node that is the first leaf.

Let $R_{s,j,k}(n)$ be the number of labels in the leaves of $T_{s,j,k}(n)$. We call the resulting sequence $R(n)$ a *label counting sequence*. See Table 3.1 for the first twenty terms of the sequence $R_{2,3,2}(n)$ corresponding to Figure 3.1.

In the following we fix the parameters s, j, k; for convenience, where there is no confusion we drop the subscripts on $T_{s,j,k}$, $T_{s,j,k}(n)$ and $R_{s,j,k}(n)$, writing $T, T(n)$ and $R(n)$ respectively. We do similarly for the terms of the recursion (2.1), writing $A(n)$ in place of $A_{s,j,k}(n)$.

Table 3.1: The sequence $R_{2,3,2}(n), n = 1..20$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$R(n)$	1	2	3	3	3	4	5	6	6	6	6	6	6	7	8	9	10	11	12	12

In what follows we establish several important structural properties of the trees T and $T(n)$. Unless otherwise noted we assume that the final label n in $T(n)$ is in a node further along in pre-order than the third k-ary subtree. It is readily verified that this assumption requires $n > 2s + k^2j + kj - j$. For such n we show combinatorially that $R(n)$ and $A(n)$ both satisfy the recursion (2.1). From this we conclude that $R(n) = A(n)$ for all n so long

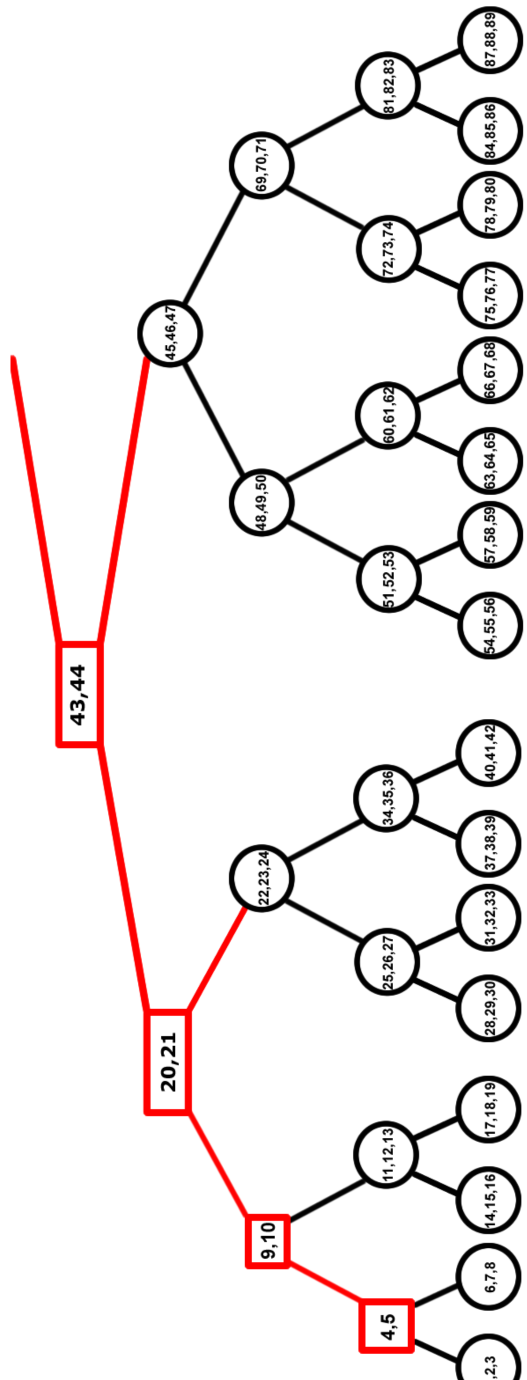

Figure 3.1: $T_{2,3,2}(89)$. The s-nodes are drawn as squares and the j-nodes as circles.

as we take for initial conditions that $R(n) = A(n)$ for n up to the end of the third k-ary subtree (that is, $n \leqslant 2s + k^2j + kj - j$). We say that these initial conditions for (2.1) *follow the tree*.

The condition that $n > 2s + k^2j + kj - j$ is made necessary because we seek a combinatorial argument that applies to all the values of n under consideration. In evaluating $A(n)$ the recursion (2.1) *looks back* to terms much earlier in the sequence. So in order to apply the calculation from the recursion to match the sequence $R(n)$ that counts labels in the tree, it is necessary for a sufficiently large portion of the tree prior to the label n to be defined.[8]

The *pruning process*, analogous to the one introduced in [14], yields a property of T that is central to its connection to the recursion (2.1). Delete all the leaves of T. In the resulting tree convert the first s-node (the one now in the bottom left position) to a j-node (note that doing so just means changing the number of labels in it from s to j). Denote this new tree by T^*. It is evident from the structure of T that T^* and T are the same up to renumbering of the labels in T^*. That is, we have the following:

Lemma 3.1. *The k-ary trees T^* and T are the same, up to renumbering of the labels.*

Likewise, we define $T^*(n)$ to be the *pruned* version of $T(n)$: take $T(n)$, remove all its leaves, replace the first s-node with a j-node, and relabel $T^*(n)$ in pre-order. Note that $T^*(n)$ is another k-ary tree. Denote by $R^*(n)$ the number of leaf labels of $T^*(n)$.

We apply this pruning process to establish certain counting results. In order to clarify and simplify our discussion, but in an abuse of notation, we describe the labels on the tree $T^*(n)$ that results from the pruning process in terms of the original labels of $T(n)$. That is, in the course of the argument we speak of *deleting* or *shifting* around existing labels of $T(n)$. As a result the labels of $T^*(n)$ consist of a subset of the labels of $T(n)$ and we do not bother to relabel $T^*(n)$ from 1 on. For example, if we delete the label 1 in the first spot of $T(n)$, then move the label 11 into the first spot formerly occupied by the label 1, we will describe this as putting an "11" in place of 1, or in $1's$ spot, and deleting the eleventh spot (formerly occupied by the label 11). For our present purposes this approach is convenient and acceptable: as we shall soon see, our only interest is in counting the number of labels in the leaves of $T^*(n)$. The precise numbers that comprise these labels are irrelevant, all that matters is the quantity of these labels and their placement. [9]

The next result relates $R^*(n)$, the number of labels in the leaves in the pruned tree $T^*(n)$, to the number of labels in the nodes on the penultimate level of $T(n)$.

Lemma 3.2. *If $T(n)$ and $T(m)$ have the same number of penultimate level labels (say x labels), then $R^*(n) = R^*(m) = x - s + j$.*

[8] In fact, fewer initial conditions suffice. We can prove that for $s = 0$, kj initial conditions are required while for $s > 0$, we require $(s-1) + (k+1)j$ initial conditions. However, if we assume only the minimum number of initial conditions the combinatorial argument becomes unnecessarily more complicated by some special cases for the early parts of the tree.

[9] All of this is equivalent to saying that $R(n)$ is not affected if we take $T(n)$ and permute its labels. Formally speaking, we are showing correspondences with equivalence classes of trees rather than the trees themselves.

Proof. The leaves of $T^*(n)$ are exactly the penultimate level nodes of $T(n)$, and the number of labels in them is $x-s+j$: there are x labels in the penultimate nodes of $T(n)$, but when we change the first s-node to a j-node in the course of creating $T^*(n)$, s labels are deleted and j are added. Thus, $R^*(n) = x - s + j$. Applying the same argument to $T(m)$ we conclude that $R^*(n) = R^*(m) = x - s + j$. □

Lemma 3.3. *For all $n > 0$ the tree $T^*(n)$ is identical to the tree $T(n - s + j - R(n))$, up to renumbering of labels.*

Proof. Clearly $T^*(n) = T(y)$ for some $y \leqslant n$. To see that $y = n - s + j - R(n)$, note that in the pruning process to form $T^*(n)$ we delete $R(n)$ labels by removing the leaves of $T(n)$ (since by definition $R(n)$ is the number of entries in the entire bottom row), while by changing the first s-node to a j-node we remove s labels and add j labels. □

It follows immediately that $R^*(n) = R(n - s + j - R(n))$, which, like Lemma 3.2, provides an expression for evaluating $R^*(n)$. Substituting $n - qj$ for n in the previous equation, we obtain a key relation between R^* and R, which we now state as a lemma.

Lemma 3.4. *Suppose $0 < q \leqslant k$. Then $R^*(n - qj) = R(n - s - (q-1)j - R(n - qj))$.*

The term $R(n-s-(q-1)j-R(n-qj))$ in the above formula bears a clear resemblance to the general term on the right hand side of the recursion (2.1). The basic idea behind our proof that the sequence $R(n)$ satisfies (2.1) is to show how the number of labels in the leaves of $T(n)$ relates to the number of labels in the leaves of $T^*(n - qj)$ for $0 < q \leqslant k$.

As might be expected, the placement of the n^{th} label turns out to be important in this calculation. Our strategy is to demonstrate first that it is easy to count the number of labels in the leaves of *complete* trees $T(n)$, by which we mean trees where all the penultimate nodes present have their full complement of k children each, and each of these children (leaves) contains j labels. For trees that are not complete we apply this idea by *filling in* the missing labels so that we have a complete tree, and then adjusting the leaf label count on this completed tree.

In the next two lemmas we establish some easy facts about the position of labels on nodes in general and on penultimate nodes in particular.

Lemma 3.5. *Suppose that n is the d^{th} label on a j-node, where $1 \leqslant d \leqslant j$. Suppose that for some integer q all the labels between $n - qj$ and n are on j-nodes. Then $n - qj$ is the d^{th} label on its j-node.*

Proof. The label $n - d + 1$ is necessarily the first label on the same node as n. Thus the labels $n - d - j + 1$ through $n - j$ are the first d labels on the preceding (with respect to pre-order) j-node, that is, $n - j$ is the d^{th} label on this node. The result follows by repeating this argument. □

Lemma 3.6. *If the label n is on a penultimate level node, then that node is the only penultimate node containing any of the kj labels from $n - kj + 1$ through n. If n is on a leaf, then the parent of that leaf is the only penultimate node that can contain any of the labels $n - kj + 1$ through n. If n is neither on a leaf nor on a penultimate node, then none of the labels $n - kj + 1$ through n are on penultimate level nodes.*

Proof. Note that (by pre-order) every penultimate level node (except the first one) in T is followed immediately by k leaves, each with j labels. Thus, if n is on a penultimate level node, then there are at least kj labels between n and the preceding penultimate node (since the leaves of the preceding penultimate node contain kj labels). The same logic shows that if n is on a leaf, the only penultimate node that can contain any of the labels $n - kj$ through n is the parent of that leaf. Finally, if n is not on a leaf or a penultimate node then none of the preceding kj labels can be on any penultimate level node since there would have to be at least kj labels on the children of that penultimate node, and all of these labels would have to be prior to the label n in pre-order. This completes the proof. $\square$

We now formalize the notion of completeness that we discussed above.

Definition 3.7. For a given value of n, we say that $T(n)$ is *complete* if each of its penultimate level nodes has k children and each of these children has all j labels filled in.

For example, looking at Figure 3.1, we see that both $T_{2,3,2}(33)$ and $T_{2,3,2}(49)$ are complete. Note that if $T(n)$ is complete, it is not necessarily the case that $T^*(n)$ is complete, for example $T_{2,3,2}(49)$. The next result shows that as promised above, it is relatively easy to evaluate $R(n)$ in terms of $R^*(n)$ when $T(n)$ is complete.

Lemma 3.8. *If $T(n)$ is complete, then $R^*(n) = \frac{R(n)}{k}$.*

Proof. Since $T(n)$ is complete all its penultimate level nodes have k children and each child has j labels. But $T(n)$ has $R(n)$ labels in its leaves so there are $\frac{R(n)}{j}$ leaves in $T(n)$ (since the leaves of $T(n)$ are precisely the children of the penultimate nodes) and $\frac{R(n)}{jk}$ penultimate level nodes (counting the first s-node). Thus $T^*(n)$ will have $\frac{R(n)}{kj}$ leaves and $\frac{R(n)}{k}$ labels in these nodes (each leaf node has j labels, what was formerly the first s-node is now a j-node, and all of the former penultimate level nodes have k children so have their full complement of j labels). By definition, this means $\frac{R(n)}{k} = R^*(n)$. $\square$

As Lemma 3.8 illustrates, calculations involving the number of leaf labels become greatly simplified when the tree is complete. For this reason, the following lemma, which relates the leaf label count of a tree with the leaf label count of its *completion*, is central to much of the remainder of the discussion in this section. We define $\Delta_{s,j,k}(n)$, or $\Delta(n)$ where there is no confusion, as the least nonnegative number such that $T_{s,j,k}(n+\Delta_{s,j,k}(n))$ is complete.

Lemma 3.9. *If $\Delta(n) \leqslant kj$ then $R^*(n) = \frac{R(n+\Delta(n))}{k} = \frac{R(n)+\Delta(n)}{k}$ while if $\Delta(n) > kj$ then $R^*(n) = \frac{R(n+\Delta(n))}{k} - \Delta(n) + kj$.*

Proof. Since n is the final label on $T(n)$ and $T(n)$ is not complete, n must be either on a penultimate level node or on a leaf (but it cannot be the last entry on the k^{th} child of a penultimate node, since then $T(n)$ would be complete). If n is either the last entry on a penultimate node or is on a leaf, then $\Delta(n) \leqslant kj$ (since we have to fill in at most kj leaf entries to make $T(n)$ complete). In this case, $T(n)$ and $T(n+\Delta(n))$ are identical except on the leaf level, and so by Lemma 3.2, $R^*(n) = R^*(n+\Delta(n))$. By Lemma 3.8, since $T(n+\Delta(n))$ is complete, $R^*(n) = R^*(n+\Delta(n)) = \frac{R(n+\Delta(n))}{k} = \frac{R(n)+\Delta(n)}{k}$, where the last equality holds as $R(n+\Delta(n)) = R(n)+\Delta(n)$ in this case, since the labels from $n+1$ through $n+\Delta(n)$ are all leaf labels.

If n is on a penultimate level node but is not the last entry on that node then $\Delta(n) > kj$. We can use the same argument as above, but we also need to account for the fact that the penultimate node containing n is missing $\Delta(n) - kj$ labels. Simply subtract them off at the end of the above calculation to show that $R^*(n) = \frac{R(n+\Delta(n))}{k} - \Delta(n) + kj$. □

We now show how $R(n)$, the number of leaves in $T(n)$, relates to the number of leaves in pruned subtrees of $T(n)$. When combined with Lemma 3.4, this result provides the correspondence we seek between the labeled k-ary trees and the meta-Fibonacci sequences $A(n)$ defined by (2.1).

Theorem 3.10. *For all $n \geqslant 2s + k^2 j + kj - j$,*

$$R(n) = R^*(n-j) + R^*(n-2j) + \cdots + R^*(n-kj). \tag{3.1}$$

Proof. We consider various cases that depend on the location of the label n in the tree $T(n)$.

Case 1: The label n is on a leaf. In this case, note that by Lemma 3.6 the only penultimate level node located in the last kj labels is the parent of the leaf containing the label n. We need to know the location of the leaf that contains the label n in order to determine the location of the label $n-kj$, which is either on the parent of the leaf containing n, or on a node previous (with respect to pre-order) to the parent of the leaf containing n.

We consider two possibilities: either n is on the kth leaf of its parent (that is, the rightmost leaf), or it is not.

