Acta Cybernetica **24** (2019) 249–262.

Reconstruction of Rooted Directed Trees*

Dénes Bartha[a]

Abstract

Let T be a rooted directed tree on n vertices, rooted at v. The rooted subtree frequency vector (*RSTF-vector*) of T with root v, denoted by $\mathrm{rstf}(T, v)$ is a vector of length n whose entry at position k is the number of subtrees of T that contain v and have exactly k vertices. In this paper we present an algorithm for reconstructing rooted directed trees from their rooted subtree frequencies (up to isomorphism). We show that there are examples of nonisomorphic pairs of rooted directed trees that are *RSTF-equivalent*, that is they share the same rooted subtree frequency vectors. We have found all such pairs (groups) for small sizes by using exhaustive computer search. We show that infinitely many nonisomorphic *RSTF-equivalent* pairs of trees exist by constructing infinite families of examples.

Keywords: tree reconstruction, subtree size frequencies, rooted directed trees

1 Introduction

Reconstruction of certain combinatorial structures from given partial information plays an important role in several problems such as reconstructibility of strings [5, 3, 1], trees, graphs [8, 7], matrices [9, 4] etc.

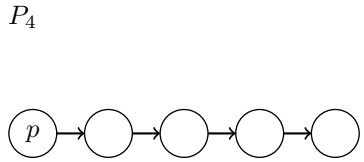
The motivation behind this paper comes from mass spectrometry data analysis. The problem we investigate is the possibility of reconstruction of an unlabeled directed rooted tree with n vertices, given the number of rooted directed subtrees frequencies of size $1, 2, \ldots, n$, which we call the *RSTF-vector*. In [2] the authors investigated the problem of reconstructibility of unlabeled free trees and defined *STF-vector* with the sum of all the *RSTF-vectors* of the subtrees of a given tree. Because there is no reconstruction algorithm of free trees given in the literature, the approach presented in this paper could be the first step towards such an algorithm.

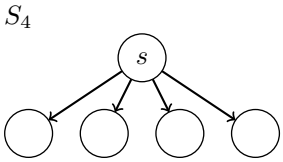
In Section 2 we give the formal definition of the *RSTF-vector*, *RSTF-polynomial*, *well formed representation* and show how to construct them from a given rooted

*This submission is for the special issue of CSCS 2018. Talent Management in Autonomous Vehicle Control Technologies – The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

[a]Eötvös Loránd University, E-mail: denesb@gmail.com

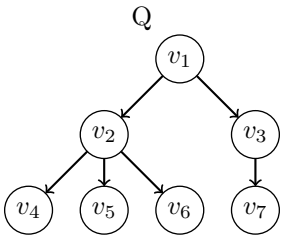
DOI: 10.14232/actacyb.24.2.2019.5





$\text{rstf}(P_4, p) = [1, 1, 1, 1, 1]$
$r(P_5, p) = 1 + x + x^2 + x^3 + x^4$

$\text{rstf}(S_4, s) = [1, 4, 6, 4, 1]$
$r(S_4, s) = 1 + 4x + 6x^2 + 4x^3 + x^4$



$\text{rstf}(Q, v_1) = [1, 2, 5, 7, 7, 4, 1]$
$r(Q, v_1) = 1 + 2x + 5x^2 + 7x^3 + 7x^4 + 4x^5 + x^6$

Figure 1: P_4 denotes a path of length 4 rooted at p, S_4 a star with 4 leaves rooted at v_2, and a more complex tree Q on 7 nodes rooted at v_1. The corresponding *RSTF-vectors* and *RSTF-polynomials* are given below the trees.

directed tree. In section 3 we introduce the algorithm *RRDT* that can reconstruct the rooted directed tree corresponding to a given polynomial. In section 4 we show some results on *RSTF-equivalent* trees. In the Conclusion section we propose new research directions.

The main problem we investigate is the method of reconstructing an unlabeled rooted directed tree from its rooted subtree frequencies [2]. The motivation of the problem comes from mass spectrometry data analysis, as the *RSTF-vector* models frequency data from mass spectrometry. Although unique reconstruction from this vector is not always possible, we still find the mathematical and algorithmic aspects of the problem worth investigating. Also, as a practical application, a molecule database search filter might be created using RSTF indexing, but this is outside the scope of the present paper and could be the topic of future research.

