CS 412: Algorithm Analysis

Department of Computer Science
University of Regina
Fall 2020
Instructor: Dr. Boting Yang

Project

Due date: December 14, 2020 at 11:59pm (Regina time)

1 Introduction

This project is for CS412 students. In this project, each student must write a program to implement Algorithm 1 in Section 3. Definitions and notations are given in Section 2 which may help you to understand the algorithm. The submission requirements are as follows.

1. You are NOT allowed to work in groups on the project. You must maintain the confidentiality of your project; do not provide any opportunity for others to copy any of your work.
2. Any close resemblances in the submitted code will be assumed to be the result of cheating. Copying of other people's projects is plagiarism. Allowing your project to be copied will be treated the same as copying. THE CONSEQUENCE OF PLAGIARISM OR ANY OTHER FORM OF CHEATING MAY RANGE FROM A ZERO GRADE, TO FAILURE IN THE CLASS, TO EXPULSION FROM THE UNIVERSITY. Please note that the dean of the faculty will be informed of any such incident, as per university regulations. Refer to the section on Academic Misconduct and Penalties in the General University Calendar.
3. Your submission has to include at least
   - source files,
   - output screenshots of all the examples in Section 4,

- the results of the numerical experiments in Section 5,
- external documentation (readme.pdf), in which you provide to the user about how to use your program and your general idea.
- internal documentation which is a file for someone to read your code.

4. You can use C++, Java or Python programming language.
5. The project has to be submitted on UR Courses before the deadline.
6. All your files need to be zipped to one .zip file. The name of the zip file is in this format: "assignment#-YourName-YourStudentID.zip".
7. For any questions regarding the assignments and the project, please send your inquiry to the marker Ms. Tanzina Akter (tanzinacuet@gmail.com). Your subject should be something like "CS412 - Your Name - Your Student ID - subject".

2 Definitions and notation

Let G be a graph. The vertex set of G is denoted by $V(G)$. We use $u_1 \cdots u_m$ to denote a path with end vertices u_1 and u_m. The *length of a path* is the number of edges on the path. The *distance* between u and v, denoted by $\mathrm{dist}_G(u,v)$, is the length of the shortest path between u and v in G. Let H be a subgraph of G. The distance between u and H is defined to be $\mathrm{dist}_G(u,H) \coloneqq \min\{\mathrm{dist}_G(u,v) \mid v \in V(H)\}$. The *neighbourhood* of v is the set $N_G(v) \coloneqq \{u \in V(G) \mid \mathrm{dist}_G(u,v) = 1\}$. The *closed neighbourhood* of v is the set $N_G[v] \coloneqq \{u \in V(G) \mid \mathrm{dist}_G(u,v) \leq 1\}$. For $k \geq 0$, we generalize this concept to the *k-th closed neighbourhood* of v, which is the set

$$N_G^k[v] \coloneqq \{u \in V(G) \mid \mathrm{dist}_G(u,v) \leq k\}.$$

The *closed neighbourhood* of $U \subseteq V(G)$ is defined as the set $N_G[U] = \{u \in V(G) \mid \mathrm{dist}_G(u,U) \leq 1\}$.

The *degree* of v is the number of edges incident on v, denoted $\deg_G(v)$. A *leaf* is a vertex that has degree one. For $U \subseteq V(G)$, we use $G[U]$ to denote the subgraph induced by U, which consists of all vertices of U and all of the edges that connect vertices of U in G. We use $G - U$ to denote the subgraph

$G[V(G) - U]$. If U contains a single vertex u, then for simplicity, we use $G - u$ for $G - \{u\}$.

