Fully Leafed Induced Subtrees

Alexandre Blondin Massé[1], Julien de Carufel[2],
Alain Goupil[2], and Élise Vandomme[1]

[1] Laboratoire de Combinatoire et d'Informatique Mathématique,
Université du Québec à Montréal, Canada
[2] Laboratoire Interdisciplinaire de Recherche en Imagerie et en Combinatoire,
Université du Québec à Trois-Rivières, Canada

March 15, 2017

Abstract. We consider the problem IS_ℓ^i of deciding whether there exists an induced subtree with i vertices and ℓ leaves in a given simple graph. We also study the associated optimization problem MLIS^i, that consists in computing the maximal number of leaves realized by an induced subtree with i vertices, in some classical families of graphs. We prove that the IS_ℓ^i problem is NP-complete in general. In addition, we exhibit a polynomial time algorithm for the MLIS^i problem in the case of trees.

Introduction

Decision and optimization problems on subtrees of graphs have been the object of many investigations in the past decades. In 1984, Payan *et al.* [7] discussed the maximum number of leaves, called the *leaf number*, that can be realized by a spanning tree of a given graph. This problem is known to be NP-complete even in the case of regular graphs of degree 4 [4], but some exact values were obtained for particular classes of graphs, as well as lower and upper bounds in other cases (see [6] and references therein for more details).

An interesting variation of this problem consists in replacing spanning trees by induced subtrees. To the best of our knowledge, the maximal number of leaves in induced subtrees has not been investigated yet, although similar problems were considered. For example, in 1986, Erdös *et al.* [5] showed that the problem $\mathrm{IS}^{>i}$ of finding an induced subtree of a given graph G with more than i vertices is NP-complete. In the data mining community, the detection of subgraph patterns, in particular of induced subtrees are used in information retrieval [10]. This requires efficient algorithms for the enumeration of induced subtrees. For instance, Wasa *et al.* [9] proposed an efficient parametrized algorithm for the generation of induced subtrees in a graph.

In this paper, we consider the following decision problem about induced subtrees and its associated optimization problem.

Problem 1 (IS_ℓ^i)*.* Given a simple graph G and two positive integers i and ℓ, does there exist an induced subtree of G with i vertices and ℓ leaves?

*Problem 2 (*MLIS^i*).* Given a simple graph G, compute the maximum number $L_G(i)$ of leaves that can be realized by an induced subtree of G with i vertices.

We prove that the IS$^i_\ell$ problem is NP-complete by reducing it to the problem IS$^{>i}$. For the MLIS^i problem, we describe a polynomial time algorithm when the graph G is a tree.

Our interest in the problems MLIS^i and IS$^i_\ell$ comes from the study of tree-like polyominoes and polycubes having a maximum number of leaves for a given number of cells. More precisely, each polyominoe (resp. polycube) has an underlying graph whose vertices are the square cells (resp. cube cells) and such that two vertices are adjacent whenever their cells share a common edge (resp. face). In [2], an explicit formula is given for the maximal number of leaves in a tree-like polyomino and polycube of given size. However, since the underlying graph of a polyomino (resp. polycube) is a subgraph of the infinite square lattice (resp. cubic lattice), this result can be rephrased as determining the maximal number of leaves in an induced subtree of $(\mathbb{Z}^d, A_d)$ with i vertices for $d = 2, 3$, where $A_d = \{\{p, p'\} \in \mathbb{Z}^d \times \mathbb{Z}^d \mid \mathrm{dist}(p, p') = 1\}$ and dist is the Euclidean distance.

This manuscript is organized as follow. In Section 1, we recall basic notions and introduce the function L_G that computes the maximal number of leaves realized by an induced subtree of G having a fixed number of vertices. We study the function L_G in classical families of graphs. In Section 2, we investigate the complexity of the decision problem IS$^i_\ell$ and show that it is NP-complete in general. In Section 3, we exhibit a polynomial algorithm to compute L_G when G is a tree so that the problem MLIS^i is polynomial in the case of trees. We conclude with some perspectives on future work in Section 4.