2 Basic definitions

In the paper x is used for the variable of univariate polynomials denoted by $f, g, \ldots$ Unless otherwise stated, polynomials have integer coefficients. The letter n usually denotes the number of nodes of an unlabeled rooted directed tree. Trees are denoted

by capital letters $P, Q, R, S, \ldots$

Definition 1. *Let $T = (V, E)$ be a rooted directed tree on $n (n \geq 1)$ vertices, rooted at vertex $v \in V$. The vector* $\mathrm{rstf}(T, v) = [r_1, \ldots, r_n]$ *is called the rooted subtree frequency vector (RSTF for short) of T with root v, where each r_i shows the number of those i-sized subtrees of T that contain v.*

We can represent *RSTF-vectors* with polynomials by choosing the entries of the vectors as the appropriate coefficients of the polynomial.

Definition 2. *Let T be a rooted directed tree, v the root of T, with* $\mathrm{rstf}(T, v) = [r_1, r_2, \ldots, r_n]$. *The* RSTF-polynomial *of T with root v, denoted by $r(T, v)$ is defined by $r(T, v) = r_1 + r_2 x + r_3 x^2 + \cdots + r_n x^{n-1}$.*

Figure 1 shows three examples on *RSTF-vectors* and *polynomials*. The reason why we use *RSTF-polynomials* instead of vectors is that we can easily calculate the *RSTF-polynomial* from a given rooted directed tree graph structure as shown in Lemma 1 [2].

Lemma 1. *Given a tree T with root v, one can calculate the RSTF-polynomial in $O(n^2)$ time using the following recursive formula:*

$$\mathrm{rstf}(T, v) = \prod_{i=1}^{k} (1 + x \cdot \mathrm{rstf}(T_i, v_i)) \,,$$

where k is the number of children of v, which are denoted by v_i, and T_i is the subtree rooted at v_i.

To illustrate the use of this lemma, we compute the *RSTF-polynomial* of tree Q given in Figure 1, applying the recursive approach step by step (Q_i denotes the subtree rooted at v_i, $i = 2, \ldots, 7$):

$$\begin{aligned}
\mathrm{rstf}(Q, v_1) &= (1 + x \cdot \mathrm{rstf}(Q_2, v_2)) \cdot (1 + x \cdot \mathrm{rstf}(Q_3, v_3)) \\
\mathrm{rstf}(Q_2, v_2) &= (1 + x \cdot \mathrm{rstf}(Q_4, v_4)) \cdot (1 + x \cdot \mathrm{rstf}(Q_5, v_5)) \cdot (1 + x \cdot \mathrm{rstf}(Q_6, v_6)) \\
\mathrm{rstf}(Q_3, v_3) &= (1 + x \cdot \mathrm{rstf}(Q, v_7)) \\
\mathrm{rstf}(Q_4, v_4) &= \mathrm{rstf}(Q_5, v_5) = \mathrm{rstf}(Q_6, v_6) = \mathrm{rstf}(Q_7, v_7) = 1
\end{aligned}$$

If we substitute back to $\mathrm{rstf}(Q, v_1)$, and expand the product, the resulted polynomial's coefficient sequence gives the *RSTF-vector* of Q:

$$\begin{aligned}
\mathrm{rstf}(Q, v_1) &= (1 + x \cdot ((1 + x \cdot 1) \cdot (1 + x \cdot 1) \cdot (1 + x \cdot 1)) \cdot (1 + x \cdot (1 + x \cdot 1))) \\
&= \mathbf{1} + \mathbf{2}x + \mathbf{5}x^2 + \mathbf{7}x^3 + \mathbf{7}x^4 + \mathbf{4}x^5 + \mathbf{1}x^6
\end{aligned}$$

A polynomial may have many different representations, but in the case of *RSTF-polynomials*, there is a specific representation from which the tree structure corresponding to the polynomial is easy to determine (as in the above form of $\mathrm{rstf}(Q, v_1)$. We therefore introduce the following formal definition.

Definition 3. *We call a representation of a polynomial f well-formed, if it is either of the form*

- $f = 1$, *or*
- $f = (1+x\cdot f_1)\cdot(1+x\cdot f_2)\cdots(1+x\cdot f_k)$ *with* $k \geq 1$ *where the* f_j $(k = 1,\ldots,k)$ *are themselves polynomials in well-formed representation.*

Every polynomial f with a well-formed representation is an *RSTF-polynomial*, and every *RSTF-polynomial* has a well-formed representation. Using the notations in the definition, the connection is that the root has k children and the subtrees rooted in these children have *RSTF-polynomials* f_j for $j = 1,\ldots,k$. Note that well-formed representations consist only of 1, +, x, $\cdot$ and () symbols, and they can be generated by a context-free grammar.