A rooted tree is a tree where a single vertex is marked as the root. Let $T^{[r]}$ denote a rooted tree T with root r. Every vertex $v \neq r$ of the tree is connected with root r by a unique path where the *parent* of v is the sole neighbour of v in the unique path. If u is the parent of v, then v is a *child* of u. For a vertex $v \in V(T^{[r]})$, if a vertex u is on the unique path from r to v, then we say that v is a *descendant* of u, and u is an *ancestor* of v. For a vertex $v \in V(T^{[r]})$, we will use $T^{[v]}$ to denote the subtree of $T^{[r]}$ induced by v and all its descendants, where v is the root of this subtree. We will extensively use the notation of $T^{[v]} - v$ to denote the forest induced by $V(T^{[v]}) - \{v\}$, where $T^{[v]}$ is a rooted subtree of $T^{[r]}$. Note that each component in the forest $T^{[v]} - v$ is rooted at the vertex that is a child of v in $T^{[v]}$.

The one-visibility cops and robber game is played on a graph by two players: cop player and robber player. The cop player controls a set of cops and the robber player controls a single robber. The robber has full information about the locations of all cops, but the cops have the information about the location of the robber only when there is a cop whose distance to the robber is at most one. The game is played over a sequence of rounds. Each *round* consists of a cops' turn followed by a robber's turn. At round 0, the cops are placed on a set of vertices and then the robber is placed on a vertex. At each of the following rounds, the cops move first and the robber move next. At round i, $i \geq 1$, each cop either moves from the current vertex to a neighbouring vertex or stays still, then the robber does the same. The cops *see* the robber if the closed neighbourhood of the cops contains the robber. The cops *capture* the robber if one of them occupies the same vertex as the robber. If this happens in a finite number of rounds, then the *cops win*; otherwise, the *robber wins*. The *one-visibility cop number* of a graph G, denoted by $\mathsf{c}_1(G)$, is the minimum number of cops required to capture the robber on G.

If G is not connected, from the above definition, we know that $\mathsf{c}_1(G)$ is the sum of the one-visibility cop number of each component of G. We define $c_1^*(G)$ to be the largest possible $\mathsf{c}_1(G')$, where G' is a component in G. $c_1^*(G) = \max\{\mathsf{c}_1(G') \mid G' \text{ is a component in } G\}$.

We say that a cop *vibrates* between two adjacent vertices x and y for a consecutive sequence of rounds if in these rounds, the cop alternates two actions: "sliding from x to y" and "sliding from y to x". We say a subgraph is *cleared* at some moment if it is certain that this subgraph does not contain the robber at the moment; otherwise, the subgraph is *dirty* or *contaminated*.

For simplicity, we will use $T^{[v]} - N^3[v]$ for $T^{[v]} - N^3_{T^{[v]}}[v]$, which is the forest obtained from the rooted tree $T^{[v]}$ by deleting the vertices of $N^3_{T^{[v]}}[v]$. Similarly, if there is no ambiguity we will simply use $\mathrm{dist}(u,v)$, $N[v]$, $N^2[v]$ and $N^3[v]$ without subscripts.

Definition 2.1. (k-pre-branching, k-weakly-branching, k-branching) Let $T^{[v]}$ be a rooted tree with $\mathsf{c}_1(T^{[v]}) = k \geq 1$. We call v a *k-pre-branching vertex* if $c_1^*(T^{[v]} - N^2[v]) = k$ and $\mathsf{c}_1(T^{[u]}_{2v}) = k$, where $T^{[u]}_{2v}$ is a tree obtained from two copies of $T^{[v]}$ by connecting each root v to a new root u.

We call v a *k-weakly-branching vertex* if one of the three forests, $T^{[v]} - v$, or $T^{[v]} - N[v]$, or $T^{[v]} - N^2[v]$, has exactly two components whose root is a k-pre-branching vertex in the component.

We call v a *k-branching vertex* if $c_1^*(T^{[v]} - N^2[v]) = k$, the forest $T^{[v]} - N^2[v]$ has exactly one component whose root is a k-weakly-branching vertex in the component, and the forest $T^{[v]} - N^3[v]$ has no component whose root is a k-weakly-branching vertex.