1 Preliminaries

All graphs considered in this text are simple and undirected unless stated otherwise. Let $G = (V, E)$ be a graph and $U \subseteq V$. We denote by $G[U]$ the subgraph of G induced by the vertex set U. Given two vertices u and v, we denote by $d(u, v)$ the *distance* between u and v, that is the number of edges in a shortest path between u and v. The *degree* of a vertex u is the number of vertices that are at distance 1 from u. We denote by $n_i(G)$ the number of vertices of degree i in G and by $n(G) = |V|$ the total number of vertices of G which is called the *size* of G. In particular, when G is a tree, then $n_1(G)$ is its number of leaves.

Definition 3. *Given a finite or infinite graph $G = (V, E)$, we denote by $\mathcal{T}_G(i)$ the family of all induced subtrees of G with exactly i vertices. The function $L_G(i)$ maps the integers $i \in \{2, \ldots, n(G)\}$ to the maximal number of leaves that can be realized by a tree in $\mathcal{T}_G(i)$, i.e.*

$$L_G(i) = \max\{n_1(T) \mid T \in \mathcal{T}_G(i)\}.$$

As is customary, we set $\max \emptyset = -\infty$*. An induced subtree T of G with i vertices is called* fully leafed *when $n_1(T) = L_G(i)$.*

The following observation is immediate.

Proposition 4. *In any graph G with at least two vertices, $L_G(2) = 2$. Moreover, if G contains at least three vertices and is non-isomorphic to K_n, the complete graph on n vertices, then $L_G(3) = 2$.*

Another useful fact is the following.

Proposition 5. *For any simple graph G, the sequence $(L_G(i))_{i=2,3,\ldots,n(G)}$ is non-decreasing if and only if G is a tree.*

Proof. If G is a tree, then $L_G(i)$ cannot be decreasing because if a subtree T_1 of G contains a subtree T_2 then $n_1(T_1) \geq n_1(T_2)$. If G is not a tree, then either it contains a cycle or it is not connected. In both cases, G has no subtree with $n(G)$ vertices. Therefore $L_G(n(G)) = -\infty$ and $L_G(2) = 2$ which implies that there exists a decreasing step in the sequence $L_G(i)$. □

We end this section by computing the function $L_G(i)$ for well known families of graphs. In the case of finite graphs, Table 1 presents the results for the *complete graph* K_n with n vertices, the *cyclic graph* $\mathcal{C}_n$ with n vertices, the *wheel* W_n with $n + 1$ vertices and the *bipartite complete graph* $K_{p,q}$ with $p + q$ vertices. Proofs are omitted as they are straightforward.

$L_{K_n}(i) = \begin{cases} 2 & \text{if } i = 2, \\ -\infty & \text{if } 3 \leq i \leq n. \end{cases}$	$L_{K_{p,q}}(i) = \begin{cases} 2 & \text{if } i = 2, \\ i-1 & \text{if } 3 \leq i \leq \max(p,q)+1, \\ -\infty & \text{if } \max(p,q)+1 < i \leq p+q. \end{cases}$
$L_{\mathcal{C}_n}(i) = \begin{cases} 2 & \text{if } 2 \leq i < n, \\ -\infty & \text{if } n \leq i \leq n. \end{cases}$	$L_{W_n}(i) = \begin{cases} 2 & \text{if } 2 = i \text{ or } \lfloor \frac{n}{2} \rfloor + 1 < i < n, \\ i-1 & \text{if } 3 \leq i \leq \lfloor \frac{n}{2} \rfloor + 1, \\ -\infty & \text{if } n \leq i \leq n+1. \end{cases}$
$L_{\mathcal{Q}_3}(i) = \begin{cases} 2 & \text{if } i \in \{2,3,5\}, \\ 3 & \text{if } i = 4, \\ -\infty & \text{if } 6 \leq i \leq 2^3. \end{cases}$	$L_{\mathcal{Q}_4}(i) = \begin{cases} 2 & \text{if } i \in \{2,3\}, \\ 3 & \text{if } i \in \{4,6,8\}, \\ 4 & \text{if } i \in \{5,7,9\}, \\ -\infty & \text{if } 10 \leq i \leq 2^4. \end{cases}$

Table 1. Function L_G for some classical families of finite graphs

Explicit formulas can also be obtained for some infinite graphs. For all $i \geq 3$,

1. $L_{\mathcal{P}_\infty}(i) = 2$ in the infinite path $\mathcal{P}_\infty$,
2. $L_{\mathcal{B}_\infty}(i) = \lfloor \frac{i}{2} \rfloor + 1$ in the infinite binary tree $\mathcal{B}_\infty$,
3. $L_{\mathcal{S}_\infty}(i) = i - 1$ in the infinite star $\mathcal{S}_\infty$.