A necessary but not sufficient condition for a polynomial f to have a well-formed representation is that it can be written as $f = (1+xf_1)\cdot(1+xf_2)\cdots(1+xf_k)$ with polynomials f_j that have constant term 1 (but not necessarily RSTF-polynomials themselves). We will use this as a filter in our algorithms later. For later use, we define the concept of *RSTF-candidate factor* and *RSTF-candidate representation.*

Definition 4. *We call a polynomial RSTF-candidate factor if both the constant and linear coefficients are equal to 1. We call a representation of polynomial f RSTF-candidate representation if f is written as a product of RSTF-candidate factors, i.e. as $f = (1+xf_1)\cdot(1+xf_2)\cdots(1+xf_k)$ where all polynomials f_j have constant term* 1.

As explained later, the algorithms for finding well-formed representations (or otherwise put, corresponding trees) for a polynomial f will operate by first finding all *RSTF-candidate representations* and then recursively checking whether the polynomials f_j are RSTF-polynomials or not, and if they are, giving all their well-formed representations.

3 Methods

3.1 Reconstruction algorithm

Our main goal is to construct an algorithm that has a polynomial f as input and all trees having f as *RSTF-polynomial* as output. With the help of Lemma 1 we can construct such an algorithm. As discussed in the previous section, for finding the tree structure it is sufficient to give the well-formed representation of the polynomial.

The main idea is to factorize the input polynomial into irreducible factors and then group the factors so that this grouping yields a *well-formed* representation (using recursive calls in the process).

Let a_1 be the linear coefficient of f and let p_i denote the distinct irreducible factors of f, with exponent k_i. Then the irreducible factorization and the well-formed representation (necessarily having a_1 factors by matching the linear terms) are both equal to f:

$$\begin{aligned} f &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \\ &= p_1^{k_1} \cdot p_2^{k_2} \cdots p_m^{k_m} \\ &= (1 + x \cdot f_1) \cdot (1 + x \cdot f_2) \cdots (1 + x \cdot f_{a_1}) \end{aligned} \tag{1}$$

Each factor in the well-formed representation is the product of some irreducible factors. We call a partition of the multiset of irreducible factors a *proper grouping* of irreducible factors if the product of polynomials within the groups is always of the form $1 + xf_j$ where f_j is an RSTF-polynomial.

Once we have a well-formed representation, the tree structure can be given quickly:

Lemma 2. *From a well formed RSTF-polynomial on degree $n-1$ we can reconstruct a corresponding tree in n steps.*

Proof. Let f be a *well formed RSTF-polynomial* corresponding to a tree. Since $deg(f) \geq 0$ we can assume that the tree has at least one node that we create in advance. Then using Lemma 1 we have to count the number of outer blocks $(\ldots) \cdots (\ldots)$ and create as many new nodes (children) on the next level that we connect with the previous parent node with a directed edge (the edge points to the children). We build up the tree using this simple rule for each block recursively. Because there are exactly n pair of parentheses jumbled in f, this process takes n steps. □

The pseudocode of the *RRDT* algorithm can be seen in Algorithm 1. Figure 3.1 shows an example on how it works.

For any given polynomial the function gives back the corresponding well-formed polynomial (or polynomials) if one exists, otherwise returns $\downarrow$. First it checks the base cases. Note that the constant term and the linear coefficient must be equal to 1 (see Lemma 1). The next step is to check whether the solution can be found in the dictionary (*known*) by using dynamic programming approach (memoization). The upcoming part consists of two main cases: when the linear coefficient of the given polynomial (denoted by $f[x]$) is 1 and when it is greater than 1. Note that because of the well-formed property equation (1) it cannot be 0 or negative.

When $f[x] = 1$, we have to call the function recursively for $\frac{f-1}{x}$ because if there is a solution, then f has the form: $1 + x \cdot (\ldots)$. In this case we have to store $(1 + x \cdot result)$ in the dictionary.

Otherwise if $f[x] > 1$, we perform polynomial factorization that can be computed efficiently (polynomial time) using LLL [6] or other similar methods. The factorization gives the prime factors with the appropriate powers in the form of $f = p_1^{k_1} \cdots p_m^{k_m} \cdot q_1^{k_{m+1}} \cdots q_s^{k_{m+s}}$ (where p_i and q_j are prime factors). We can observe that among the factors there will be *RSTF-polynomials* p_i that can be represented by well-formed polynomials according to equation (1) and the remaining non-*RSTF-polynomials* q_j have different form (e.g. have a constant term different from 1).