Subcase 1.1: The label n is on a rightmost leaf. Let n be the $(kj-c)^{th}$ label on the set of its sibling leaves, where $c < j$. Because all of the trees $T(n-j), \ldots, T(n-(k-1)j)$ have the same number of labels in penultimate nodes, we apply Lemma 3.2 to conclude that $R^*(n-j) = R^*(n-2j) = \cdots = R^*(n-(k-1)j)$. Since all of these trees are missing fewer than kj labels to be complete, the first part of Lemma 3.9 applies, so each of these quantities equals $R^*(n-j) = \frac{R(n-j)+\Delta(n-j)}{k} = \frac{R(n)-j+(j+c)}{k} = \frac{R(n)+c}{k}$, since clearly $\Delta(n-j) = j+c$. On the other hand, the tree $T(n-kj)$ necessarily requires an additional $\Delta(n-kj) = kj+c$ labels to be complete. By the second part of Lemma 3.9 $R^*(n-kj) = \frac{R(n-kj+\Delta(n-kj))}{k} - \Delta(n-kj) + kj = \frac{R(n+c)}{k} - c = \frac{R(n)+c}{k} - c$, the last equality holding because the final c labels are all leaf labels. Thus, $R^*(n-j) + R^*(n-2j) + \cdots + R^*(n-kj) = (k-1)\frac{R(n)+c}{k} + (\frac{R(n)+c}{k} - c) = R(n)$ as required.

Subcase 1.2: The label n is not on a rightmost leaf. Once again we examine the trees $T(n-j), \ldots, T(n-kj)$. In this case some of these trees are missing the parent of the leaf with the label n. Let n be the $(qj-c)^{th}$ label on the set of sibling leaves, where $0 \leqslant c < j$; note that $q < k$ since n is not on a rightmost leaf. From Lemma 3.6 we have that the labels $n-qj+c+1$ through n are on leaves, while the j labels $n-(q+1)j+c+1$ through $n-qj+c$ are on the parent node of n, which is on the penultimate level since n is on a leaf. As discussed above $n-(q+1)j+c+1$ through $n-qj+c$ are the only labels in the last kj that are on the penultimate level. Thus the exact location of the labels in the last kj prior to $n-(q+1)j+c+1$ is irrelevant; they are either on the first level (they are leaves), or on at least the third level.

By the above argument the trees $T(n), T(n-j), \ldots, T(n-(q-1)j)$ all have the same number of penultimate level labels. Thus, by Lemma 3.2, $R^*(n) = R^*(n-j) = R^*(n-2j) = \cdots = R^*(n-(q-1)j)$. Since n is the $(qj-c)^{th}$ label we apply Lemma 3.9 with $\Delta(n) = c+(k-q)j$ to $T(n)$ to yield $R^*(n) = \frac{R(n)+c+(k-q)j}{k}$. Thus, $R^*(n) = R^*(n-j) = R^*(n-2j) = \cdots = R^*(n-(q-1)j) = \frac{R(n)+c+(k-q)j}{k}$. Next, $T(n-qj)$ is missing $\Delta(n-qj) = kj+c$ labels to be complete, so by Lemma 3.9 $R^*(n-qj) = \frac{R(n)+c+(k-q)j}{k} - c$. Finally we calculate the number of leaf labels in $T(n-(q+1)j), \ldots, T(n-kj)$. By Lemma 3.6 none of the labels $n-kj+1$ through $n-qj$ are on the penultimate level. It follows that the number of penultimate level labels in each of these trees is precisely $j-c$ less than the number of penultimate level labels in $T(n-qj)$, since this latter tree contains $j-c$ labels in the penultimate node that is the parent of the leaf containing n. Thus, $R^*(n-(q+1)j) = \cdots = R^*(n-kj) = R^*(n-qj)-(j-c) = \frac{R(n)+c+(k-q)j}{k} - j$. So the sum $R^*(n-j)+\cdots+R^*(n-kj) = \frac{R(n)+c+(k-q)j}{k}(q-1) + \frac{R(n)+c+(k-q)j}{k} - c + (\frac{R(n)+c+(k-q)j}{k} - j)(k-q)$ $= \frac{R(n)+c+(k-q)j}{k}k - c - j(k-q) = R(n)+c+(k-q)j - c - (k-q)j = R(n)$, as required.

Case 2: The label n is not on a leaf. While n may or may not be on a node on the penultimate level, it follows by Lemma 3.6 that the labels $n-kj+1$ through $n-j$ cannot be on nodes on the penultimate level. Thus all of the trees $T(n-j), T(n-2j), \ldots, T(n-kj)$ have the same number of penultimate nodes, so by Lemma 3.2 $R^*(n-pj)$ are all equal for $1 \leqslant p \leqslant k$.

We complete the proof by showing that $R^*(n-pj) = \frac{R(n)}{k}$ for $1 \leqslant p \leqslant k$. There are two cases to consider: the label n is on a penultimate node or not.

Assume that n is not on a node on the penultimate level. Then $T(n)$ must be complete, since it cannot be the case that there are missing labels on a penultimate node (since that would imply n was on this penultimate node, which is not the case) and it cannot be that a penultimate node has leaf children with missing labels (since by preorder, after filling in the labels of a penultimate node, all the leaves are filled in before filling labels in nodes on another level). Thus, by Lemma 3.8 $R^*(n) = \frac{R(n)}{k}$. But since n is not on a node on the penultimate level (and n is not on a leaf), $T(n)$ has the same number of penultimate nodes as $T(n-j)$. Hence by Lemma 3.2 $R^*(n-pj) = \frac{R(n)}{k}$ for $0 \leqslant p \leqslant k$, which completes the argument in this case.

Assume that n is on a node on the penultimate level. Then $n-j$ must be on a rightmost leaf node or a node that is at least level 3 (note that here we use our assumption that n is past the third complete binary subtree). If $n-j$ is on level 3 or higher then $T(n-j)$ is complete and the labels $n-j+1, \ldots, n$ are on nodes that are not leaves.. Otherwise $n-j$ is on a rightmost leaf node and $T(n-j)$ is missing $\Delta(n-j) = c$ labels to be complete, $0 \leqslant c \leqslant j-1$. If $c = 0$ then $T(n-j)$ is complete and again the labels $n-j+1, \ldots, n$

are on nodes that are not leaves.. Otherwise, the labels $n-j+1, \dots n-j+c$ are on the rightmost leaf with $n-j$ and the labels $n-j+c+1, \dots, n$ are on a node that is not a leaf. So, in all cases $T(n-j)$ is missing c leaf labels and $j-c$ nonleaf labels compared to $T(n)$. Thus, $R(n-j)+c = R(n)$. By Lemma 3.9, $R^*(n-j) = \frac{R(n-j)+\Delta(n-j)}{k} = \frac{R(n-j)+c}{k} = \frac{R(n)}{k}$, as required. □

Combining equation (3.1) and Lemma 3.4 we have the following result: for all $s, j \in \mathbf{N}, k > 1$, and for all $n \geqslant 2s + k^2 j + kj - j$

$$R(n) = \sum_{q=1}^{k} R(n - s - (q-1)j - R(n - qj)). \tag{3.2}$$

That is, $R(n)$, the number of labels in the leaves of the binary tree $T(n)$, equals $A(n)$, where $A(n)$ is defined by (2.1) together with the following initial conditions: for all i in the interval $0 \leqslant i \leqslant 2s + k^2 j + kj - j$, $A(i) = R(i)$.

Finally, as in [18], the above result also holds for $s < 0$, where once again the initial conditions for the recursion must follow the tree. We briefly sketch how this works. In this case the labeling scheme for the tree, which generalizes the one explained in [18], is as follows: initially we label as for $s = 0$, then for each level of the binary tree except for the leaf level we delete the smallest $-s$ labels on that level (that is, the $-s$ labels immediately to the right of each s-node). Finally we renumber all the remaining labels in the tree according to preorder. See Figure 3.2 for the case $s = -3, j = 2, k = 2$.

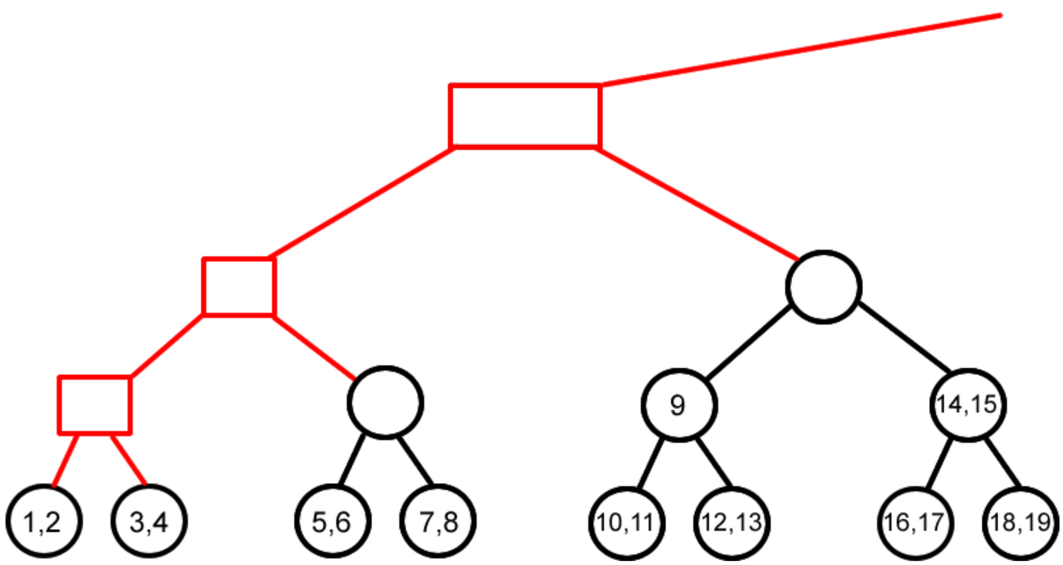

Figure 3.2: The initial part of the tree $T_{-3,2,2}$.

The pruning process for the tree extends in the natural way. As before delete all the leaves and change the first s-node to a j-node. Now however we must add $-s$ labels to fill in the empty j-nodes on the new bottom level (because there is no longer an s-node on this level). From this point the proof of the result follows essentially as above.

4 Combinatorial Interpretation for the Family [0,j : 2j,3j]

In this section we turn our attention to the meta-Fibonacci recursion (2.2), which generalizes the recursion $(s, 1 : 2 + s, 3)$ for which a combinatorial interpretation is described in [3]. It turns out that for $j > 1$ we can generalize this interpretation by adapting the labeling scheme and pruning process to the same infinite tree structure described in [3][10].

This infinite labeled tree, which we call the *BLT* tree, can be constructed as follows: Let $H_{s,j}$ denote a tree that consists of an infinite number of rooted trees joined together at their respective roots plus a single initial isolated node. Figure 4.1 shows the initial portion of $H_{s,j}$ where $s = 3$ and $j = 2$. The i^{th} rooted subtree of $H_{s,j}$ has 2^{i-1} chains of length 2 descending from the root. We say that the roots of these subtrees are at *level 3* in $H_{s,j}$; the children of these nodes (that is, the first node in the chains of length 2 from the roots) are said to be at *level 2*, while the children of the level 2 nodes (the grandchildren of the roots) are at *level 1*, the bottom of the tree. These bottom nodes are also called the *leaves* of $H_{s,j}$. This ordering of the levels of $H_{s,j}$, which differs from that described in [3], is selected because it is analogous to the one in the previous section. Note that each successive root has twice as many level 1 and level 2 descendants as the previous root. The isolated node is also considered to be a leaf on level 1.

We label the nodes of the tree $H_{s,j}$ with the positive integers as follows: all of the nodes except the nodes on level 3 receive j labels (so are called j-nodes). The nodes on level 3 receive s labels (so are called s-nodes). Note that if $s = 0$ the level 3 nodes have no labels. The tree is labeled in pre-order: we begin with the isolated node, which receives the j labels 1 through j. The first s-node is labeled next with the s labels $j + 1, .., j + s$. The unique level 2 child of the first s-node is labeled next, followed by its level 1 grandchild, each with the first j consecutive integers not yet used. This process is repeated so that each s-node is labeled, followed by its first child, then first grandchild, second child, second grandchild and so on from left to right. For example, the second s-node has two level 2 children and two level 1 grandchildren, so the labeling takes place as s-node, level 2 node, level 1 node, level 2 node, level 1 node. See Figure 4.1, where the labeling for $H_{3,2}$ is shown up to the label 39.

Let $H_{s,j}(n)$ denote that portion of the tree $H_{s,j}$ up to and including the n^{th} label. Note that the last node may not contain its full complement of either s or j labels. See Figure 4.2 for $H_{3,2}(28)$. Let $G_{s,j}(n)$ be the number of labels that occur in the leaves of $H_{s,j}(n)$. For example, $G_{3,2}(28) = 10$, since there are 10 leaf labels in $H_{3,2}(28)$. Where there is no confusion we often drop the subscripts on H, $H(n)$ and $G(n)$. In what follows we show that $G(n)$ satisfies the meta-Fibonacci recursion (2.2).

For each s-node beyond the first s-node, we call the leftmost level 2 child (respectively, level 1 grandchild) of this s-node and every second child subsequent to it an *odd* level 2 (respectively, level 1) node. The remaining level 2 (respectively, level 1) nodes of this

[10] Although the results we will derive in this section can also be proved inductively, the proofs would be excessively difficult. Therefore we proceed with an argument similar to that of Section 3.

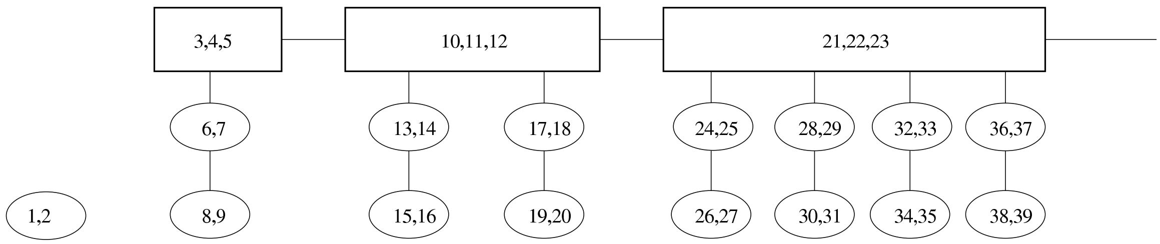

Figure 4.1: The BLT tree $H_{3,2}$

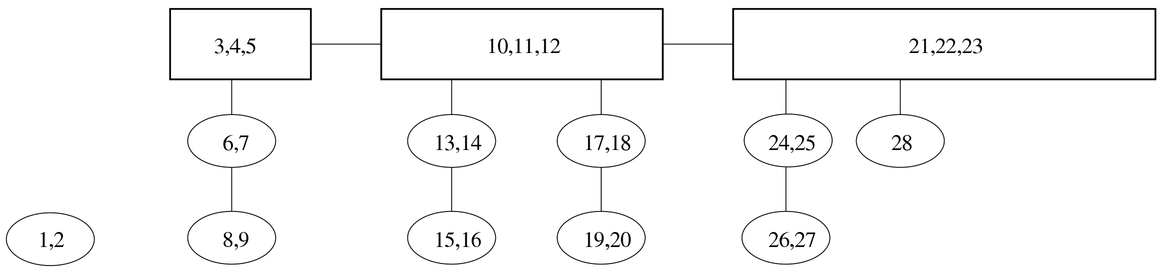

Figure 4.2: The BLT tree $H_{3,2}(28)$

s-node are called *even* nodes. The descendants of the first s-node are called even. For example, in Figure 4.1, the node containing the labels 15 and 16 is odd, and the node containing the labels 28 and 29 is even. In this way, since the isolated node (which we consider odd) occurs before the first s-node is a level 1 node, we are able to maintain the pattern that there is one even level 2 node for every two level 1 nodes. This is analogous to the previous section, where there were two level 1 nodes for every level 2 node. This proves to be important for the pruning process that we define below for this tree.