The infinite square lattice $\mathbb{Z}^2$. Blondin Massé *et al.* [2] determined the maximal number of leaves $L_{\mathbb{Z}^2}(i)$ in induced subgraphs with i vertices of the square lattice. The function $L_{\mathbb{Z}^2}$ satisfies the linear recurrence

$$L_{\mathbb{Z}^2}(i) = \begin{cases} 2 & \text{if } i = 2, \\ i - 1 & \text{if } i = 3, 4, 5, \\ L_{\mathbb{Z}^2}(i-4) + 2 & \text{if } i \geq 6. \end{cases}$$

Hence, the asymptotic growth of $L_{\mathbb{Z}^2}(i)$ is $i/2$.

The infinite cubic lattice $\mathbb{Z}^3$. The authors of [2] also gave the maximal number of leaves $L_{\mathbb{Z}^3}(i)$ in induced subgraphs of the discrete space $\mathbb{Z}^3$ with i vertices. It also satisfies a linear recurrence

$$L_{\mathbb{Z}^3}(i) = \begin{cases} f(i) + 1 & \text{if } i = 6, 7, 13, 19, 25, \\ f(i) & \text{if } 2 \leq i \leq 40 \text{ and } i \neq 6, 7, 13, 19, 25, \\ f(i-41) + 28 & \text{if } 41 \leq i \leq 84, \\ L_{\mathbb{Z}^3}(i-41) + 28 & \text{if } i \geq 85, \end{cases}$$

where f is the function defined by

$$f(i) = \begin{cases} \lfloor (2i+2)/3 \rfloor & \text{if } 0 \leq i \leq 11, \\ \lfloor (2i+3)/3 \rfloor & \text{if } 12 \leq i \leq 27, \\ \lfloor (2i+4)/3 \rfloor & \text{if } 28 \leq i \leq 40. \end{cases}$$

so that the asymptotic growth of $L_{\mathbb{Z}^3}(i)$ is $28i/41$.

2 Complexity of the Problem

In this section, we study the complexity of the decision problem IS_ℓ^i.

Theorem 6. *The problem* IS_ℓ^i *is NP-complete.*

Proof. It is clear that IS_ℓ^i is in NP. To show that it is NP-complete, we reduce it to the problem INDUCED SUBTREE ($\text{IS}^{>i}$): *Given a graph G and a positive integer i, does there exist an induced subtree of G with strictly more than i vertices?* The problem $\text{IS}^{>i}$ was shown NP-complete by Erdös *et al.* in 1986 [5].

Consider the map f that associates to an instance (G, i) of $\text{IS}^{>i}$ with $G = (V, E)$ the instance $(H, 2(i+1), i+1)$ of IS_ℓ^i such that the graph H is obtained as a copy of G and an additional copy V' of V with an edge between $v \in V$ and its corresponding vertex $v' \in V'$ for each $v \in V$. Clearly, this map is polynomial as the graph obtained has $2|V|$ vertices and $|E| + |V|$ edges. See Figure 1 for an example.

If (G, i) is a positive instance of $\text{IS}^{>i}$, i.e. an instance for which the answer is yes, then $f(G, i) = (H, 2(i+1), i+1)$ is a positive instance of IS_ℓ^i. Indeed, assume that the graph G has an induced subtree with strictly more than i vertices. Then it has in particular an induced subtree T with $i+1$ vertices. So, in H, the set containing the vertices of T and the $i+1$ corresponding vertices of V' induces a subtree with $2(i+1)$ vertices and exactly $i+1$ leaves.

Conversely, if the instance $f(G,i) = (H, 2(i+1), i+1)$ is a positive instance of IS_ℓ^i, then (G,i) is a positive instance of $\mathrm{IS}^{>i}$. Indeed, assume that H contains an induced subtree T with $2(i+1)$ vertices and $i+1$ leaves. Observe that, in H, any set of k vertices of V and j vertices of V' induces a non-connected subgraph if $k < j$. Hence, the induced subtree T has at most $i+1$ vertices in V'. In other words, T has at least $i+1$ vertices in V. So the vertices of T that belong to V induce a subtree in G that has at least $i+1$ vertices.