Algorithm 1 Reconstruct well-formed RSTF-polynomial

$known \leftarrow \{\}$ ▷ dictionary for the known polynomials
procedure RRDT($f = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$)
  if $f = 1$ **then**
    return 1 ▷ Base cases
  else if $a_0 \neq 1 \vee a_n \neq 1$ **then**
    return $\downarrow$
  else if $f \in known$ **then**
    return $known[f]$
  else if $a_1 = 1$ **then** ▷ If the linear coefficient is 1
    $rp \leftarrow \text{RRDT}(\frac{f-1}{x})$;
    if $rp = \downarrow$ **then**
      $known[f] \leftarrow \downarrow$
    else
      $known[f] \leftarrow (1 + x \cdot rp)$
    end if
  else ▷ If the linear coefficient is greater than 1
    $f = r_1^{k_1} \cdots r_t^{k_t}$ ▷ Factorization step
    if $t < a_1$ **then** ▷ If there are less factors than $a_1 \rightarrow$ no solution
      $known[f] \leftarrow \downarrow$
    else
      $f = p_1^{k_1} \cdots p_m^{k_m} \cdot q_1^{k_{m+1}} \cdots q_s^{k_{m+s}}$ ▷ p_i: *RSTF-polynomials*
      ▷ q_j: non-*RSTF-polynomials*
      ▷ Find the proper groupings: $g_1, \ldots, g_{a_1}$
      if $\exists f = g_1 \cdots g_{a_1}, \forall i \in [1, a_1] : \text{RRDT}(g_i) \neq \downarrow$ **then**
        $rp \leftarrow \prod_{i=1}^{a_1} (1 + x \cdot \text{RRDT}(\frac{g_i - 1}{x}))$
        $known[f] \leftarrow rp$
      else
        $known[f] \leftarrow \downarrow$
      end if
    end if
  end if
  return $known[f]$
end procedure

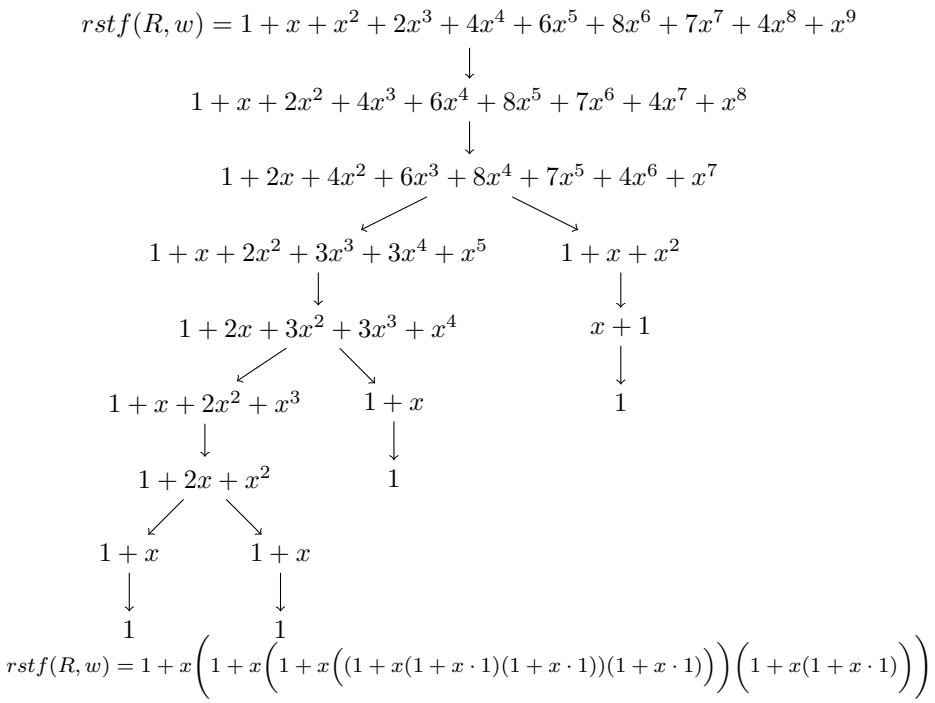


Figure 2: Given a polynomial $1 + x + x^2 + 2x^3 + 4x^4 + 6x^5 + 8x^6 + 7x^7 + 4x^8 + x^9$. Each node in the graph (except the root w) is constructed by and connected with the appropriate parent node.

We need to find proper groupings of the prime factors $f = g_1 \cdot g_2 \cdots g_{a_1}$, where each g_i is already well-formed. Note that the number of such groups is exactly $f[x] = a_1$. This step is nontrivial and needs further explanation how it can be done reasonably fast, therefore we devote the following subsection to this subroutine.