As in Section 3, we establish several important structural properties of the trees H and $H(n)$. Here too, we assume that the final label in $H(n)$ is in a node sufficiently far along in the tree; in this case we assume $n > 2s + 4j$. With this assumption, we demonstrate combinatorially that for such values of n, $G(n) = B(n)$. With this, we conclude that $G(n) = B(n)$ for all n, so long as we take as initial conditions for $B(n)$ that $B(n) = G(n)$ for $n \leqslant 2s+4j$; we say that these initial conditions *follow the tree*[11]. We call the sequence $G(n)$ a label counting sequence.

We now define a pruning process on both H and $H(n)$ respectively that creates a new BLT tree of each type by removing certain nodes, rearranging others and labeling the resulting tree. The pruning process on the infinite tree H is as follows: begin by removing the first s-node, together with all its labels, and all the leaves (and their respective labels) of H (including the isolated node). In the new tree we take the lone level 2 node of the first s-node (that was removed) and slide it down to make it the first isolated node at

[11] As in the previous section, fewer initial conditions suffice. We can prove that for $s = 0$, $3j$ initial conditions are required while for $s > 0$, we require $s - 1 + 4j$ initial conditions. However, if we assume only the minimum number of initial conditions the combinatorial argument becomes unnecessarily more complicated by some special cases for the early parts of the tree.

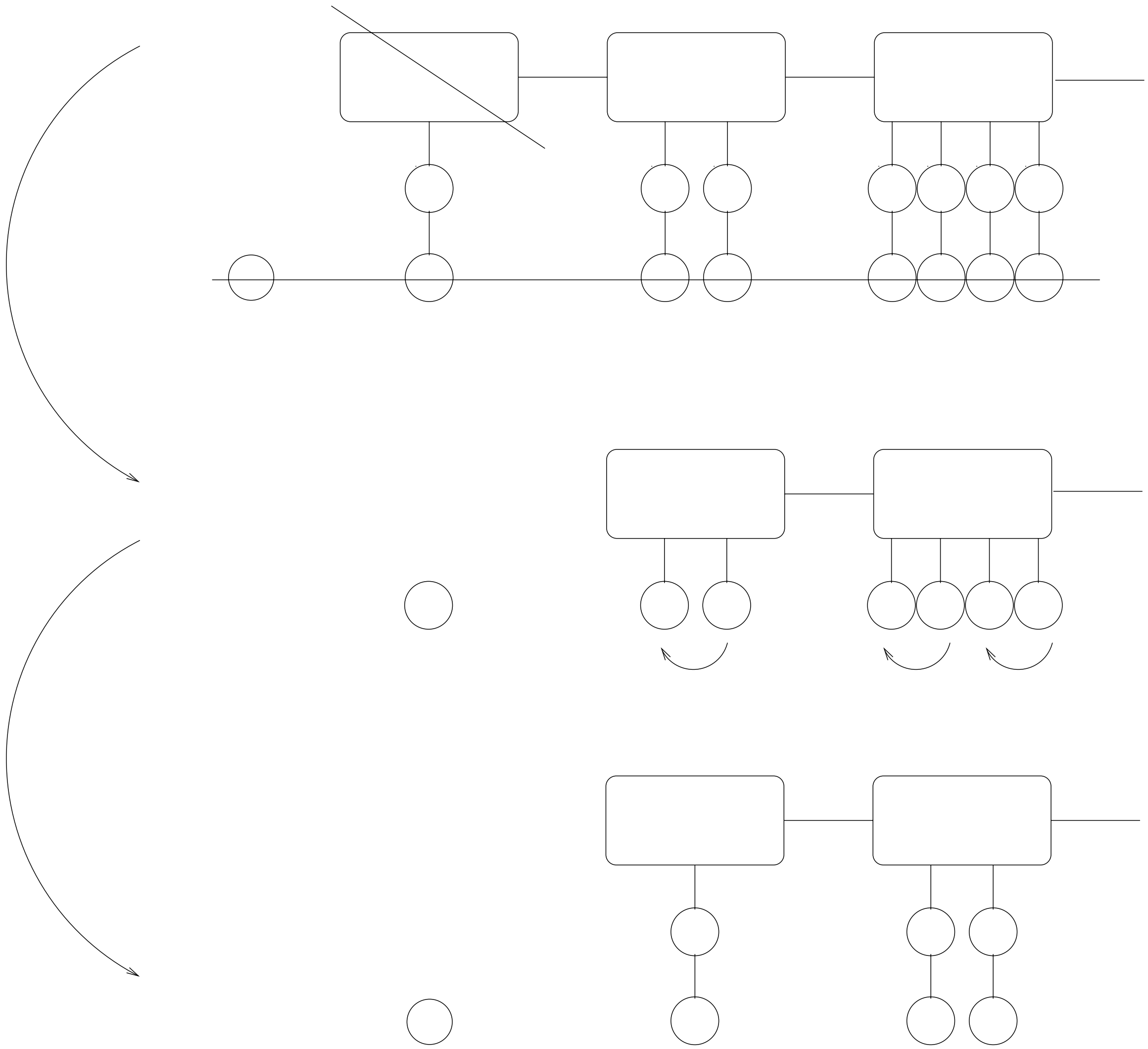

Figure 4.3: The pruning process on the infinite BLT tree H

level 1. Take every even node formerly at level 2 (the even children of the s-nodes) and shift them down to level 1 as the child of its former level 2 sibling on its immediate left. See Figure 4.3. Call this new infinite tree H^*. It is evident from Figure 4.3 that H^* is the same infinite tree as H once the labels of H^* are renumbered. That is, we have the following proposition:

Proposition 4.1. *The infinite trees H^* and H are identical up to renumbering of the labels.*

The pruning process that we adopt for the finite tree $H_{s,j}(n)$ is very similar to that for H, but with one additional step at the end: in the nodes of the new pruned tree, which we call $H^*_{s,j}(n)$, we place j new labels in the first j available positions with respect to preorder. Adding j labels after deleting the first s-node mirrors the step in the pruning process in Section 3, where we change the first s-node to a j-node. For convenience we

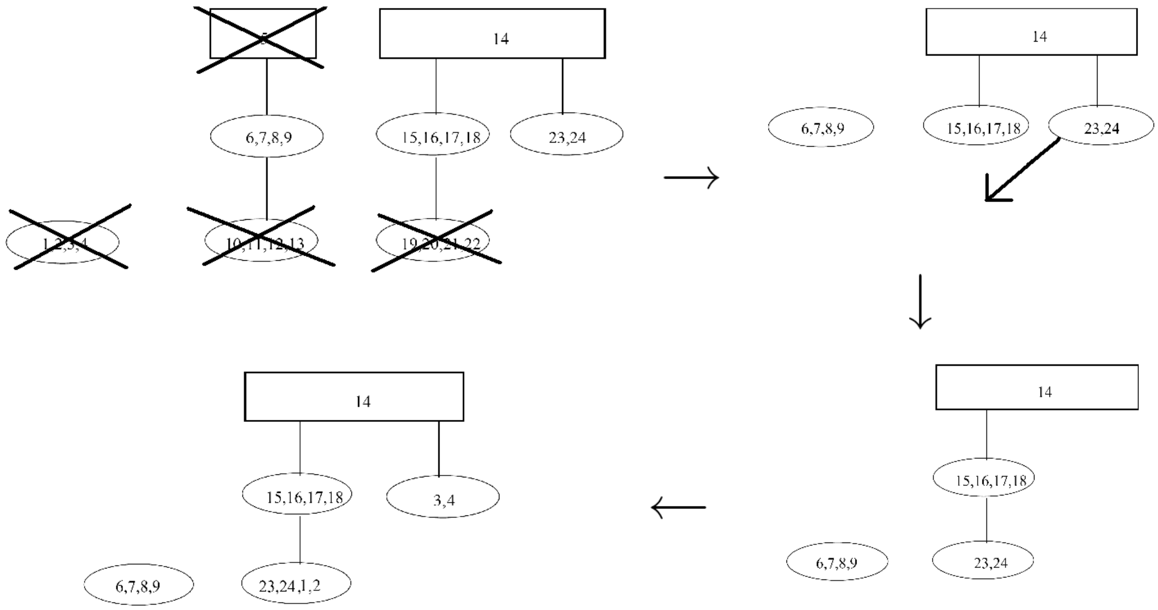

Figure 4.4: The pruning process on the BLT tree $H_{1,4}(24)$

can use the j labels $1, \ldots, j$, all of which are no longer in $H^*_{s,j}(n)$ (since they necessarily were in the original isolated node of $H_{s,j}(n)$ that was eliminated from the tree in the pruning process). See Figure 4.4 for an example of the pruning process on $H_{1,4}(24)$ and the resulting tree $H^*_{1,4}(24)$. Note that when adding these additional j labels it may be necessary to create a new node (either a j-node or s-node) in which to place some of these j additional labels; this new node may be only partially filled in. Denote by $G^*_{s,j}(n)$ the number of labels on the leaves of $H^*_{s,j}(n)$; where convenient we drop the subscripts, writing $H^*(n)$ and $G^*(n)$.

It is evident that $H^*(n)$ is $H(x)$ for $x = n - s - G(n) + j$: this is because in the pruning process on $H_{s,j}(n)$ that results in $H^*(n)$ we are removing $s + G(n)$ labels (the s labels come from the s-node that is removed, and the $G(n)$ labels correspond to the labels in all the leaves that are removed) and then adding back j labels. Therefore, we have the following fact:

Proposition 4.2. *The tree $H^*(n)$ is identical to $H(x)$ up to renumbering of the labels, where $x = n - s - G(n) + j$.*

To make this identification precise we should really complete the construction of the tree $H^*(n)$ by re-labeling $H^*(n)$ according to the same rules that we applied to label the original tree $H_{s,j}(n)$ from which it was derived. If we were to do so then it is evident that the two trees would be the same. As in Section 3, in what follows we omit this final gloss in the interests of simplicity, since generally we do not need to refer to the individual labels of $H^*(n)$ in our arguments; further, by numbering the j labels that we add as $1, \ldots, j$ it is easier for the reader to keep track of them in the figures.

The pruning process that we have defined on $H_{s,j}(n)$ leads directly to some key results for showing that $G(n)$ satisfies the meta-Fibonacci recursion (2.2).

Lemma 4.3. *$G^*(n-j) = G(n-s-G(n-j))$ and $G^*(n-3j) = G(n-s-2j-G(n-3j))$*

Proof. By the definition of $G(n)$ and $G^*(n)$ it suffices to show that $H^*(n-j) = H(n-s-G(n-j))$ and $H^*(n-3j) = H(n-s-2j-G(n-3j))$. Both of these follow immediately from Proposition 4.2 with $n-j$ and $n-3j$, respectively, in place of n. □

From Lemma 4.3 it is evident that if $G(n) = G^*(n-j) + G^*(n-3j)$ then $G(n)$ satisfies the meta-Fibonacci recursion (2.2). Thus, our interest lies in counting $G^*(n-j)$ and $G^*(n-3j)$, the number of leaves in the pruned trees $H^*(n-j)$ and $H^*(n-3j)$, respectively. In certain cases this is a relatively easy task. For example, suppose that $H(n)$ has the property that its last label n occurs as the j^{th} label on an even leaf (an even level 1 node). Adopting similar terminology to the preceding section, we call such a BLT tree $H(n)$ *complete*. Then we have the following result:

Proposition 4.4. *If $H(n)$ is complete, then (1) $G^*(n-j) = \frac{G(n)}{2}$; (2) $G^*(n-3j) = \frac{G(n)}{2}$; and (3) $G(n) = G(n-s-G(n-j)) + G(n-s-2j-G(n-3j))$.*

Proof. Note that (3) is is an immediate consequence of (1) and (2) together with Lemma 4.3. We focus on the proof of (1) and (2) in turn.

(1) The sequence $G^*(n-j)$ counts the number of leaves in the pruned tree $H^*(n-j)$, so we apply the pruning process to $H(n-j)$. Since $H(n)$ is complete, the label n is the j^{th} label in an even level 1 node. Thus $n-j$ is the j^{th} label in the level 2 parent of the node containing n. When we apply the pruning process to $H(n-j)$ this last level 2 node becomes the final leaf in $H^*(n-j)$, and $n-j$ is the jth and final label on this leaf. To complete the pruning process we add j new labels to the end of the pruned tree; since a level 1 node is followed by a node on level 2 or by a node on level 3 and then another node on level 2, it is necessarily the case that none of these additional j labels appear in a leaf.

We now count the number of leaf labels in $H^*(n-j)$. The leaf nodes in $H^*(n-j)$ are precisely the nodes that previously were even level 2 nodes in $H(n-j)$. But the level 2 nodes (and their labels) are the same in both $H(n-j)$ and $H(n)$, since the only difference between these trees is the last leaf node in $H(n)$ that is not in $H(n-j)$. Since $H(n)$ is complete, the number of even level 2 nodes in $H(n)$ is exactly half the total number of level 2 nodes in the tree, which in turn is the same as the number of leaf nodes in $H(n)$. Finally, again since $H(n)$ is complete, all the level 2 nodes contain the full complement of j labels. Combining these observations we deduce that $G^*(n-j)$, the number of labels on the leaves of $H^*(n-j)$ equals the number of labels on the even level 2 nodes of $H(n-j)$, which is the same as the number of labels on the even level 2 nodes of $H(n)$, which is $\frac{G(n)}{2}$. That is, $G^*(n-j) = \frac{G(n)}{2}$.

(2) The argument is similar to the above. Since $H(n)$ is complete, it has an even number of leaves, the last level 2 node is even and $n-3j$ is the j^{th} (and last) label in an odd level 2 node that has its full complement of labels. Thus, $H(n-3j)$ has two fewer leaves than $H(n)$ so $G(n-3j) = G(n)-2j$. Further, when we apply the pruning process

to $H(n-3j)$ this final odd level 2 node remains in place, and the additional j labels that are added in forming $H^*(n-3j)$ must necessarily be placed in a new level 1 node (leaf) that is the child of this final level 2 node.

The leaf labels on $H^*(n-3j)$ consist of the labels that were on the even level 2 nodes of $H(n-3j)$, as well as the additional j labels added to the final leaf in $H^*(n-3j)$ as just described. But we have already seen that the last node of $H(n-3j)$ is an odd level 2 node so $H(n-3j)$ has an even number of leaves, in fact, precisely two fewer than $H(n)$. Thus, $H(n-3j)$ has half as many even level 2 nodes as leaves. Since all the leaves have j labels we get that the number of labels in the even level 2 nodes of $H(n-3j)$ is $\frac{G(n-3j)}{2} = \frac{G(n)}{2} - j$. Thus we have that $G^*(n-3j) = (\frac{G(n)}{2} - j) + j = \frac{G(n)}{2}$. $\square$

To establish more generally that $G(n) = G(n-s-G(n-j)) + G(n-s-2j-G(n-3j))$ we analyze how the tree $H(n)$ can deviate from being complete and amend the calculation of $G^*(n-j)$ and $G^*(n-3j)$ accordingly. The details of this approach are provided in the following series of lemmas, culminating in the proof of Theorem 4.10.