Therefore, $\mathrm{IS}^{>i} \leq \mathrm{IS}_\ell^i$ and IS_ℓ^i is NP-complete. $\square$

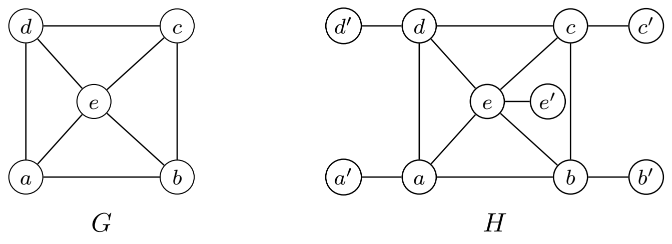

Fig. 1. Illustration of the polynomial transformation $f(G,i) = (H, 2(i+1), i+1)$

3 Trees

Hereafter, we describe in details an algorithm with polynomial time complexity based on the dynamic programming paradigm.

Before going further, we need additional definitions. A *rooted tree* $\widehat{T} = (T,u)$ is a couple where $T = (V,E)$ is a tree and $u \in V$ is a distinguished vertex called the *root* of $\widehat{T}$. Rooted graphs have a natural orientation with arcs pointing away from the root. A *leaf* of a rooted tree is therefore a vertex v with outdegree $\deg^+(v) = 0$. In particular, if the rooted tree consists of a single vertex, then this vertex is a leaf. The functions n and n_1 are defined accordingly by

$$n(\widehat{T}) = n(T) \text{ and } n_1(\widehat{T}) = \mathrm{Card}\left(\{v \in \widehat{T} \mid \deg^+(v) = 0\}\right).$$

Similarly, a *rooted forest* $\widehat{F}$ is a collection of rooted trees. It follows naturally that

$$n_1(\widehat{F}) = \sum_{\widehat{T} \in \widehat{F}} n_1(\widehat{T}).$$

Let $\widehat{T}$ be any rooted tree with n vertices and $L_{\widehat{T}} : \{0,1,\ldots,n\} \to \mathbb{N}$ be defined by

$$L_{\widehat{T}}(i) = \max\{n_1(\widehat{T'}) : \widehat{T'} \preceq \widehat{T} \text{ and } n(\widehat{T'}) = i\}, \tag{1}$$

where $\preceq$ denotes the relation "being a rooted subtree with the same root". Roughly speaking, $L_{\widehat{T}}(i)$ denotes the maximum number of leaves that can be realized by some rooted subtree of size i of $\widehat{T}$. This map is naturally extended to rooted forests. Let $\widehat{F} = \{\widehat{F_1}, \dots, \widehat{F_k}\}$ be a rooted forest and set

$$L_{\widehat{F}}(i) = \max \left\{ \sum_{j=1}^{k} n_1(\widehat{T'_j}) : \widehat{T'_j} \preceq \widehat{F_j} \text{ and } \sum_{j=1}^{k} n(\widehat{T'_j}) = i \right\}. \tag{2}$$

Let $C(i,k)$ be the set of all weak compositions $\lambda = (\lambda_1, \dots, \lambda_k)$ of i in exactly k non-negative parts. Then Equation (2) is equivalent to

$$L_{\widehat{F}}(i) = \max \left\{ \sum_{j=1}^{k} L_{\widehat{F_j}}(\lambda_j) \;\middle|\; \lambda \in C(i,k) \right\}. \tag{3}$$

Assuming that $L_{\widehat{F_j}}$ is known for $j = 1, 2, \dots, k$, a naive computation of $L_{\widehat{F}}$ using Equation (3) is not done in polynomial time, since

$$|C(i,k)| = \binom{i+k-1}{i}.$$

However, the next lemma ensures that this can be done in polynomial time.