Let u be a child of v in $T^{[v]}$. If u is a k-pre-branching vertex (resp. k-weakly-branching vertex, k-branching vertex) in $T^{[u]}$, then we say that u is a *k-pre-branching child* (resp. *k-weakly-branching child*, *k-branching child*) of v. Similarly, we can define the *k-pre-branching descendant*, *k-weakly-branching descendant*, and *k-branching descendant* of v.

Definition 2.2. Let $T^{[v]}$ be a rooted tree with $\mathsf{c}_1(T^{[v]}) = k \geq 1$. The *$k$-pre-branching indicator* $I^k_{\mathrm{pb}}(v)$ and the *k-weakly-branching indicator* $I^k_{\mathrm{wb}}(v)$ are defined to be:

$$I^k_{\mathrm{pb}}(v) = \begin{cases} 1, & \text{if } v \text{ is a } k\text{-pre-branching vertex in } T^{[v]}; \\ 0, & \text{otherwise.} \end{cases}$$

$$I^k_{\mathrm{wb}}(v) = \begin{cases} 1, & \text{if } v \text{ is a } k\text{-weakly-branching vertex in } T^{[v]}; \\ 0, & \text{otherwise.} \end{cases}$$

Definition 2.3. Let $T^{[v]}$ be a rooted tree with $\mathsf{c}_1(T^{[v]}) = k \geq 1$. The *$k$-initial-counter* $J^k(v)$ and the *k-weakly-counter* $J^k_{\mathrm{w}}(v)$ are defined as follows:

$$J^k(v) = \begin{cases} 0, & I^k_{\mathrm{pb}}(v) = 0 \text{ and } c_1^*(T^{[v]} - v) = k-1; \\ 1, & I^k_{\mathrm{pb}}(v) = 0,\ c_1^*(T^{[v]} - N[v]) = k-1 \text{ and } c_1^*(T^{[v]} - v) = k; \\ 2, & I^k_{\mathrm{pb}}(v) = 0,\ c_1^*(T^{[v]} - N^2[v]) = k-1 \text{ and } c_1^*(T^{[v]} - N[v]) = k; \\ 0, & \text{otherwise.} \end{cases}$$

$$J_{\mathrm{w}}^{k}(v)=\begin{cases}0, & \text{if } I_{\mathrm{wb}}^{k}(v)=1 \text{ and } v \text{ has exactly two } k\text{-pre-branching children} \\ & \text{and no } k\text{-weakly-branching child;} \\ 1, & \text{if } I_{\mathrm{wb}}^{k}(v)=1 \text{ and } v \text{ has exactly one } k\text{-weakly-branching child} \\ & \text{and this child } u \text{ has } J_{\mathrm{w}}^{k}(u)=0; \\ 2, & \text{if } I_{\mathrm{wb}}^{k}(v)=1 \text{ and } v \text{ has exactly one } k\text{-weakly-branching child} \\ & \text{and this child } u \text{ has } J_{\mathrm{w}}^{k}(u)=1; \\ 0, & \text{otherwise.}\end{cases}$$

Definition 2.4. (label $L_{T^{[v]}}(v)$, value $|L_{T^{[v]}}(v)|$) Let $T^{[v]}$ be a rooted tree. The *label* of v in $T^{[v]}$, denoted by $L_{T^{[v]}}(v)$, is a sequence

$$(s_1, v_1; s_2, v_2; \dots; s_m, v_m; I_{\mathrm{wb}}^{s_m}(v), J_{\mathrm{w}}^{s_m}(v); I_{\mathrm{pb}}^{s_m}(v), J^{s_m}(v)),$$

where s_i and v_i are defined in the following procedure:

1. If $T^{[v]}$ contains only one vertex, then $s_1 = 1$, $v_1 =\perp$, $I_{\mathrm{wb}}^{s_i}(v) = J_{\mathrm{w}}^{s_i}(v) = I_{\mathrm{pb}}^{s_i}(v) = J^{s_i}(v) = 0$, and return $L_{T^{[v]}}(v) = (1, \perp; 0, 0; 0, 0)$; otherwise, set $i \leftarrow 1$ and $T_1^{[v]} \leftarrow T^{[v]}$.