At the outset we fix n, s and j and define two new parameters. Let $\Delta_{s,j}(n)$, or $\Delta(n)$ where it is clear, be the smallest nonnegative integer such that the tree $H_{s,j}(n+\Delta_{s,j}(n))$ is complete. If $\Delta(n) = 0$ then $H_{s,j}(n)$ is complete. Otherwise, the magnitude of $\Delta(n)$ provides some indication of how far the tree $H_{s,j}(n)$ is from being complete. Note that $\Delta(n) < 4j + s$ since at worst the label n is the first label of an s-node; in this case, to complete the tree we must add $s-1$ labels to that node and j labels to each of four additional j-nodes following this s-node.

We denote by $\delta_{s,j}(n)$, or $\delta(n)$ where it is clear, the smallest non-negative integer such that all of the j-nodes of $H_{s,j}(n+\delta_{s,j}(n))$ contain j labels. If $\delta(n) = 0$ then the last j-node contains j labels. Otherwise the last node of $H_{s,j}(n)$ (in preorder) is a j-node that has $j - \delta(n)$ labels.

In the next lemma we derive some formulas for $G(n)$ and $G(n-2j)$ involving $\Delta(n)$ and $\delta(n)$ that we use to evaluate $G^*(n-j)$ and $G^*(n-3j)$ when $H(n)$ is not complete.

Lemma 4.5. *(1) If $0 \leqslant \Delta(n) < j$ (that is, the label n is on an even level 1 node) then $G(n+\Delta(n)) = G(n) + \Delta(n)$ and $G(n-2j) + j = G(n)$.*
(2) If $j \leqslant \Delta(n) < 2j$ (that is, the label n is on an even level 2 node) then $G(n+\Delta(n)) = G(n) + j$ and $G(n-2j) + j = G(n)$.
(3) If $2j \leqslant \Delta(n) < 3j$ (that is, the label n is on an odd level 1 node) then $G(n+\Delta(n)) = G(n) + j + \delta(n)$. In this case the location of the label $n-2j$ varies by s, j, and whether or not n is on the first odd level 1 node of a s-node.
(a) If $n-2j$ is on a level 3 node, then $G(n-2j) + j - \delta(n) = G(n)$.
(b) If $n-2j$ is on a level 1 node, then $G(n-2j) + j = G(n)$ if the node containing the label $n-2j$ and the node containing the label n are both descendants of the same s-node; otherwise, $G(n-2j) + j - s = G(n)$.
(4) If $3j \leqslant \Delta(n)$ (that is, n is on an odd level 2 node or a s-node (level 3 node)) then $G(n+\Delta(n)) = G(n) + 2j$. Again, we need to split into subcases based on where $n-2j$ is located, which varies by the relative value of s, j, and $\delta(n)$ and by whether n is on the first odd level 2 node of a s-node.

(a) If $n-2j$ is on a level 3 node, then $G(n-2j) = G(n)$.
(b) If $n-2j$ is the $(j-\beta)^{th}$ label on a level 1 node (where $\beta \geqslant 0$), then $G(n-2j)+\beta = G(n)$.
(c) If $n-2j$ is on a level 2 node, then $G(n-2j)+j = G(n)$.

Proof. A tree is complete when its last label occurs as the j^{th} label on an even level 1 node. In each case above, to derive the formula for $G(n)$ the basic idea is to figure out how many of the $\Delta(n)$ labels that must be added to complete the tree $H(n)$ are leaf labels. To derive the formula for $G(n-2j)$ we count those leaf labels that are present in $H(n)$ but are not in $H(n-2j)$.

(1) In this case all of the missing $\Delta(n)$ labels are leaf labels, so $G(n+\Delta(n)) = G(n)+\Delta(n)$. Notice that here $\Delta(n) = \delta(n)$.

Since the label n in $H(n)$ is on an even level 1 node, the label $n-2j$ in the tree $H(n-2j)$ is the last label on an odd level 1 node. Clearly this node contains $j-\Delta(n)$ labels, so the tree $H(n-2j)$ requires $j+\Delta(n) = \Delta(n-2j)$ labels to be complete. So the leaf labels that are on $H(n)$ but not $H(n-2j)$ are the $\Delta(n)$ leaf labels missing from the leaf with label $n-2j$ and the $j-\Delta(n)$ leaf labels in $H(n)$ that are on the leaf with n. Thus we get a total of j leaf labels that are on $H(n)$ but not $H(n-2j)$. So $G(n-2j)+j = G(n)$.

(2) Since n is on an even level 2 node in $H(n)$, the leaf labels that are added to $H(n)$ to obtain $H(n+\Delta(n))$ are exactly the j labels of the level 1 child of the node with n. Hence $G(n+\Delta(n)) = G(n)+j$.

The label $n-2j$ will always be on the odd level 2 node directly to the left of the node with label n, so the leaf labels that are present in $H(n)$ but not in $H(n-2j)$ are the j labels on the level 1 child of the node with label $n-2j$. Hence, $G(n-2j)+j = G(n)$.

(3) To complete $H(n)$ we must fill the node containing the label n, then add a new level 2 node with j labels, together with its level 1 child and j labels, immediately to the right of the node in $H(n)$ containing the label n. This adds $\delta(n)$ leaf entries to fill the level 1 node with n, and j leaf entries to fill the newly created level 1 node to its immediate right. Thus $G(n+\Delta(n)) = G(n)+j+\delta(n)$.

We now compute the value of $G(n-2j)$ in subcases (a) and (b) in the statement of the lemma.

(a) The level 3 node containing the label $n-2j$ has two descendants in $H(n)$ - a level 2 node with j labels and a level 1 node with $j-\delta(n)$ labels containing the final label n. The only leaf labels on $H(n)$ but not on $H(n-2j)$ are the $j-\delta(n)$ leaf labels on the node with n, and so $G(n-2j)+j-\delta(n) = G(n)$.

(b) If both the labels n and $n-2j$ are on level 1 nodes that are descendants of the same s-node then the situation is essentially identical to (1) for the present purposes and $G(n-2j)+j = G(n)$. If this is not the case, then there is an s-node between the two level 1 nodes containing the labels n and $n-2j$. But there are only a total of $2j$ labels missing between $H(n-2j)$ and $H(n)$, and we know that $j+s$ of these are not leaf labels. Hence, the number of missing leaf labels is $2j-(j+s) = j-s$ so $G(n-2j)+j-s = G(n)$.

(4) To complete $H(n)$ we need to add and completely label two level 1 j-nodes. This means that $G(n+\Delta(n)) = G(n)+2j$. Note that in the process of completing $H(n)$, depending on where the label n is located, we may also need to add and/or label one or

perhaps two level 2 nodes and possibly add labels to the level 3 node if it contains the label n.

(a) This case only occurs when s is large enough so that the label n is either on a s-node or n is the only level 2 child of an s-node and $n-2j$ is on this same s-node. This means that there are no leaf labels between n and $n-2j$ and hence $G(n-2j)=G(n)$, or else we would be in case (c).

(b) Note that it must be the case that there is an s-node between $n-2j$ and n (if n is on an s-node, then that s-node is the one between the nodes containing the labels $n-2j$ and n). Since n is on a level 2 or s-node the only leaf labels that are on $H(n)$ but not $H(n-2j)$ are the β leaf labels on the leaf with the label $n-2j$. Thus $G(n-2j)+\beta=G(n)$.

(c) In this case there may or may not be an s-node between $n-2j$ and n. Regardless, the only leaf labels between $n-2j$ and n are on the node that is the level 1 child of the level 2 node with the label $n-2j$. Because n is not on that child node, all j of the labels on that level 1 node are on $H(n)$ but not on $H(n-2j)$. Therefore, $G(n-2j)+j=G(n)$. $\square$

When $H(n)$ is not complete, the computation of $G^*(n-j)$, the number of leaves in the tree that results from pruning the tree $H(n-j)$, is influenced by the nature of the additional $\Delta(n)$ labels required to complete $H(n)$. In the following we consider separately four cases, each of which is defined by the location of the label n.

Recall that pruning the tree $H(n-j)$ involves three separate steps: (1) eliminate all the leaves and the first s-node, together with the labels in these nodes; (2) shift down each even level 2 node to level 1 so that it is the unique child of the odd level 2 node that is to its immediate left; and (3) add j additional labels 1,2,. . .,j to the tree (which may involve adding a new s-node or j-node that is partially filled). In the course of the proofs of the following four lemmas, we find ourselves simultaneously pruning a pair of trees, such as $H(n-j)$ and $H(n+\Delta(n)-j)$, and comparing the pruned trees $H^*(n-j)$ and $H^*(n+\Delta(n)-j)$ that result. It is useful to compare what the trees look like after the first two steps in the pruning process, that is, before the additional j labels are added to the trees. For convenience we refer to the tree that results from just these first two steps on the tree $H(x)$ as the intermediate tree of $H(x)$ in the pruning process.

Lemma 4.6. *If n is a label on an even level 1 node in $H(n)$, then $G^*(n-j)=\frac{G(n)+\delta(n)}{2}$.*

Proof. Here $0 \leqslant \Delta(n) < j$ and $\delta(n)=\Delta(n)$. By the definition of $\Delta(n)$, the tree $H(n+\Delta(n))=H(n+\delta(n))$ is complete. By Proposition 4.4(1), $G^*(n+\delta(n)-j)=\frac{G(n+\delta(n))}{2}$. By Lemma 4.5(1), $G(n+\delta(n))=G(n)+\delta(n)$. To complete the proof we show that $G^*(n+\delta(n)-j)=G^*(n-j)$

Consider the two trees $H(n-j)$ and $H(n+\delta(n)-j)$. The only difference between these two trees is that $H(n+\delta(n)-j)$ has a full complement of j labels in its last node, whereas $H(n-j)$ has $j-\delta(n)$ labels in its last node. We will simultaneously transform these trees to $H^*(n-j)$ and $H^*(n+\delta(n)-j)$ respectively.

Following the pruning procedure outlined earlier in this section we eliminate the leaves of both trees and the first s-nodes. Then in each tree we shift each even level 2 node to be level 1 child of its neighbor on its immediate left. Since the last node in each of the trees $H(n-j)$ and $H(n+\delta(n)-j)$ is an even level 2 node, this node gets shifted down to a level 1 node in $H^*(n-j)$ and $H^*(n+\delta(n)-j)$. Finally, we add j labels to each of the trees $H^*(n-j)$ and $H^*(n+\delta(n)-j)$. For the tree $H^*(n+\delta(n)-j)$, since the final node at present is a leaf node with all j labels, it follows that all of these additional j labels will be placed into either a new level 2 node or in a new s-node and possibly also into a new level 2 node. For the tree $H^*(n-j)$ that derives from $H(n)$, the first $\delta(n)$ labels are placed into the last node, which again is a leaf node. The remaining $j-\delta(n)$ labels are placed into either a new level 2 node or possibly into a new s-node as well as a new level 2 node, depending on the relative sizes of $\delta(n)$, s and j. In any event, both the trees $H^*(n-j)$ and $H^*(n+\delta(n)-j)$ have a leaf label count that is equal; that is, $G^*(n+\delta(n)-j) = G^*(n-j)$, as desired. □

Lemma 4.7. *If n is a label on an even level 2 node in $H(n)$, then $G^*(n-j) = \frac{G(n)+j}{2}$.*

Proof. Since the label n is on an even level 2 node in $H(n)$, we have $j \leqslant \Delta(n) < 2j$. By Proposition 4.4(1), $G^*(n+\Delta(n)-j) = \frac{G(n+\Delta(n))}{2}$. By Lemma 4.5(2), $G(n+\Delta(n)) = G(n)+j$. To complete the proof we must show that $G^*(n+\Delta(n)-j) = G^*(n-j)$

As in the previous lemma we simultaneously transform $H(n-j)$ and $H(n+\Delta(n)-j)$ to $H^*(n-j)$ and $H^*(n+\Delta(n)-j)$ respectively and argue that the leaf label count of these transformed trees are equal. Note that since the label n is on an even level 2 node, the label $n+\Delta(n)$ on the complete tree $H(n+\Delta(n))$ is the last label on an even leaf, the label $n+\Delta(n)-j$ is the last label on an even level 2 node and the label $n-j$ is on an odd leaf. See Figure 4.5 for an illustration of the simultaneous pruning process.

Initially, $H(n-j)$ and $H(n+\Delta(n)-j)$ have only full level 2 nodes, and $H(n+\Delta(n)-j)$ has 1 more level 2 node. In the pruning process on the trees $H(n-j)$ and $H(n+\Delta(n)-j)$ first we eliminate all the leaves and the initial s-node, then shift even level 2 nodes down. At this point the tree that began as $H(n+\Delta(n)-j)$ has one more leaf node than the tree that started as $H(n-j)$ so the leaf label count in this tree is higher by j. To complete the pruning process on each tree we add j labels to each tree in the first j available positions in preorder. We will show that these j additional labels are leaves for $H^*(n-j)$ and not leaves for $H^*(n+\Delta(n)-j)$, which cancels out the difference of j leaf labels.

Consider first the situation for the tree that started as $H(n-j)$: since the label $n-j$ was on an odd leaf that was eliminated in the pruning process, the last node in the intermediate tree is an odd level 2 node that has its full complement of j labels. So the first available position with respect to pre-order is a level 1 node. Thus, the addition of j labels to the intermediate tree of $H(n-j)$ to complete the pruning process increases the leaf label count by j.

The label $n+\Delta(n)-j$ is the last label on an even level 2 node in $H(n+\Delta(n)-j)$. In the pruning process of $H(n+\Delta(n)-j)$ this even level 2 node shifts down to a leaf, so the next available position in the intermediate tree at this stage is a level 2 node, unless the next node with respect to pre-order is an s-node. In this case, some or all of the labels will be placed into the s-node before the next level 2 node. In either case the additional j labels that complete the process of creating the pruned tree $H^*(n+\Delta(n)-j)$ do not affect its leaf count.