Lemma 7. *Let $k \geq 1$ be an integer and $\widehat{F} = \{\widehat{F_1}, \dots, \widehat{F_k}\}$ be a rooted forest with n vertices. Then, for $i \in \{0, \dots, n\}$,*

$$L_{\widehat{F}}(i) = \begin{cases} L_{\widehat{F_1}}(i) & \text{if } k = 1, \\ \max\{L_{\widehat{F'}}(j) + L_{\widehat{F_1}}(i-j) \mid 0 \leq j \leq i\} & \text{if } k \geq 2, \end{cases} \tag{4}$$

where $\widehat{F'} = \{\widehat{F_2}, \dots, \widehat{F_k}\}$. Therefore, if $L_{\widehat{F_j}}$ is known for $j = 1, 2, \dots, k$, then $L_{\widehat{F}}$ can be computed in $\mathcal{O}(kn^2)$ time.

Proof. The first part follows from Equation (3) and the fact that, for $k \geq 2$, we have

$$C(i,k) = \{(j, \lambda_1, \lambda_2, \dots, \lambda_{k-1}) \mid 0 \leq j \leq i, \lambda \in C(i-j, k-1)\}.$$

For the time complexity, one notices that for a given i, the recursive step of Equation (4) is applied $k-1$ times, where each step is done in $\mathcal{O}(n)$. Since $L_{\widehat{F}}(i)$ is computed for $i = 1, 2, \dots, n$, the total time complexity is $\mathcal{O}(kn^2)$. □

Finally, we describe how one computes $L_{\widehat{T}}$ from its children.

Lemma 8. *Let $\widehat{T}$ be some rooted tree with root u. Let $\widehat{F}$ be the rooted forest induced by the children of u. Then*

$$L_{\widehat{T}}(i) = \begin{cases} i & \text{if } i = 0, 1, \\ L_{\widehat{F}}(i-1) & \text{if } 2 \leq i \leq n(\widehat{T}). \end{cases} \tag{5}$$

Proof. The cases $i = 0, 1$ are immediate. Assume that $i \geq 2$. Since any rooted subtree of $\widehat{T}$ must in particular include the root u and since u is not a leaf, all the leaves are in $\widehat{F}$ and the result follows. $\square$

Combining Lemmas 7 and 8, we obtain the following result.

Theorem 9. *Let $T = (V, E)$ be a undirected tree with $n \geq 2$ vertices. Then L_T can be computed in $\mathcal{O}(n^3 \Delta)$ time and $\mathcal{O}(n^2)$ space where Δ denotes the maximal degree of a vertex in T.*

Proof. For any edge $\{u, v\} \in E$ of T, if it is removed from T, then we obtain two rooted trees: a tree $\widehat{T}(v \to u)$ rooted in u and a tree $\widehat{T}(u \to v)$ rooted in v. Using both Lemmas 7 and 8, we compute the values of $L_{\widehat{T}(u \to v)}$ and $L_{\widehat{T}(v \to u)}$ for each edge $\{u, v\}$ by storing the intermediate values to avoid redundant computations. The overall time complexity is

$$\sum_{\{u,v\} \in E} \left(\mathcal{O}(\deg(u) n^2) + \mathcal{O}(\deg(v) n^2) \right) = \mathcal{O}(n^3 \Delta),$$

by Lemma 7 and the fact that $|E| = n - 1$.

Next, let the function $L_{\{u,v\}} : \{2, 3, \ldots, n\} \to \mathbb{N}$ be defined by

$$L_{\{u,v\}}(i) = \max \left\{ L_{\widehat{T}(u \to v)}(j) + L_{\widehat{T}(v \to u)}(i - j) \;\middle|\; 1 \leq j \leq i - 1 \right\},$$

i.e. $L_{\{u,v\}}(i)$ is the maximum number of leaves that can be realized by all subtrees of T containing the edge $\{u, v\}$ and having i vertices. Clearly, $L_{\{u,v\}}$ is computed in time $\Theta(n)$ when the functions $L_{\widehat{T}(u \to v)}$ and $L_{\widehat{T}(v \to u)}$ have been computed. Hence, since any optimal subtree with $i \geq 2$ vertices has at least one edge, the optimal value $L_T(i)$ must be stored in at least one edge, so that

$$L_T(i) = \max \left\{ L_{\{u,v\}}(i) \;\middle|\; \{u, v\} \in E \right\},$$

which is computed in $\Theta(n)$ time as well. The global time complexity is therefore $\mathcal{O}(n^3 \Delta)$, as claimed. Finally, the space complexity of $\mathcal{O}(n^2)$ follows from the fact that each of the $n - 1$ edges stores information of size $\mathcal{O}(n)$. $\square$

Example 10. Consider the tree depicted in Figure 2 without the orientation and with only one edge between u and v.