3.2 Find the proper groupings

The naïve approach is to try all the possible combinations of the prime factors and filter out the proper settings (each group represents a *rooted, directed tree*). In most cases (when we don't have many different kind of prime factors) this approach - generating all the a_1-sized partitions of a multiset - could work. But note that this is even worse than finding all the permutations of a multiset (with repetitions allowed) because we also have to distribute parentheses.

Fortunately we can give a better method to solve this problem. By the well-formed property, in each group of a proper grouping, the linear coefficient of the product must be 1. This is seen by expanding the product of all the members of the group: $1 + x + b_1x^2 + \ldots + b_{l-1}x^{l-1} + x^l$ (for some degree l).

Recall that *RSTF-candidate representations* are exactly the ones having this property of the linear term. So we will look at all *RSTF-candidate representations*, and then we have to recursively check whether the *RSTF-candidate factors* are indeed *RSTF-polynomials* or not, and if they are, function RRDT gives all their well-formed representations recursively.

A grouping of irreducible factors that gives an RSTF-candidate representation is easy to verify: we only need to sum the linear coefficients. We call a partition of a multiset of *integers* a proper integer grouping if the sum in every group is exactly 1.

Note that the prime factorization could give back non-*RSTF-polynomials* where the constant term is not equal to one (negative, 0 or greater than 1). Hence we first create a multiset of the linear coefficients of the *prime polynomials* $p_i[x]$, $q_j[x]$. We then find all proper integer groupings where the sum of each group is 1. In Algorithm 2, calling FindProperIntegerGroupings$(A, \emptyset, \emptyset)$ for some multiset of integers A will output all such proper integer groupings. Note that this uses a *DFS* approach.

Algorithm 2 Finding the proper grouping of an integer multiset

```
1: procedure FindProperIntegerGroupings(A, G, group)
2:     if ∑_{g_i ∈ group} g_i > 1 or ∑_{a_i ∈ A} a_i < 1 then
3:         return
4:     end if
5:     if ∑_{g_i ∈ group} g_i = 1 then
6:         G ← G ∪ {group}
7:         group ← ∅
8:         if A = ∅ then
9:             output G
10:        end if
11:    end if
12:    for ∀a ∈ A do
13:        FindProperIntegerGroupings(A \ {a}, G, group ∪ {a})
14:    end for
15: end procedure
```

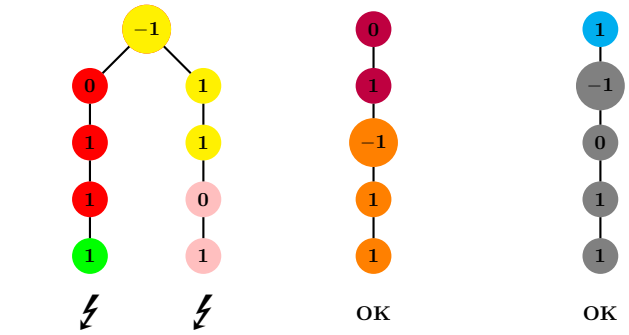
Note that FindProperIntegerGroupings might output the same partition multiple times. To avoid this we introduce the following ideas.

We define a $\preceq$ relation on the subsets (multisets) of a finite set $A \subset \mathbb{Z}$ in the following way: $\forall x, y \subseteq A$: (where x, y are multisets) $x \preceq y \iff |x| < |y|$ *or* $(|x| = |y|$ *and* $[x] \leq [y])$, where $x = \{x_1 \cdot d_1, x_2 \cdot d_2, \ldots, x_n \cdot d_n\}$, $x_1 < x_2 < \cdots < x_n$, $[x] = [\underbrace{x_1, \ldots, x_1}_{d_1}, \underbrace{x_2, \ldots, x_2}_{d_2}, \ldots \underbrace{x_n, \ldots, x_n}_{d_n}]$ here $\leq$ represents the lexicographic relation. Now we can extend function FindProperIntegerGroupings with the following things:

- In line 13 take the elements of A in *monotonically increasing order.*

- Add a new line between line 5 and 6: "$if\ \neg(y \preceq group)\ then\ return$", where y denotes the last group that we have added to G.

The following figure shows an example on how to find the groupings of the multiset $A = \{-1, 0, 1, 1, 1\}$ (the colours denote different groups).