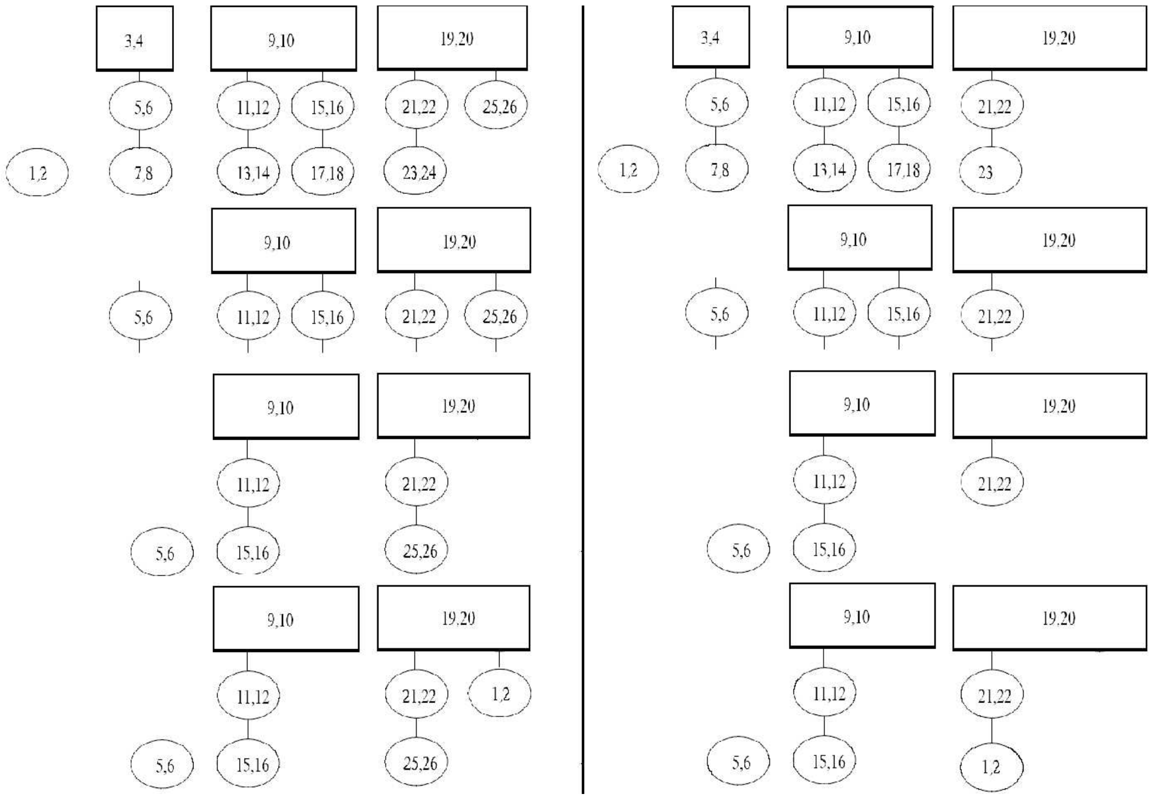

Figure 4.5: Simultaneous pruning process as in Lemma 4.7 with $n = 25$, so $\Delta(n) = 3, \delta(n) = 1$. On the right is $H_{2,2}(n-j) = H_{2,2}(23)$, and on the left is $H_{2,2}(n-j+\Delta(n)) = H_{2,2}(26)$.

The intermediate tree for $H(n+\Delta(n)-j)$ has j more labels than the intermediate tree for $H(n-j)$. But the j labels added at the end of the pruning process add j to the leaf label count for $H^*(n-j)$ but nothing to the leaf label count for $H^*(n+\Delta(n)-j)$. It follows that the leaf counts in $H^*(n-j)$ and $H^*(n+\Delta(n)-j)$ are equal, that is, $G^*(n+\Delta(n)-j) = G^*(n-j)$. Thus we have $G^*(n-j) = G^*(n+\Delta(n)-j) = \frac{G(n+\Delta(n))}{2} = \frac{G(n)+j}{2}$.

□

Lemma 4.8. *If the label n is on an odd level 1 node in $H(n)$, then $G^*(n-j) = \frac{G(n)+j-\delta(n)}{2}$.*

Proof. Since the label n is on an odd level 1 node, $2j \leqslant \Delta(n) < 3j$. By Proposition 4.4(1), $G^*(n+\Delta(n)-j) = \frac{G(n+\Delta(n)}{2}$. By Lemma 4.5(3), $G(n+\Delta(n)) = G(n)+j+\delta(n)$. To complete the proof we must show that $G^*(n+\Delta(n)-j) = G^*(n-j)+\delta(n)$

It is evident that the label $n-j$ is on the odd level 2 node that is the parent of the odd level 1 node containing n. Since the tree $H(n+\Delta(n))$ is complete, it contains two additional nodes (one even level 2 and level 1 node, respectively) compared to $H(n)$; each of these nodes has its full complement of j labels. The tree $H(n+\Delta(n)-j)$ has just the additional even level 2 node with its j labels. Further, in $H(n+\Delta(n)-j)$ the odd level

1 node containing the label $n-j$ contains $\delta(n)$ more labels than the odd level 1 node containing the label $n-j$ in $H(n-j)$.

We simultaneously prune the trees $H(n-j)$ and $H(n+\Delta(n)-j)$ to $H^*(n-j)$ and $H^*(n+\Delta(n)-j)$ respectively. For each tree first we begin by eliminating the leaves and the first s-node, together with their respective labels. Then we shift the even level 2 nodes to be level 1 nodes.

Note that in the intermediate tree formed in the pruning process for $H(n-j)$ the odd level 2 node containing the label $n-j$, which is its last node, does not get eliminated and does not get shifted down. We now complete the pruning process on $H(n-j)$ by adding j labels to this intermediate tree, placing $\delta(n)$ labels in its last odd level 2 node containing the label $n-j$ and $j-\delta(n)$ in the odd level 1 child of this node.

In the intermediate tree formed in the pruning process for $H(n+\Delta(n)-j)$ the odd level 1 node that is the child of the odd level 2 node containing the label $n-j$ is already full. Thus the j labels added to complete the pruning process for this tree are placed in an even level 2 node, so the leaf label count is not affected. It follows that the only difference between $H^*(n+\Delta(n)-j)$ and $H^*(n-j)$ on the leaf level is that the final level 1 node in $H^*(n-j)$ is missing $\delta(n)$ labels. Hence $G^*(n+\Delta(n)-j) = G^*(n-j)+\delta(n)$ as required. □

Lemma 4.9. *If the label n is on an odd level 2 node or a level 3 node in $H(n)$, then $G^*(n-j) = \frac{G(n)}{2}$.*

Proof. Because of the position of the label n in $H(n)$, we know that $3j \leqslant \Delta(n) \leqslant 4j+s-1$. By Prop. 4.4(1), $G^*(n+\Delta(n)-j) = \frac{G(n+\Delta(n))}{2}$. By Lemma 4.5(4), $G(n+\Delta(n)) = G(n)+2j$. To complete the proof we must now show that $G^*(n+\Delta(n)-j) = G^*(n-j)+j$

Since the label n is on an odd level 2 node or a level 3 node, the label $n-j$ is on an even level 1 node or an s-node. It follows that the tree $H(n+\Delta(n)-j)$ has two more level 2 nodes (one even, one odd) than the tree $H(n-j)$.

We simultaneously prune $H(n-j)$ and $H(n+\Delta(n)-j)$ to $H^*(n-j)$ and $H^*(n+\Delta(n)-j)$ respectively. For each tree first we eliminate the leaves and the initial s-node and shift the even level 2 nodes to become level 1 nodes. Since the label $n-j$ is on an even level 1 node or an s-node, it follows that the intermediate tree for $H(n-j)$ must have a final level 1 node with a full complement of j labels. Since it only has full level 2 nodes, the intermediate tree formed from $H(n+\Delta(n)-j)$ has one additional leaf node with j labels compared to the intermediate tree from $H(n-j)$, coming from its one extra even level 2 node.

We now complete the pruning process on $H(n-j)$ by adding the j labels to its intermediate tree. From what we have just noted above, these labels are added to a level 2 node, unless the next available node with respect to preorder is an s-node, in which case some or all of the labels are placed in this s-node first. In any case, the labels are not placed in a level 1 node and so do not affect the leaf count.

Similarly, we complete the pruning process on $H(n+\Delta(n)-j)$ to produce $H^*(n+\Delta(n)-j)$. Here too since the last node in the intermediate tree is a level 1 node with j labels, none of the additional j labels are placed in a level 1 node.

It follows that the difference in the leaf label count between $H^*(n-j)$ and $H^*(n+\Delta(n)-j)$ is the difference in the leaf label count between the two intermediate trees, which is j. Thus, we have shown $G^*(n+\Delta(n)-j) = G^*(n-j)+j$. This completes the proof. $\square$

We are now ready to prove the main result of this section.

Theorem 4.10. *Let $H(n)$ be a BLT-tree as described above, and let $G(n)$ be the number of leaf labels of $H(n)$. Then for $n > 2s+4j$, $G(n)$ satisfies the [0,j : 2j,3j]-type meta-Fibonacci recursion (2.2) $G(n) = G(n-s-G(n-j)) + G(n-s-2j-G(n-3j))$*

Proof. By Lemma 4.3, $G(n-s-G(n-j)) + G(n-s-2j-G(n-3j)) = G^*(n-j) + G^*(n-3j)$, so we need to show that $G^*(n-j)+G^*(n-3j) = G(n)$. As in the preceding lemmas we examine four cases based upon the value of $\Delta(n)$, or equivalently, the location of the final label n in $H(n)$.

Case 1: Suppose $0 \leqslant \Delta(n) < j$. Then n is a label on an even level 1 node and $n-2j$ is a label on an odd level 1 node. From Lemmas 4.6, 4.8 and 4.5(1), we obtain $G^*(n-j)+G^*(n-3j) = \frac{G(n)+\delta(n)}{2} + \frac{G(n-2j)+j-\delta(n)}{2} = \frac{G(n)+\delta(n)}{2} + \frac{G(n)-\delta(n)}{2} = G(n)$.

Case 2: Suppose $j \leqslant \Delta(n) < 2j$. Then n is a label on an even level 2 node and $n-2j$ is a label on an odd level 2 node. From Lemmas 4.7, 4.9 and 4.5(2) we obtain $G^*(n-j)+G^*(n-3j) = \frac{G(n)+j}{2} + \frac{G(n-2j)}{2} = \frac{G(n)+j}{2} + \frac{G(n)-j}{2} = G(n)$.

Case 3: Suppose $2j \leqslant \Delta(n) < 3j$. Then n is a label on an odd level 1 node. We must consider three subcases based on the location of $n-2j$.

Subcase 3.1: If $n-2j$ is on an s-node node, then applying Lemmas 4.8, 4.9 and 4.5(3a), we determine that $G^*(n-j)+G^*(n-3j) = \frac{G(n)+j-\delta(n)}{2} + \frac{G(n-2j)}{2} = \frac{G(n)+j-\delta(n)}{2} + \frac{G(n)-j+\delta(n)}{2} = G(n)$.

Subcase 3.2: If $n-2j$ is a label on a level 1 node, and there is no s-node between $n-2j$ and n, then $n-2j$ is the $(j-\delta(n))^{th}$ label on its node. Applying Lemmas 4.8, 4.6 and 4.5(3b), we see that $G^*(n-j)+G^*(n-3j) = \frac{G(n)+j-\delta(n)}{2} + \frac{G(n-2j)+\delta(n)}{2} = \frac{G(n)+j-\delta(n)}{2} + \frac{G(n)-j+\delta(n)}{2} = G(n)$.

Subcase 3.3: If $n-2j$ is a label on a level 1 node, and there is an s-node between $n-2j$ and n, then $n-2j$ is the $(j-\delta(n)+s)^{th}$ label on its node. Applying Lemmas 4.8, 4.6 and 4.5(3b), we see that $G^*(n-j)+G^*(n-3j) = \frac{G(n)+j-\delta(n)}{2} + \frac{G(n-2j)+\delta(n)-s}{2} = \frac{G(n)+j-\delta(n)}{2} + \frac{G(n)-j+s+\delta(n)-s}{2} = G(n)$.

Case 4: Suppose $3j \leqslant \Delta(n) < s+4j$. Then the label n is on either an odd level 2 node or an s-node. We consider three subcases based on the position of $n-2j$ (note that we do not need to have direct information about whether n is on an odd level 2 node or an s-node).

Subcase 4.1: If $n-2j$ is on a level 3 node, then applying Lemmas 4.9 and 4.5(4a) we obtain $G^*(n-j)+G^*(n-3j) = \frac{G(n)}{2} + \frac{G(n-2j)}{2} = \frac{G(n)}{2} + \frac{G(n)}{2} = G(n)$.

Subcase 4.2: The label $n-2j$ is on a level 1 node; let it be the $(j-\beta)^{th}$ label on its node, where $\beta \geqslant 0$. Then applying Lemmas 4.9, 4.6 and 4.5(4b), we observe that $G^*(n-j)+G^*(n-3j) = \frac{G(n)}{2} + \frac{G(n-2j)+\beta}{2} = \frac{G(n)}{2} + \frac{G(n)-\beta+\beta}{2} = G(n)$.

Subcase 4.3: The label $n-2j$ is on a level 2 node. Applying Lemmas 4.9, 4.7 and 4.5(4c) yields the result that $G^*(n-j)+G^*(n-3j) = \frac{G(n)}{2} + \frac{G(n-2j)+j}{2} = \frac{G(n)}{2} + \frac{G(n)-j+j}{2} = G(n)$.

This completes the proof of the final case and the proof of Theorem 4.10. □

That is, $G(n)$, the number of labels in the leaves of the BLT tree $H(n)$, equals $B(n)$, where $B(n)$ is defined by (2.2) together with the following initial conditions: for all i in the interval $0 \leqslant i \leqslant 2s+4j$, $B(i) = G(i)$.

To date, in contrast to the situation in Section 3, we have not found any way to extend the above result for $s < 0$. In fact, there doesn't seem to be any set of initial conditions for which the recursion (2.2) with $s < 0$ has a solution.

5 Applications of the Combinatorial Interpretations

In this section we demonstrate how the combinatorial interpretations that we have developed for the solution sequences to the meta-Fibonacci recursions (2.1) and (2.2) provide a powerful tool for deriving properties of these sequences, such as closed forms, asymptotic properties, and frequency functions. The *frequency function* associated with a given integer sequence $L(n)$ is $f_L(m) := |L^{-1}(m)|$, that is, $f_L(m)$ is the number of n such that $L(n) = m$. The frequency function is especially useful when dealing with slow (or at least monotone) integer sequences, since all of the appearances of a given m appear consecutively.

We begin by providing a simple derivation for the frequency function of $G_{s,j}(n)$ (or $G(n)$ where there is no confusion), the leaf-label counting sequence which solves (2.2). If n is the m^{th} leaf label on H but is not the last label on its node, then the frequency function $f_G(m) = 1$, since n will be immediately followed in H by another leaf label. If n is the m^{th} leaf label on H and is the last label on its node but not the last label on its subtree, then n will be followed immediately by a level 2 node with j nonleaf entries, and then another level 1 node, so G will assume the value m at n and continue until $n+j+1$, the next leaf label. Thus, $f_G(m) = 1+j$. If n is the m^{th} leaf label on H and is the last label on its subtree, then n will be immediately followed by a s-node, a level 2 node, and then a level 1 node, since s-nodes occur only directly after the end of subtrees. Therefore, n is followed by exactly $s+j$ nonleaf labels, so $f_G(m) = 1+s+j$. This covers all possible cases, and we have determined that the frequency function for $G(n)$ is:

$$f_G(m) = \begin{cases} 1 & \text{if } m \not\equiv 0 \bmod j \\ 1+s+j & \text{if } m = 2^d j \text{ for some } d > 0 \\ 1+j & \text{otherwise} \end{cases}$$

Using our combinatorial interpretation we now determine a closed form expression for $G_{s,j}(n)$ that generalizes very broadly the formulas derived in [3] for the special cases $s=0, j=1$ and $s=1, j=1$.