By Theorem 9, the computation of $L_T(i)$ with $i \in \{2, \ldots, 14\}$ requires to first compute the function $L_{\{x,y\}}$ for each edge $\{x, y\} \in E$. Figure 2 illustrates the computation of $L_{\{u,v\}}$ for the specific edge $\{u, v\}$. Indeed, the blue arc (v, u) stores the value of $L_{\widehat{T}(v \to u)}$. As $L_{\widehat{T}(v \to u)}$ is computed recursively on the subtrees rooted in the children, the other blue arcs hold intermediate computation necessary for $L_{\widehat{T}(v \to u)}$. Similarly, the red edges hold the intermediate values of the recursive computation of $L_{\widehat{T}(u \to v)}$.

We now describe in detail the computation of the function $L_{\widehat{T}(v \to u)}$. Using the recursion, we first consider the leaves $\{u_1, u_2, u_3, u_4, u_5, u_7\}$ of the blue rooted

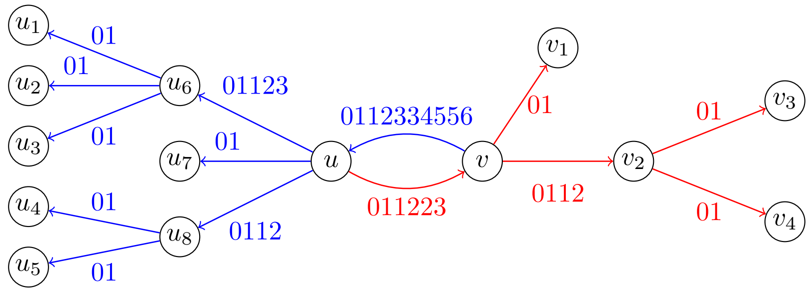

Fig. 2. Intermediate computations of the function $L_{\{u,v\}}$ for the edge $\{u,v\}$

subtree $\widehat{T}(v \to u)$. The function $L_{\widehat{T}(u_6 \to u_1)}$ has domain $\{0,1\}$. By the first case of Equation (5), we have $L_{\widehat{T}(u_6\to u_1)}(0) = 0$ and $L_{\widehat{T}(u_6\to u_1)}(1) = 1$. This information is represented In Figure 2 by the word 01 on the arc (u_6, u_1). This computation is similar for all blue leaves and the values of their respective functions are stored in their incident arc. Consider now the rooted forest $\widehat{F} = \{\widehat{u_1}, \widehat{u_2}, \widehat{u_3}\}$ consisting of three rooted trees of size 1. Using the notation of Lemma 7, we have for $j = 1, 2, 3$,

$$L_{\widehat{u_j}}(i)_{i\in\{0,1\}} = L_{\widehat{T}(u_6\to u_j)}(i)_{i\in\{0,1\}} = (0,1)$$

and we compute the function $L_{\widehat{F'}}$ of domain $\{0,1,2\}$ where $\widehat{F'} = \{\widehat{u_2}, \widehat{u_3}\}$:

- $L_{\widehat{F'}}(0) = \max\{L_{\widehat{u_3}}(0) + L_{\widehat{u_2}}(0)\} = 0$,
- $L_{\widehat{F'}}(1) = \max\{L_{\widehat{u_3}}(0) + L_{\widehat{u_2}}(1), L_{\widehat{u_3}}(1) + L_{\widehat{u_2}}(0)\} = 1$,
- $L_{\widehat{F'}}(2) = \max\{L_{\widehat{u_3}}(1) + L_{\widehat{u_2}}(1)\} = 2$.

Thus we obtain the following values for the function $L_{\widehat{F}}$ of domain $\{0,1,2,3\}$:

- $L_{\widehat{F}}(0) = \max\{L_{\widehat{F'}}(0) + L_{\widehat{u_1}}(0)\} = 0$,
- $L_{\widehat{F}}(1) = \max\{L_{\widehat{F'}}(0) + L_{\widehat{u_1}}(1), L_{\widehat{F'}}(1) + L_{\widehat{u_1}}(0)\} = 1$,
- $L_{\widehat{F}}(2) = \max\{L_{\widehat{F'}}(1) + L_{\widehat{u_1}}(1), L_{\widehat{F'}}(2) + L_{\widehat{u_1}}(0)\} = 2$,
- $L_{\widehat{F}}(3) = \max\{L_{\widehat{F'}}(2) + L_{\widehat{u_1}}(1)\} = 3$.