Solutions: $\Big\{\big\{\{-1,0,1,1\},\{1\}\big\},\big\{\{0,1\},\{-1,1,1\}\big\}\Big\}$

After this step another problem arises: there could be more samples of each type of polynomials, where the type corresponds to the linear coefficient. Consider the following example:

$$1 + 2x + 4x^2 + 8x^3 + 12x^4 + 15x^5 + 16x^6 + 15x^7 + 11x^8 + 5x^9 + x^{10}$$
$$= \underbrace{(1 + \mathbf{0}\cdot\mathbf{x} + x^2 + 2x^3 + x^4)}_{f_1^{(0)}} \cdot \underbrace{(1 + \mathbf{0}\cdot\mathbf{x} + x^2 + x^3)}_{f_2^{(0)}} \cdot \underbrace{(1 + \mathbf{1}\cdot\mathbf{x} + x^2)}_{f_3^{(1)}} \cdot \underbrace{(1 + \mathbf{1}\cdot\mathbf{x})}_{f_4^{(1)}}$$

Here multiset $A' = \{0, 0, 1, 1\}$ contains the linear coefficients and FindProperIntegerGroupings $(A', \emptyset, \emptyset)$ gives the proper integer groupings:

$$\Big\{\underbrace{\big\{\{0,1\},\{0,1\}\big\}}_{g_1},\underbrace{\big\{\{1\},\{0,0,1\}\big\}}_{g_2}\Big\}$$

But there are different *RSTF-candidate factors*: $\mathbf{0}: f_1^{(0)}, f_2^{(0)}$, $\mathbf{1}: f_3^{(1)}, f_4^{(1)}$ and the question is how to combine them properly:

$$\left.\begin{aligned} g_1 &= \left(f_1^{(0)} \cdot f_3^{(1)}\right) \cdot \left(f_2^{(0)} \cdot f_4^{(1)}\right) \\ g_1 &= \left(f_1^{(0)} \cdot f_4^{(1)}\right) \cdot \left(f_2^{(0)} \cdot f_3^{(1)}\right) \end{aligned}\right\} \text{Is it valid?}$$

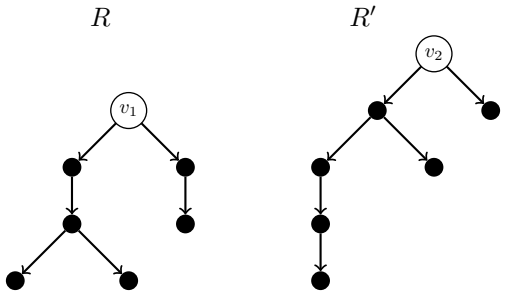


Figure 3: R and R' are two nonisomorphic rooted directed *RSTF-equivalent* trees

$$\left.\begin{aligned} g_2 &= \left(f_3^{(1)}\right)\cdot\left(f_1^{(0)}\cdot f_2^{(0)}\cdot f_4^{(1)}\right)\\ g_2 &= \left(f_4^{(1)}\right)\cdot\left(f_1^{(0)}\cdot f_2^{(0)}\cdot f_3^{(1)}\right)\end{aligned}\right\}\text{ Is it valid?}$$

Fortunately it is not so hard to get the proper settings from the possibilities if we use the above presented function *RRDT*. If one *RSTF-candidate factor* does not represent a valid *rooted directed tree*, we don't need to check the remaining factors. In this case we have to carry on and check the next possible *RSTF-candidate representation* until we find a proper solution.

Function *RRDT* reduces the degree of the polynomial by 1 when $f[x] = 1$ (or returns with a saved result). When $f[x] > 1$ the factorization step takes polynomial time. Hence the step of finding the proper grouping dominates the function where artificial examples could be given that takes exponential running time. However in practice it works fine for bigger trees on 500-1000 nodes as well and finds the solutions in a few seconds.

An implementation of the RRDT-algorithm written in sage, python can be found at `https://github.com/denesbartha/RRDT`.

4 Isomorphism and reconstructibility results

Algorithm 1 is able to reconstruct rooted directed trees up to isomorphism. There are several cases when the given polynomials determine uniquely the trees. Typical examples (see Figure 1) are P_m - path of length m (coefficients of the corresponding polynomial: $1, 1, \ldots 1$) and S_k - star with k leaves (coefficients of the corresponding polynomial: $\binom{k}{0}, \binom{k}{1}, \ldots \binom{k}{k}$) [2]. Not surprisingly there are cases when a given input polynomial represents multiple rooted directed trees. Figure 3 shows two nonisomorphic rooted directed trees that share the same *RSTF-polynomial*.