Theorem 5.1. *For fixed s and j, let $G_{s,j}(n)$ be the number of labels that occur in the leaves of $H_{s,j}(n)$. For convenience we usually write $G(n)$ in place of $G_{s,j}(n)$ where the* *abbreviated notation is clear. Then for $n \leqslant j$, $G(n) = n$. For $m \in \mathbb{N}$ and n in the interval $(m-1)s + (2^m - 1)j < n \leqslant ms + (2^{m+1} - 1)j$, set $f(n) = max\{n - ms - (2^m - 1)j, 0\}$. Then $G(n) = (2^{m-1}j) + (\lfloor \frac{f(n)}{2j} \rfloor j) + max\{f(n) - \lfloor \frac{f(n)}{2j} \rfloor 2j - j, 0\}$.*

Proof. For $n \leqslant j$ the result is immediate from the definition of $G(n)$ (the isolated node of the tree $H(n)$ contains j labels). Suppose that n is in the interval $(m-1)s + (2^m - 1)j < n \leqslant ms + (2^{m+1} - 1)j$. This is equivalent to the statement that the label n in the tree $H(n)$ is on the m^{th} rooted subtree, that is, n is either on the m^{th} s-node or is on a j-node that is a descendant of the m^{th} s-node. To see this, note that the term $(m-1)s$ in the lower bound accounts for the number of labels in the first $(m-1)$ s-nodes, while the term $(2^m - 1)j$ counts the j labels in each of the $(1 + 2 + \ldots + 2^{m-1})$ j-nodes that are descendants of these initial $m-1$ s-nodes. In a similar way we can show that the upper bound is the last label on the last leaf that descends from the m^{th} s-node.

For any such label n on the m^{th} rooted subtree, we compute $G(n)$, the number of labels in the leaves of $H(n)$. First, it is easy to see that $(1+1+2+\cdots+2^{m-2})j = (2^{m-1}j)$ counts the number of labels in the leaves prior to the m^{th} rooted subtree of $H(n)$. Next, note that $f(n)$ is the number of labels in all of the j-nodes in the m^{th} rooted subtree. This is because, by our earlier observations, $n - (m-1)s - (2^m - 1)j$ is the number of labels in the m^{th} rooted subtree. If we subtract the s entries on the m^{th} s-node itself (taking 0 instead if we get a negative number) we have the number of entries after the m^{th} s-node. A fully filled level 2 node and its level 1 descendant contain $2j$ nodes (taken together). Therefore, $\lfloor \frac{f(n)}{2j} \rfloor$ calculates the number of completely filled pairs of level 2 and level 1 nodes on the m^{th} rooted subtree, so the term $\lfloor \frac{f(n)}{2j} \rfloor j$ counts the labels in the level 1 nodes (leaves) of these pairs on the m^{th} rooted subtree.

The only possible leaf labels not yet counted are the labels in the last leaf if this node contains fewer than j labels. But since $f(n)$ is the number of labels in all of the j-nodes in the m^{th} rooted subtree, it follows that $f(n) - \lfloor \frac{f(n)}{2j} \rfloor 2j$ is 0 if all the leaves contain j labels, or else is the number of labels in any pair of level 2 and level 1 j-nodes that do not contain their full complement of labels (note that this can only happen for the last level 2 and possibly level 1 j-node). Further, the number of leaf labels associated with such a partly filled in pair of j-nodes can only be non-zero if the level 2 node contains j labels. Thus, we subtract j and take 0 if this difference is negative; that is, the number of leaf labels in the last partly filled in leaf, if there is one, is $max\{f(n) - \lfloor \frac{f(n)}{2j} \rfloor 2j - j, 0\}$. This completes the proof. □

We now focus our attention on $R_{s,j,k}(n)$, the number of labels in the leaves of the tree $T_{s,j,k}(n)$ and the solution to the meta-Fibonacci recursion (2.1) (we drop the subscripts and use $R(n)$ and $T(n)$ where there will be no confusion). The situation for $R(n)$ is more complicated than $H(n)$. Unlike $H(n)$, $R(n)$ does not seem to have a comparably simple closed form expression. However, the combinatorial interpretation for $R(n)$ provides a very elegant basis for deriving certain asymptotic properties of this sequence. It is well known that any slow solution to the meta-Fibonacci recursion (2.1) approaches $\frac{k-1}{k}n$ as n

increases.[12] However, using the combinatorial interpretation for $R(n)$ we obtain a much stronger result, namely, $R(n) = \frac{k-1}{k}n + O(log_k(n))$. In what follows, for simplicity, we focus on the case $k = 2$ with $s \geqslant 0$; in fact it is possible to adapt our argument for general k and s.

We apply a notion we introduced in Section 3. Recall that the m^{th} binary subtree of T is the subtree containing the $(m-1)^{st}$ s-node and all of its descendants. For example, in Figure 3.1, the fourth binary subtree is $T_{2,3,2}(42)$. We define the first binary subtree to be just the first j-node.

Since the m^{th} binary subtree is a binary tree with m levels, it has $2^m - 1$ nodes in total. By definition, $m-1$ of these are s-nodes so the remaining $2^m - m$ are j-nodes. So there are $(m-1)s + (2^m - m)j$ labels on the nodes of the m^{th} binary subtree. For ease of reference we define $\lambda(m) = (m-1)s + (2^m - m)j$. Further, since the m^{th} binary subtree has m levels, it has 2^{m-1} leaves, so the leaves of the m^{th} binary subtree contain $2^{m-1}j$ labels. By what we have just observed, the last label in the last leaf of the m^{th} binary subtree is $\lambda(m)$.

By a straightforward calculation it is immediate that $2^{m-1}j$, the number of labels on the leaves of the m^{th} binary subtree, can be expressed as $\frac{\lambda(m)+j+(m-1)(j-s)}{2}$. From this formula we deduce that in the special case where $T(n)$ is the m^{th} binary subtree (in other words, $n = \lambda(m)$) then $R(n) - \frac{n}{2}$ is precisely $\frac{j+(m-1)(j-s)}{2}$.

For any value of n, with fixed s and j, it is clear that the tree $T(n)$ is a subtree of a smallest binary subtree; that is, there exists a smallest m such that $T(n)$ is a subtree of the m^{th} binary subtree but not of the $(m-1)^{st}$ binary subtree. Note that this means $\lambda(m-1) = (m-2)s + (2^{m-1} - m + 1)j < n \leqslant (m-1)s + (2^m - m)j = \lambda(m)$. It follows that $T(n)$ contains the first $n - \lambda(m-1)$ labels from m^{th} binary subtree that are not in the $(m-1)^{st}$ binary subtree; the nodes in $T(n)$ on which these labels reside are the $(m-1)^{st}$ s-node, its right child, and the descendants of the right child. We refer to the portion of $T(n)$ consisting of these nodes as the m^{th} *binary residual* of $T(n)$, or just the binary residual when m is clear. Let $r(n,m)$ be the number of these $n - \lambda(m-1)$ labels that are not in the $(m-1)^{st}$ binary subtree and that are on the leaves of $T(n)$. Equivalently, $r(n,m)$ is the number of leaf labels in the m^{th} binary residual of $T(n)$. Then we have the following result, which bounds the difference between the number of labels in the leaves and half the total number of labels of the binary residual of the tree $T(n)$.

Lemma 5.2. *For fixed n, s and j let m be the smallest integer such that $T(n)$ is a subtree of the m^{th} binary subtree. Let $\lambda(m-1)$ be the number of labels in the $(m-1)^{st}$ binary subtree of T. Let $r(n,m)$ be the number of leaf labels in the binary residual of $T(n)$. Then* $min(\frac{1}{2}, \frac{s-j}{2}) \leqslant \frac{n-\lambda(m-1)}{2} - r(n,m) \leqslant \frac{(m-2)j+s}{2}$.

Proof. First we show that the boundary conditions are the best possible. We begin with the lower bound. Note that if $s - j \leqslant 1$ then the minimum is $\frac{s-j}{2}$, and otherwise, when $s - j > 1$, the lower bound is $\frac{1}{2}$. However, regardless of the value of $s - j$, we show in what follows that $\frac{n-\lambda(m-1)}{2} - r(n,m)$ will assume both values, at two distinct values of n.

[12]See [8], [5]. [2], [11], and [20] for examples of recursions with a similar structure where the solutions display this same behavior.

First we prove that when n is the last label in the m^{th} binary subtree, that is, when $n = \lambda(m)$, then $r(n,m)$ assumes the value $\frac{n-\lambda(m-1)+j-s}{2}$. To see this, consider the binary residual of $T(n)$ (the portion of $T(n)$ that consists of the $(m-1)^{st}$ s-node, the right child of the $(m-1)^{st}$ s-node, and all of its descendants). Since this portion of $T(n)$ consists of the $(m-1)^{st}$ s-node and then a binary tree with $m-1$ levels, there are $2^{m-1}-1$ j-nodes, of which 2^{m-2} are leaves. There are $2^{m-2}j$ leaf labels in this portion of $T(n)$ and these include all of the leaf labels in the binary residual. So $r(n,m) = 2^{m-2}j$. If we change the $(m-1)^{st}$ s-node to a j-node, then include it in the count, we have 2^{m-1} total nodes, and exactly half are leaves. So, in this case, $r(n,m) = \frac{n-\lambda(m-1)+j-s}{2}$ (note that the $+j-s$ transforms the s-node to a j-node), and thus $\frac{s-j}{2} = \frac{n-\lambda(m-1)}{2} - r(n,m)$.

Now consider the subtree $T(n)$ of T when n is the first label in the binary residual, meaning $n = \lambda(m-1)+1$. This label n must appear in the $(m-1)^{st}$ s-node if $s > 0$ and otherwise in a j-node that is not a leaf. So for this n we have that $n - \lambda(m-1) = 1$ and $r(n,m) = 0$. Thus, $\frac{n-\lambda(m-1)}{2} - r(n,m) = \frac{1}{2}$. Thus, we have shown that for any s and j there are always two distinct values of n such that $\frac{n-\lambda(m-1)}{2} - r(n,m)$ assumes both possible lower bounds. Which one of $\frac{1}{2}$ and $\frac{s-j}{2}$ is smaller determines the greatest lower bound.

We now illustrate the value of n for which the difference $\frac{n-\lambda(m-1)}{2} - r(n,m)$ assumes the upper bound $\frac{(m-2)j+s}{2}$. Let n be the last label in preorder before the first leaf label of the binary residual of $T(\lambda(m)-1)$, that is, n occurs immediately before the first leaf in the m^{th} but not $(m-1)^{st}$ binary subtree. For example, if $s = 2, j = 3$ and $m = 4$ then $n = 27$. See Figure 3.1, where it is easily verified that $\frac{n-\lambda(m-1)}{2} - r(n,m) = 4 = \frac{(m-2)j+s}{2}$. Note that for such a value of n we always have $r(n,m) = 0$ as we have not yet filled in any leaf labels in the binary residual. Also, by the nature of the preorder for the labeling of T, the only nodes that receive labels up to such a value of n are the $(m-1)^{st}$ s-node, the right child of this s-node, the left child of this latter node, and the left children of successive nodes all the way down to the parent of the leftmost leaf in the binary residual. But this sequence of nodes comprises 1 s-node and $m-2$ j-nodes. Hence, $\frac{n-\lambda(m-1)-(m-2)j-s}{2} = 0$, so $\frac{n-\lambda(m-1)}{2} - r(n,m) = \frac{(m-2)j+s}{2}$.

Finally we show that the stated inequalities hold for any n in the indicated interval. First, we show that if $s - j > 1$ then $\frac{1}{2} \leqslant \frac{n-\lambda(m-1)}{2} - r(n,m)$, and otherwise $\frac{s-j}{2} \leqslant \frac{n-\lambda(m-1)}{2} - r(n,m)$. Suppose that n_0 is a label in the m^{th} binary subtree and is on or after (in preorder) the first leaf in the binary residual. We show that that the value of $\frac{n_0-\lambda(m-1)}{2} - r(n_0,m)$ is bigger than or equal to the value of the corresponding difference for $n' = \lambda(m)$, that is, $\frac{\lambda(m)-\lambda(m-1)}{2} - r(\lambda(m),m) \leqslant \frac{n_0-\lambda(m-1)}{2} - r(n_0,m)$.

Our general approach is to add labels and nodes to $T(n)$ in certain discrete blocks so as ultimately to end up with the tree $T(\lambda(m))$. We may need to do so several times. For any positive integer k, define n_k to be the last label on the rightmost leaf descendant of the node containing the label $n_{k-1}+1$. The labels added in the k^{th} discrete block consist of those numbered from $n_{k-1}+1$ to n_k. Note that the labels added at the k^{th} step comprise a full binary tree, less the number α_k of labels on the node containing the label $n_{k-1}+1$. By our construction $\alpha_k = 0$ unless $k = 1$. For example, in Figure 3.1, if the starting node $n_0 = 55$ then $n_1 = 56$, $\alpha_1 = 2$, $n_2 = 59$, $\alpha_2 = 0$, $n_3 = 68$, $\alpha_3 = 0$, and we end up with the tree $T(\lambda(5))$ at $n_4 = 90 = \lambda(5)$.

We show that after each such block is added, the value of $\frac{n_k-\lambda(m-1)}{2} - r(n_k, m)$ is no bigger than $\frac{n_{k-1}-\lambda(m-1)}{2} - r(n_{k-1}, m)$. Recall that the total number of nodes on any full binary tree is twice the number of leaf nodes, less 1. Thus, for our labeled binary trees with j labels per node, the total number of labels on any full binary tree is twice the number of leaf labels less j. Therefore $(n_k - n_{k-1})$, the number of labels added, equals twice the number of leaf labels added, less the j labels on the one less node, then corrected by $-\alpha_k$. So, $n_k - n_{k-1} = 2(r(n_k, m) - r(n_{k-1}, m)) - j - \alpha_k$ so we have $n_k - n_{k-1} - 2(r(n_k, m) - r(n_{k-1}, m)) \leqslant 0$. This shows that $\frac{n_k-\lambda(m-1)}{2} - r(n_k, m) - (\frac{n_{k-1}-\lambda(m-1)}{2} - r(n_{k-1}, m)) \leqslant 0$ as desired. So we have shown that if n is on or after the first leaf of the m^{th} but not $(m-1)^{st}$ binary subtree, than the value of $\frac{n-\lambda(m-1)}{2} - r(n, m)$ is larger than the value when $n = \lambda(m)$ (that is, at the end of the binary subtree).

In the case where n is before this first leaf of the binary residual, $r(n, m) = 0$. It is obvious that reducing the value of n to $\lambda(m-1) + 1$ will not increase the value of $\frac{n-\lambda(m-1)}{2} - r(n, m)$, since we delete no leaf entries in doing so. Note that we must stay past the $(m-1)^{st}$ binary subtree by our assumption about the minimum value of n.

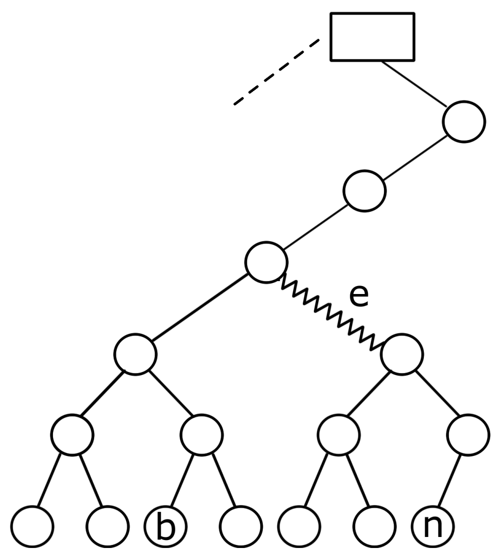

Figure 5.1: The binary residual of $T(n)$.