By Lemma 8, we are then able to compute $L_{\widehat{T}(u\to u_6)}$ as $\widehat{F} = \{\widehat{u_1}, \widehat{u_2}, \widehat{u_3}\}$ is the rooted forest induced by the children of u_6:

i	0	1	2	3	4
$L_{\widehat{T}(u\to u_6)}(i)$	0	1	$L_{\widehat{F}}(1) = 1$	$L_{\widehat{F}}(2) = 2$	$L_{\widehat{F}}(3) = 3$

Similar computations give

i	0	1	2	3	4	5	6	7	8	9
$L_{\widehat{T}(u\to u_7)}(i)$	0	1								
$L_{\widehat{T}(u\to u_8)}(i)$	0	1	1	2						
$L_{\widehat{T}(v\to u)}(i)$	0	1	1	2	3	3	4	5	5	6
$L_{\widehat{T}(u\to v)}(i)$	0	1	1	2	2	3				

Finally, when the values $L_{\widehat{T}(v\to u)}(i)_{i\in\{0,\ldots,9\}}$ and $L_{\widehat{T}(u\to v)}(i)_{i\in\{0,\ldots,5\}}$ are known, we deduce easily the values of $L_{\{u,v\}}$:

$$L_{\{u,v\}}(i)_{i\in\{2,\ldots,14\}} = (2,2,3,4,4,5,6,6,7,7,8,8,9).$$

4 Concluding Remarks

It is not clear whether the algorithm described in Theorem 9 is optimal or its time complexity analysis could be refined. However, it is worth mentioning that a procedure based on the successive deletion of leaves is not a viable strategy. Indeed, consider the tree T represented in Figure 3. We have $L_T(9) = 6$ and $L_T(7) = 5$ and there is exactly one fully leafed induced subtree with respectively 7 and 9 vertices. But the smallest of these two subtrees (in blue) is not a subgraph of the largest one (in red).

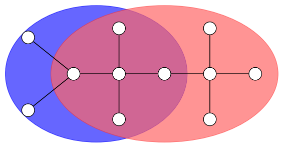

Fig. 3. A tree with its two unique fully leafed induced subtrees with 7 and 9 vertices

References

1. P. Andrews. An introduction to mathematical logic and type theory: to truth through proof. *Applied Logic Series*, 27 (Second edition). Kluwer Academic Publishers, Dordrecht, 2002.
2. A. Blondin Massé, J. de Carufel, A. Goupil, and M. Samson. Fully Leafed Tree-Like Polyominoes and Polycubes. Preprint.
3. B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85 (1): 12–75, 1990.
4. M. R. Garey, and D. S. Johnson. Computers and Intractability, A guide to the theory of NP-Completeness. *Freeman, San Francisco*, 1979.
5. P. Erdös, M. Saks, and V. T. Sós. Maximum induced trees in graphs. *J. Combin. Theory Ser. B*, 41 (1): 61–79, 1986.
6. D. J. Kleitman, and D. B. West. Spanning trees with many leaves. *SIAM J. Discrete math*, 4 (1): 99–106, 1991.
7. C. Payan, M. Tchuente, and N.H.Xuong. Arbres avec un nombre maximum de sommets pendants. *Discrete Mathematics*, 49 (3), 267–273, 1984.
8. N. Robertson, and P. Seymour. Graph minors II: algorithmic aspects of tree-width. *J. Algorithms*, 7: 308–322, 1986

9. K. Wasa, H. Arimura, and T. Uno. Efficient Enumeration of Induced Subtrees in a K-Degenerate Graph. In: Ahn HK., Shin CS. (eds) Algorithms and Computation. ISAAC 2014. Lecture Notes in Computer Science, vol 8889: 94–102. Springer, Cham, 2014.
10. M. J. Zaki. Efficiently Mining Frequent Trees in a Forest. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton Alberta, 71–80, 2002.