Definition 5. *We call two nonisomorphic rooted directed trees RSTF-equivalent if they share the same* RSTF-polynomial.

The following equation shows the *well formed RSTF-polynomials* of R and R' trees.

$$
\begin{aligned}
&\mathrm{rstf}(R, v_1) = \mathrm{rstf}(R', v_2) \\
&= x^6 + 3x^5 + 4x^4 + 4x^3 + 3x^2 + 2x + 1 \\
&= (x^3 + x^2 + 1) \cdot (x^2 + x + 1) \cdot (x + 1) \\
&= \Big(1 + x \cdot \Big(1 + x \cdot \Big(\big(1 + x \cdot 1\big) \cdot \big(1 + x \cdot 1\big)\Big)\Big)\Big) \cdot \Big(1 + x \cdot \big(1 + x \cdot 1\big)\Big) \\
&= \Big(1 + x \cdot \Big(\Big(1 + x \cdot \big(1 + x \cdot (1 + x \cdot 1)\big)\Big) \cdot \Big(1 + x \cdot 1\Big)\Big)\Big) \cdot (1 + x \cdot 1)
\end{aligned}
$$

Lemma 3. *Given two nonisomorphic rooted directed RSTF-equivalent trees T_1 and T_2, that share the same RSTF-polynomial f. If we add a new node respectively to both trees that we connect with the original roots, the resulted T_1', T_2' trees will remain RSTF-equivalent. Their RSTF-polynomial is $1 + x \cdot f$.*

Proof. By using Lemma 1 we can see that joining a new root node to a rooted directed tree Q with RSTF-polynomial g results in a new tree Q' that has RSTF-polynomial $1 + x \cdot g$. Simply applying this rule to the given nonisomorphic rooted directed RSTF-equivalent trees T_1 and T_2 with RSTF-polynomial f, we create two new nonisomorphic rooted directed trees T_1' and T_2' that share the RSTF-polynomial $1 + x \cdot f$ □

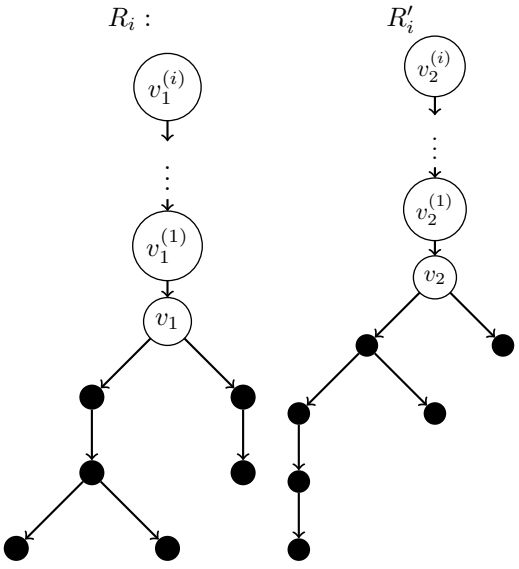


Figure 4: $\mathrm{rstf}(R_i, v_1^{(i)}) = \mathrm{rstf}(R_i', v_2^{(i)}) = 1 + x \cdot (1 + x \cdot (\ldots (1 + x \cdot \mathrm{rstf}(R, v_1)) \ldots)),\ \forall i \in \mathbb{N}^+$

Table 1: second column: $a(n)$ - the number of unlabeled rooted trees with n nodes (`https://oeis.org/A000081`); third column: the number of nonisomorphic equivalence classes; fourth column: the ratio of #equivalence classes to $a(n)$; fifth column: the maximal size equivalence class; sixth column: Shannon entropy of the equivalence classes.

n	**a(n)**	#equivalence classes	$\frac{\mathbf{ec(n)}}{\mathbf{a(n)}}$	Maximal size equivalence class	Entropy
3	2	2	1.0	1	0
4	4	4	1.0	1	0
5	9	9	1.0	1	0
6	20	20	1.0	1	0
7	48	47	0.97917	2	0.14855
8	115	112	0.97391	2	0.178
9	286	274	0.95804	2	0.25943
10	719	679	0.94437	2	0.3231
11	1842	1717	0.93214	3	0.3833
12	4766	4393	0.92174	4	0.42953
13	12486	11374	0.91094	4	0.47557
14	32973	29725	0.9015	5	0.51466
15	87811	78428	0.89315	7	0.54811
16	235381	208431	0.8855	8	0.57819
17	634847	557555	0.87825	11	0.60622
18	1721159	1499739	0.87135	11	0.63245
19	4688676	4054714	0.86479	15	0.65711
20	12826228	11011259	0.8585	16	0.68046