Now we show that for the values of n we are considering, namely, the condition that n must be in the m^{th} binary residual, $\frac{n-\lambda(m-1)}{2} - r(n, m) \leqslant \frac{(m-2)j+s}{2}$ holds. We begin with the claim that $\frac{n-\lambda(m-1)}{2} - r(n, m)$ is maximized when n's parent, grandparent, etc. are all left children, except for the right child of the s-node.

Suppose this claim is false, that is, for some n with a right child in its ancestry (other than the right child of the $(m-1)^{st}$ s-node), $\frac{n-\lambda(m-1)}{2} - r(n, m)$ is maximal. See Figure 5.1 where we have indicated the location of the label n. Locate the node that is the

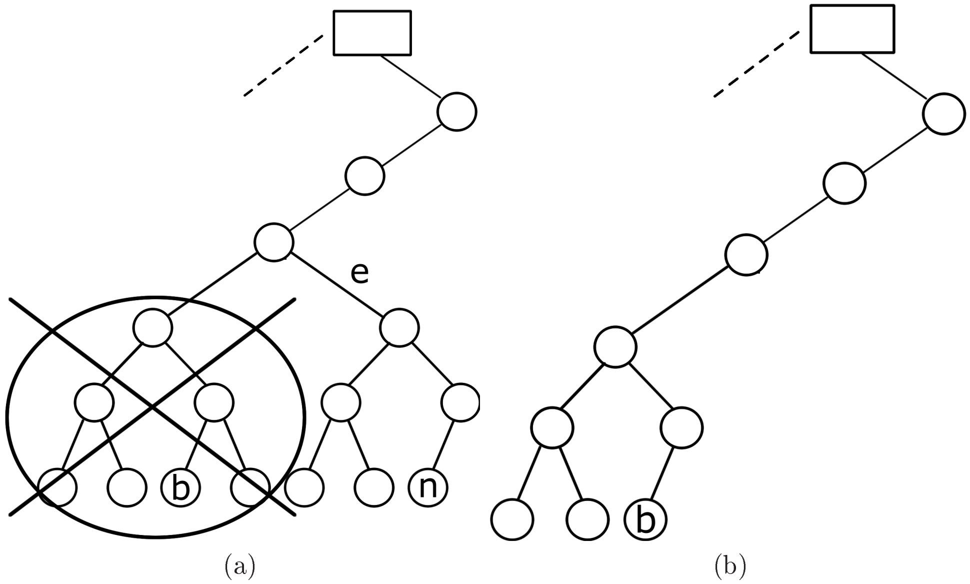

Figure 5.2: (a) $T(n)$ with X and all of its descendants removed. (b) $T(b)$

uppermost right child from which n descends (other than the right child of the $(m-1)^{st}$ s-node). In Figure 5.1 this node is called Z. We define Y to be the parent of Z; denote by X the left child of Y (we will return to X later). Call e the right edge descending from Y. We consider the label b on $T(n)$ which is found by following the same path from Y as n, except that instead of proceeding down e, we instead take the left edge at the first step. Note that X,Y and Z refer to entire j-nodes while each of b and n is an individual label among the j on its respective j-node. For example, in Figure 3.1, if $n = 35$, then $b = 26$, if $n = 78$, then $b = 57$ and if $n = 57$, then $b = 54$.

We compare $\frac{n-\lambda(m-1)}{2} - r(n,m)$ and $\frac{b-\lambda(m-1)}{2} - r(b,m)$ and we show that $\frac{b-\lambda(m-1)}{2} - r(b,m)$ is larger. In order to evaluate $\frac{b-\lambda(m-1)}{2} - r(b,m)$, we take $T(n)$ and remove X and all of its descendants. See Figure 5.2a. Note that doing this gets us a tree that is identical to $T(b)$ up to relabeling (and changing e to a left turn); see Figure 5.2b.

By preorder, since the edge e indicates a right turn, the portion of $T(n)$ that is X and all of its descendants is a full binary tree. If Y is on the y^{th} level in $T(n)$, then X is on the $(y-1)^{th}$ level so deleting X and all of its descendants removes 2^{y-2} leaf nodes and $2^{y-1} - 1$ total nodes. This means that the total number of nodes we are removing in going from $T(n)$ to $T(b)$ is one less than the number of leaf nodes we are removing. For example in Figure 5.2a, we are removing four leaf nodes and seven total nodes. Since we are only removing j-nodes with their full complement of j labels each, this means that the total number of labels removed, $n - b$, is twice the number of leaf labels removed, $r(n,m) - r(b,m)$, minus j. We write this as $n - b = 2(r(n,m) - r(b,m)) - j$, which implies $n - 2r(n,m) = b - 2r(b,m) - j$. Dividing by 2 we have $\frac{n}{2} - r(n,m) = \frac{b}{2} - r(b,m) - \frac{j}{2}$. Finally, we subtract $\frac{\lambda(m-1)}{2}$ from both sides, yielding the equation $\frac{b-\lambda(m-1)}{2} - r(b,m) =$

$\frac{j}{2} + \frac{n-\lambda(m-1)}{2} - r(n,m)$. But this contradicts our hypothesis that $\frac{n-\lambda(m-1)}{2} - r(n,m)$ is maximal, as we have just shown that $\frac{b-\lambda(m-1)}{2} - r(b,m)$ is larger by $\frac{j}{2}$. Therefore, our claim must be true: $\frac{n-\lambda(m-1)}{2} - r(n,m)$ can only be maximal when n's parent, grandparent, etc. are all left children, except for the right child of the s-node.

It remains to show that $\frac{n-\lambda(m-1)}{2} - r(n,m)$ is maximal when, in addition to having an all left ancestry as described above, n is the last label on a level 2 node (ie, n comes just before the first leaf of the binary residual).

Clearly, given any n on level 2 or above, we could increase the value of $\frac{n-\lambda(m-1)}{2} - r(n,m)$ by filling in the rest of the node with n and all of its left children until we reached the end of a level 2 node, since this would not add any leaves. So, this means that the only possible candidates for the maximal position are the label right before the first leaf label in the binary residual, or the leaf labels on that first leaf. It is obvious that adding leaf labels decreases $\frac{n-\lambda(m-1)}{2} - r(n,m)$, as each leaf label added increases $\frac{n-\lambda(m-1)}{2}$ by $\frac{1}{2}$ and increases $r(n,m)$ by 1, the net effect of which is to decrease $\frac{n-\lambda(m-1)}{2} - r(n,m)$ by $\frac{1}{2}$ per leaf label added.

Therefore it must be true as claimed that $\frac{n-\lambda(m-1)}{2} - r(n,m)$ can only be maximal when n is the final label before the first leaf label that is not in the $(m-1)^{st}$ binary subtree. This establishes the desired result. $\square$

Collecting the above results we readily obtain the following bounds on $\frac{n}{2} - R(n)$.

Theorem 5.3. *Suppose* $(m-2)s+(2^{m-1}-m+1)j < n \leqslant (m-1)s+(2^m-m)j)$*, ie, n is in the m^{th} binary residual. Then* $min\{\frac{1-j+(m-2)(s-j)}{2}, \frac{s-2j+(m-2)(s-j)}{2}\} \leqslant \frac{n}{2} - R(n) \leqslant \frac{ms-s-j}{2}$.

Proof. From Lemma 5.2 we have upper and lower bounds for the part of $\frac{n}{2} - R(n)$ that comes from binary residual of $T(n)$, namely $min(\frac{1}{2}, \frac{s-j}{2}) \leqslant \frac{n-\lambda(m-1)}{2} - r(n,m) \leqslant \frac{(m-2)j+s}{2}$. By the definition of $r(n,m)$, $r(n,m) + R(\lambda(m-1)) = R(n)$, so to change these bounds into bounds on $\frac{n}{2} - R(n)$, we need to add $\frac{\lambda(m-1)}{2}$ and subtract $R(\lambda(m-1))$.

Applying this to the entire inequality from Lemma 5.2 we get $min(\frac{1}{2}, \frac{s-j}{2}) + \frac{\lambda(m-1)}{2} - R(\lambda(m-1)) \leqslant \frac{n}{2} - R(n) \leqslant \frac{(m-2)j+s}{2} + \frac{\lambda(m-1)}{2} - R(\lambda(m-1))$.

Recall from our discussion before Lemma 5.2 that $R(\lambda(m)) - \frac{\lambda(m)}{2} = \frac{j+(m-1)(j-s)}{2}$, so $\frac{\lambda(m-1)}{2} - R(\lambda(m-1)) = -\frac{j+(m-2)(j-s)}{2}$. We substitute this value into our previous inequality and simplify to get the desired inequality. $\square$

Note that the above result is much stronger than just stating that $R(n)$ approaches $\frac{n}{2}$ as the difference $\frac{n}{2} - R(n)$ grows linearly with m whereas the value of n increases exponentially with m.

Corollary 5.4. $R(n) = \frac{n}{2} + O(log_2(n))$ *as* $n \to \infty$

Proof. If n is on the m^{th} residual subtree, then $(m-2)s + (2^{m-1} - m + 1)j < n \leqslant (m-1)s + (2^m - m)j)$. It follows that $m \in O(log_2(n))$. Now apply Theorem 5.3 to get the desired result. $\square$

Lastly, we use the combinatorial interpretation for the sequence $R(n)$ to derive its frequency function, which is significantly more complicated than the frequency function for $G(n)$. To begin, we first define the k-ary ruler $r_q^{(k)}$ as 1 plus the k-adic valuation of q. That is, if $q = k^p v$ where v is not divisible by k, then $r_q^{(k)} = p+1$. Next we define a function ϕ that we claim is the frequency function we seek. We start with the subcase $s = 0$, in which

$$\phi_{0,j,k}(m) = \begin{cases} 1 & \text{if } m \not\equiv 0 \bmod kj \\ jr_{m/j}^{(k)} - j + 1 & \text{if } m \equiv 0 \bmod kj \end{cases}$$

In general, for $s \geqslant 0$ we define $\phi_{s,j,k}$ in terms of the special case when $s = 0$:

$$\phi_{s,j,k}(m) = \begin{cases} \phi_{0,j,k}(m) & \text{if } m \neq jk^p \text{ for some nonnegative integer } p \\ \phi_{0,j,k}(m) + s & \text{if } m = jk^p \text{ for some nonnegative integer } p \end{cases}$$

We now prove that $\phi_{s,j,k}(m) = f_{R_{s,j,k}}(m)$, the frequency function of $R_{s,j,k}(n)$.

Theorem 5.5. *Let $R_{s,j,k}(n)$ be the leaf label for the tree $T_{s,j,k}(n)$, and $\phi_{s,j,k}(n)$ be as defined above. Let $f_{R_{s,j,k}}(n)$ be the frequency function for $R_{s,j,k}(n)$. Then $f_{R_{s,j,k}}(m) = \phi_{s,j,k}(m)$.*

Proof. We begin by proving that $\phi_{0,j,k}(m) = f_{R_{0,j,k}}(m)$. We consider three cases:

Case 1: $m = jk^q$ for some nonnegative integer q. Note that jk^q is precisely the number of leaf labels in the $(q+1)^{st}$ complete subtree. Therefore, the leaf label counting sequence $R_{0,j,k}(n)$ will first assume the value m at the last leaf label in the $(q+1)^{st}$ complete subtree and will first assume the value $m+1$ at the first leaf label in the $(q+2)^{nd}$ residual subtree. Recall that the $(q+2)^{nd}$ complete subtree descends from the $(q+1)^{st}$ s-node which is a level $q+2$ node, therefore, there are q nonleaf j-nodes between the $(q+1)^{st}$ s-node and the first leaf of the $(q+2)^{nd}$ residual subtree. Hence $R_{0,j,k}(n)$ assumes the value m one time (on the last leaf label of the $(q+1)^{st}$ complete subtree) plus jq times (on the aforementioned first q nonleaf j-nodes of the $(q+2)^{nd}$ residual subtree. Since $q = r_{m/j}^{(k)} - 1$, we have shown $f_{R_{0,j,k}}(jk^q) = jq + 1 = jr_{k^q}^{(k)} - j + 1 = \phi_{0,j,k}(jk^q)$.

Case 2: $m < j$. Note if $j = 1$, this case is vacuously proven. Otherwise, since $T_{0,j,k}(n)$ begins with a j-node, the first j labels will be leaf labels and so the frequency of m for $m < j$ is 1, as is $\phi_{0,j,k}(m)$.

Case 3: $m > j$ and $m \neq jk^q$ for any nonnegative integer q. Let z be the largest integer such that $jk^z < m$. We begin this case by showing that $\phi_{0,j,k}(m) = \phi_{0,j,k}(m - jk^z)$. If j does not divide m, then j does not divide $m - jk^z$ either, so it follows that $\phi_{0,j,k}(m) = \phi_{0,j,k}(m - jk^z) = 1$. On the other hand, if j does divide m, then $m = jxk^y$ where x is a natural number such that k does not divide x. In order to show $\phi_{0,j,k}(m) = \phi_{0,j,k}(m - jk^z)$, it suffices to show $r_{m/j}^{(k)} = r_{(m-jk^z)/j}^{(k)}$. If $y < z$, then $r_{(m-jk^z)/j}^{(k)} = r_{xk^y - k^z}^{(k)} = r_{k^y(x-k^{z-y})}^{(k)} = y + 1 = r_{m/j}^{(k)}$, where the penultimate equality holds because $z - y > 0$ and thus k does not divide $x - k^{z-y}$. If $y = z$, then note that $1 < x < k$, since if $x = 1$ or $x = k$, our hypothesis that $m \neq jk^q$ for any q is false, and if $x > k$, then $jk^{z+1} < m$, which contradicts z being the largest integer such that $jk^z < m$. Thus, $r_{(m-jk^z)/j}^{(k)} = r_{xk^z - k^z}^{(k)} = r_{k^z(x-1)}^{(k)} = z + 1 = r_{m/j}^{(k)}$, where the penultimate equality holds because, as $1 < x < k$, $0 < x - 1 < k - 1$ and hence

k cannot divide $x-1$. Clearly, $y > z$ is impossible by the assumption that z is the largest integer such that $jk^z < m$. So we have shown that $\phi_{0,j,k}(m) = \phi_{0,j,k}(m - jk^z)$.

Next, we show that $f_{R_{0,j,k}}(m) = f_{R_{0,j,k}}(m - jk^z)$. Recall that the $(z+2)^{nd}$ residual subtree (the one to which the m^{th} leaf label belongs) consists of an s-node, and $k-1$ copies of the $(z+1)^{st}$ complete subtree (with the s-nodes replaced by j-nodes). Since $m \neq jk^q$ for any q, the m^{th} leaf label is not the last label in its complete subtree. We now go to subcases.

Sub-case 3.1: if the m^{th} leaf label belongs to the p^{th} of these $k-1$ copies of the $(z+1)^{st}$ complete subtree for $p > 1$, then the $(m - jk^z)^{th}$ leaf label occupies the same position in the $(p-1)^{st}$ copy. Hence $f_{R_{0,j,k}}(m) = f_{R_{0,j,k}}(m - jk^z)$.

Sub-case 3.2: the m^{th} leaf label belongs to the first of these $k-1$ copies of the $(z+1)^{st}$ complete subtree, but is not the last label in that copy. Then the $(m - jk^z)^{th}$ leaf label occupies the same relative position in the $(z+1)^{st}$ complete subtree as the m^{th} leaf label does in the first copy of the $(z+1)^{st}$ complete subtree that lies in the $(z+2)^{nd}$ residual subtree. The only difference between the original $(z+1)^{st}$ complete subtree in which the $(m - jk^z)^{th}$ leaf label lies, and the first copy of the $(z+1)^{st}$ complete subtree in which the m^{th} leaf label lies, is that the original has empty (since $s = 0$) s-nodes in place of the upper left j-nodes in the first copy. However, the upper left j-nodes in the first copy come in preorder before the leaves in the first copy (unlike the s-nodes, which are dispersed in preorder between leaves). But since the s-nodes in the original are empty, this difference has no effect on the number of nonleaf labels following the m^{th} and $(m - jk^z)^{th}$ leaf labels. Therefore, because they have the same relative positions and the change between s and j-nodes from the original to the copy does not matter, $f_{R_{0,j,k}}(m) = f_{R_{0,j,k}}(m - jk^z)$.

Sub-case 3.3: if the m^{th} leaf label is the last label in the first copy of the $(z+1)^{st}$ complete subtree that lies in the $(z+2)^{nd}$ residual subtree, then the m^{th} leaf label is immediately followed in preorder by the 2^{nd} copy of the $(z+1)^{st}$ complete subtree. Observe that because $m \neq jk^q$ for any q, this sub-case only arises if $k > 2$. Then $m - jk^z$ will be the last label in the $(z+1)^{st}$ complete subtree, and hence will be followed by an s-node (empty, as $s = 0$) and then the first copy of the $(z+1)^{st}$ complete subtree lying in the $(z+2)^{nd}$ residual subtree. Thus the labels following both the m^{th} and $(m - jk^z)^{th}$ leaf labels are the labels of a copy of the $(z+1)^{st}$ complete subtree lying in the $(z+2)^{nd}$ residual subtree, so $f_{R_{0,j,k}}(m) = f_{R_{0,j,k}}(m - jk^z)$.

Now that we have shown that if $m > j$ and $m \neq jk^q$ for any nonnegative integer q, then $\phi_{0,j,k}(m) = \phi_{0,j,k}(m - jk^z)$ and $f_{R_{0,j,k}}(m) = f_{R_{0,j,k}}(m - jk^z)$ where z is the largest integer such that $jk^z < m$. So, given such m, replace m with $m' = m - jk^z$, and we know that $\phi_{0,j,k}(m) = f_{R_{0,j,k}}(m)$ if and only if $\phi_{0,j,k}(m') = f_{R_{0,j,k}}(m')$. If we repeat this process, we will eventually either reach a $m' < j$ or an m' such that $m' = jk^q$ for some nonnegative integer q, at which point either Case 1 or Case 2 covers that value of m'. Therefore, $\phi_{0,j,k}(m) = f_{R_{0,j,k}}(m)$ for all j, k, m.

Finally, we show that for $s \geqslant 0$, $\phi_{s,j,k}(m) = f_{R_{s,j,k}}(m)$. To do so, it suffices to show that if $m = jk^p$ for some integer p, then $f_{R_{s,j,k}}(m) = f_{R_{0,j,k}}(m) + s$, and otherwise $f_{R_{s,j,k}}(m) = f_{R_{0,j,k}}(m)$. But this is obvious, since the only difference between $T_{s,j,k}$ and $T_{0,j,k}$ is the s-nodes, which appear directly after the end of complete subtrees, and hence their inclusion will add s to the frequency of $m = jk^p$ for p any nonnegative integer, since jk^p is the leaf label count in the $(p+1)^{st}$ complete subtree. $\square$

6 Revisiting the Initial Conditions

Earlier we noted that the behavior of the solutions to the recursions (2.1) and (2.2) depends on the initial conditions that are assumed. Here we explore the role of the initial conditions further.

In Section 2 we described the empirical investigation of (1.1) that led to the identification of the parameters for the recursions (2.1) and (2.2). But note that in a manner of speaking we have carried out a "bait and switch" regarding the initial conditions. The initial conditions used in Section 2 to identify (2.1) and (2.2) consisted of some number of $1s$ followed by a single 2. But our analysis of these recursions in Sections 3 and 4 via the infinite trees T and H incorporated initial conditions that follow the trees. These initial conditions (for $j > 1$) are very different from a string of $1s$ followed by a single 2. Indeed, the first j initial conditions for each of the trees is the sequence of integers $1, 2, 3, \ldots, j$.

So far we have proved nothing about the solutions to (2.1) and (2.2) with the initial conditions from Section 2, although we did speculate there that based on the empirical data it appears that these sequences, like the ones described in Sections 3 and 4, are slow growing. Rather than prove directly that these solutions are slow, we ask more generally if it is possible to identify which sets of initial conditions together with the recursion (2.1) or (2.2) will lead to a slow growing solution. It is evident that if the entire sequence is to be slow then it is necessary that the initial conditions are slow. [13] However, this condition is not sufficient: for example, the recursion (2,3 : 5,6) together with the slow six initial conditions $1, 2, 3, 4, 5, 6$ generates the solution whose next six values are $2, 2, 2, 7, 3, 10$ so the sequence is not slow.

At this stage we provide a test that identifies whether a given set of initial conditions for the recursion (2.1) in the special case with $s \geqslant 0$ and $k = 2$ generates a slow growing sequence.[14] Notice that in this case we must have at least $2j$ initial conditions to start the calculation of the solution sequence from the recursion.

Fix $s \geqslant 0$, $j > 0$, $k = 2$ and the set of r initial conditions, $r \geqslant 2j$. If the solution is to be slow, these r initial conditions must be slow. Let $A(n) = A_{s,j,2}(n)$ be the sequence generated by the recursion (2.1) and the given r initial conditions. Observe that $A(n)$ may become undefined. For notational convenience, define $A^1(n) = A(n - s - A(n - j))$ and $A^2(n) = A(n - s - j - A(n - 2j))$ (note $A^2(n) = A^1(n - j)$). Further, define the difference sequences $\Lambda(n) := A(n + 1) - A(n)$, $\Lambda^1(n) = A^1(n + 1) - A^1(n) = A(n + 1 - s - A(n + 1 - j)) - A(n - s - A(n - j))$ and $\Lambda^2(n) = A^2(n + 1) - A^2(n) = A(n + 1 - s -$

[13] We have identified sets of initial conditions that are not slow and where the resulting solution is *eventually slow*, by which we mean that the sequence is slow from some point on. An example is the recursion (1,2 : 3,4) with the initial conditions $5, 3, 1, 2, 2, 2, 2$.

[14] Substantial empirical evidence suggests that a comparable test may hold for the recursion (2.2) but so far we have not been able to prove it.

$j - A(n+1-2j)) - A(n-s-j-A(n-2j))$. Clearly, $\Lambda(n) = \Lambda^1(n) + \Lambda^2(n)$ and $A(n)$ is slow if and only if for all n, $\Lambda(n) \in \{0, 1\}$.

Lemma 6.1. *Suppose that $A(n)$ is slow up to at least the term $m > r$ (that is, for all $p < m$, $\Lambda(p) \in \{0, 1\}$). Then for $i \in \{1, 2\}$, $\Lambda^i(m) \in \{0, 1\}$.*

Proof. Since $j > 0$ and $A(n)$ is slow up to at least m, $\Lambda(m-j) \in \{0, 1\}$. If $\Lambda(m-j) = 1$, then $m - s - A(m-j) = m - s - (A(m+1-j) - 1) = m + 1 - s - A(m+1-j)$, so $A^1(m+1) = A^1(m)$ and hence $\Lambda^1(m) = A^1(m+1) - A^1(m) = 0$. On the other hand, if $\Lambda(m-j) = 0$, then $m+1-s-A(m+1-j) = m+1-s-A(m-j) = 1+(m-s-A(m-j))$, and hence $\Lambda^1(m) = A(m+1-s-A(m+1-j)) - A(m-s-A(m-j)) = A(1+m-s-A(m-j)) - A(m-s-A(m-j)) = \Lambda(m-s-A(m-j)) \in \{0, 1\}$.

Since $A^2(n) = A^1(n-j)$, then $\Lambda^2(n) = \Lambda^1(n-j)$ so the above result also holds for Λ^2. ☐

From Lemma 6.1 the following is immediate:

Lemma 6.2. *Suppose that $m > r$ is the smallest integer such that $\Lambda(m) \notin \{0, 1\}$. Then $\Lambda^1(m) = \Lambda^2(m) = 1$ and $\Lambda(m) = 2$.*

Proof. Since $A(n)$ is slow up to m, by Lemma 6.1, $\Lambda^1(m), \Lambda^2(m) \in \{0, 1\}$. Since $\Lambda(m) = \Lambda^1(m) + \Lambda^2(m) \notin \{0, 1\}$, it must be that $\Lambda^1(m) = \Lambda^2(m) = 1$ and $\Lambda(m) = 2$. ☐

In order to use the preceding lemma, we want to have a useful way to determine the value of Λ^i.

Lemma 6.3. *Assume that $A(n)$ is slow up to at least $m > r$. Then for all $p \leqslant m$, $\Lambda^1(p) = 1$ if and only if $\Lambda(p-j) = 0$ and $\Lambda(p-s-A(p-j)) = 1$, and $\Lambda^2(p) = 1$ if and only if $\Lambda(p-2j) = 0$ and $\Lambda(p-s-j-A(p-2j)) = 1$. Equivalently, $\Lambda^1(p) = 0$ if and only if $\Lambda(p-j) = 1$ or $\Lambda(p-s-A(p-j)) = 0$, and $\Lambda^2(p) = 0$ if and only if $\Lambda(p-2j) = 1$ or $\Lambda(p-s-j-A(p-2j)) = 0$.*

Proof. This result has already been proved as part of the proof of Lemma 6.1. ☐

We now prove the condition that confirms that the recursion (2.1) has a slow solution.

Theorem 6.4. *Assume that $A(n)$ is slow up to at least $n = r + j$. Then $A(n)$ is slow.*

Proof. Suppose not. Then there must be a minimum index m so that $\Lambda(m) \notin \{0, 1\}$. By Lemma 6.2, $\Lambda(m) = 2$ and $\Lambda^1(m) = \Lambda^2(m) = 1$. By Lemma 6.3, $\Lambda^1(m) = 1$ implies that $\Lambda(m-j) = 0$ and $\Lambda(m-s-A(m-j)) = 1$; in a similar way, from $\Lambda^2(m) = 1$ we conclude that $\Lambda(m-2j) = 0$ and $\Lambda(m-s-j-A(m-2j)) = 1$. Because $\Lambda(m-j) = 0$, by Lemma 6.3, $\Lambda(m-2j) = 1$ or $\Lambda(m-s-j-A(m-2j)) = 0$. But this contradicts what we just proved, hence our initial assumption must have been false. Therefore, $A(n)$ is a slow solution. ☐

7 Next Steps

We have seen that there appears to be a correspondence between certain families of meta-Fibonacci sequences defined by recursions whose parameters satisfy certain constraints and infinite trees that are closed under a natural pruning operation. In particular, shifting and multiplying the parameters of the recursion corresponds to changing the number of labels in the nodes of the tree in a way that is consistent across all three families of sequences that we have discussed.

This suggests the existence of a more general correspondence between slow-growing meta-Fibonacci sequences that arise from appropriate specifications of the parameters and labeled trees. Further work could focus directly on identifying different sets of constraints for the parameters for such other families of meta-Fibonacci recursions, together with suitable initial conditions; some initial efforts in this direction have already led to some interesting findings that will be reported in a subsequent communication. Conversely, it may be possible to reverse the process, that is, define trees closed under pruning and try to relate their label count to a meta-Fibonacci sequence. In either case, as has been shown elsewhere, it could be fruitful to consider a broader range of (possibly weighted) labeling schemes for the trees. In this vein, we do not discount the possibility that a different labeling scheme on one or more of our existing trees, together with a different set of initial conditions, could lead to a graphical combinatorial interpretation for more families of slow meta-Fibonacci sequences.

Acknowledgements

We are very grateful to Professor Frank Ruskey for helpful discussions of this material and his review of earlier versions of this paper, and to Nigel Chan for his excellent research assistance.

References

[1] R.B.J.T. Allenby and R.C. Smith, Some sequences resembling Hofstadter's, *J. Korean Math. Soc.* 40 (2003), 921-932.

[2] B. Balamohan, A. Kuznetsov and S. Tanny, On the behaviour of a variant of Hofstadter's Q-sequence, *J. Integer Sequences* 10 (2007), Article 07.7.1.

[3] B. Balamohan, Z. Li, and S. Tanny, A combinatorial interpretation for certain relatives of the Conolly sequence, *J. of Integer Sequences* 11 (2008), Article 08.2.1.

[4] M. Cai and S. Tanny, How the shift parameter affects the behavior of a family of meta-Fibonacci sequences, *J. of Integer Sequences* (2008), Article 08.3.6.

[5] J. Callaghan, J. J. Chew III and S. Tanny, On the behavior of a family of meta-Fibonacci sequences, *SIAM J. Discrete Math.* 18 (2005), 794-824.

[6] B. W. Conolly, Meta-Fibonacci sequences, in S. Vajda, ed., *Fibonacci & Lucas Numbers, and the Golden Section*, Wiley, New York, 1986, pp. 127-137.

[7] C. Deugau and F. Ruskey, Complete k-ary trees and generalized meta-Fibonacci sequences, *Fourth Colloquium on Mathematics and Computer Science: Algorithms, trees, Combinatorics and Probabilities*, DMTCS Proceedings Series, 2006 AG, pp. 203-214.

[8] S. Golomb, Discrete chaos: sequences satisfying "strange" recursions, preprint, undated.

[9] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics, Second Edition*, Addision-Wesley, Massachusetts, 1994,

[10] R.K. Guy, Unsolved problems in number theory, *Problem Books in Math*, Springer, New York, Third Edition, 2004.

[11] J. Higham and S. Tanny, More well-behaved meta-Fibonacci sequences, *Congr. Numer.*, 98 (1993), 3-17.

[12] D.R. Hofstadter, *Godel, Escher, Bach: An Eternal Golden Braid*, Basic Books, New York, 1979.

[13] D.R. Hofstadter, *Metamagical Themas*, Basic Book, NY, 1985.

[14] B. Jackson and F. Ruskey, Meta-Fibonacci sequences, binary trees and extremal compact codes, *Electron. J. of Combin.* 13 (2006), R26.

[15] K. Pinn, Order and chaos in Hofstadter's Q(n) sequence, *Complexity 4* (1999), no. 3, 41-46.

[16] F. Ruskey, private communication, August 2007.

[17] F. Ruskey, S. Chandran, A. Das, and B. Jackson, Isoperimetric sequences for infinite complete binary trees and their relation to meta-Fibonacci sequences and signed almost binary partitions, preprint, 2009.

[18] F. Ruskey and C. Deugau, The combinatorics of certain k-ary meta-Fibonacci sequences, preprint, 2009.

[19] N. J. A. Sloane, *Online Encyclopedia of Integer Sequences*, http://www.research.att.com/∼njas/sequences.

[20] S. Tanny, A well-behaved cousin of the Hofstadter sequence, *Discrete Math.* 105 (1992), 227-239.