Theorem 1. *There are infinitely many RSTF-equivalent pairs of trees exist.*

Proof. It is enough if we find only one RSTF-equivalent pair of rooted directed trees. By joining arbitrary many new nodes to their roots respectively (Lemma 3) we always get new nonisomorphic rooted directed *RSTF-equivalent* trees. For example we can alter the given pair of directed trees rooted at v_1, v_2 in Figure 4 by joining new roots to them respectively arbitrary many times. This creates new nonisomorphic rooted directed *RSTF-equivalent* pair of trees. □

RSTF-equivalency forms an equivalence relation where the classes are the sets of nonisomorphic rooted directed trees with n nodes. Using exhaustive computer search we have found all the equivalence classes up to $n = 20$. Table 1 summarizes the results. Here we applied the Shannon entropy of the equivalence classes s.t. for a fixed tree size n that has $a(n)$ number of nonisomorphic rooted directed trees with *RSTF-equivalent* classes of $C(n) = \{c_1, \dots c_m\}$, $m \leq n$, $H_0(C(n)) = -\sum \log_2(|c_i|/m)|c_i|/m$. Up to $n = 6$, $H_0(C(n)) = 0$ which means that every

equivalence class contains only one element (in other words there are no *RSTF-equivalent* pairs). For $n > 6$ sizes the entropy rises.

Note that the maximum number of equivalence classes $ec(n)$ cannot exceed the number of nonisomorphic rooted directed trees $a(n)$, hence $\frac{ec(n)}{a(n)} \leq 1$. Also because there are infinite nonisomorphic rooted directed *RSTF-equivalent* tree classes exist (Lemma 1), $0 < \frac{ec(n)}{a(n)} < 1$, for $n \geq 7$. Our conjecture is that $lim_{n\to\infty} \frac{ec(n)}{a(n)} = 0$.

5 Conclusion and future work

In this paper we gave a concrete reconstruction algorithm *RRDT* for rooted directed trees. The main future goal is to extend these results for free trees / simple graphs that could be used in bioinformatics or spectrometry data analysis. We also plan to analyze the time complexity of the algorithm formally.

We were using only univariate polynomials and unlabeled trees. We aim to extend the above presented approach using multivariate polynomials representing labeled rooted directed trees.

Although it seems hard, in theory the factorization step of the reconstruction algorithm could be modified s.t. it would produce correct groupings of a given polynomial that represents a rooted directed tree.

I would also like to thank the anonymous referee for the valuable comments and suggestions.

References

[1] Acharya, Jayadev, Das, Hirakendu, Milenkovic, Olgica, Orlitsky, Alon, and Pan, Shengjun. String reconstruction from substring compositions. *SIAM Journal on Discrete Mathematics*, 29(3):1340–1371, 2015. DOI: 10.1137/140962486.

[2] Bartha, Dénes and Burcsi, Péter. Reconstructibility of trees from subtree size frequencies. *Studia Universitatis Babes-Bolyai, Mathematica*, 59(4):435–442, 2014.

[3] Dudik, Miroslav and Schulman, Leonard J. Reconstruction from subsequences. *Journal of Combinatorial Theory, Series A*, 103(2):337–348, 2003. DOI: 10.1016/s0097-3165(03)00103-1.

[4] Kós, Géza, Ligeti, Péter, and Sziklai, Péter. Reconstruction of matrices from submatrices. *Mathematics of Computation*, 778:1733–1747, 2009. DOI: 10.1090/s0025-5718-09-02210-8.

[5] Krasikov, I. and Roditty, Y. On a reconstruction problem for sequences. *Journal of Combinatorial Theory, Series A*, 77(2):344–348, 1997. DOI: 10.1006/jcta.1997.2732.

[6] Lenstra, A. K., Lenstra, H. W., and Lovász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. DOI: 10.1007/bf01457454.

[7] Manvel, Bennet. Reconstruction of trees. *Canadian Journal of Mathematics*, 22(1):55–60, 1970. DOI: 10.4153/CJM-1970-007-4.

[8] Manvel, Bennet. On reconstructing graphs from their sets of subgraphs. *Journal of Combinatorial Theory, Series B*, 21(2):156–165, 1976. DOI: 10.1016/0095-8956(76)90056-3.

[9] Manvel, Bennet and Stockmeyer, Paul K. On reconstruction of matrices. *Mathematics Magazine*, 44(4):218–221, 1971. DOI: 10.2307/2689082.

Received 7th September 2018