2. Set $s_i \leftarrow \mathsf{c}_1(T_1^{[v]})$. Then we have one of the following cases:

   (a) If v is an s_i-branching vertex in $T_1^{[v]}$, then $I_{\mathrm{wb}}^{s_i}(v) = J_{\mathrm{w}}^{s_i}(v) = I_{\mathrm{pb}}^{s_i}(v) = J^{s_i}(v) = 0$, and return $L_{T^{[v]}}(v) = (s_1, v_1; \dots; s_i, v; 0, 0; 0, 0)$.

   (b) If v has an s_i-branching descendant in $T_1^{[v]}$, let v_i be this vertex. Set $T_1^{[v]} \leftarrow T_1^{[v]} - V(T_1^{[v_i]})$, $i \leftarrow i + 1$, and go back to Step 2.

   (c) If v is an s_i-weakly-branching vertex in $T_1^{[v]}$, then $I_{\mathrm{pb}}^{s_i}(v) = J^{s_i}(v) = 0$, $I_{\mathrm{wb}}^{s_i}(v) = 1$, and $J_{\mathrm{w}}^{s_i}(v)$ can be determined by Definition 2.3; return $L_{T^{[v]}}(v) = (s_1, v_1; \dots; s_i, \perp; 1, J_{\mathrm{w}}^{s_i}(v); 0, 0)$.

   (d) If v is an s_i-pre-branching vertex in $T_1^{[v]}$, then $I_{\mathrm{wb}}^{s_i}(v) = J_{\mathrm{w}}^{s_i}(v) = J^{s_i}(v) = 0$, and $I_{\mathrm{pb}}^{s_i}(v) = 1$; return $L_{T^{[v]}}(v) = (s_1, v_1; \dots; s_i, \perp; 0, 0; 1, 0)$.

   (e) $I_{\mathrm{wb}}^{s_i}(v) = J_{\mathrm{w}}^{s_i}(v) = I_{\mathrm{pb}}^{s_i}(v) = 0$, and $J^{s_i}(v)$ can be determined by Definition 2.3; return $L_{T^{[v]}}(v) = (s_1, v_1; \dots; s_i, \perp; 0, 0; 0, J^{s_i}(v))$.

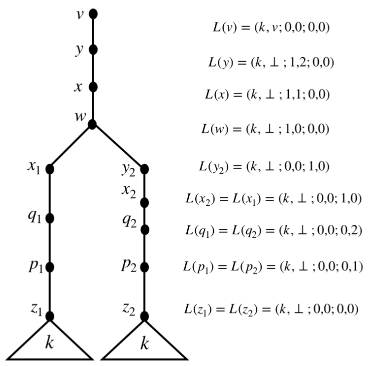
The *value* of $L_{T^{[v]}}(v)$, denoted by $|L_{T^{[v]}}(v)|$, is equal to s_1, which is also the copnumber of $T^{[v]}$.

From Definition 2.4, we know that $\mathsf{c}_1(T^{[v]}) = |L_{T^{[v]}}(v)| = s_1$.

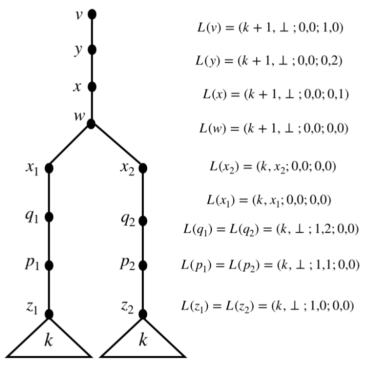
If there is no ambiguity, we simply use $L(v)$ for $L_{T^{[v]}}(v)$. In Algorithm 1, we will compute the copnumber of subtrees in the reverse order $s_m, s_{m-1}, \ldots, s_1$. For convenience, in the rest of the paper we let $t_i = s_{m-i+1}$ and $x_i = v_{m-i+1}$; i.e.,

$$L(v) = (t^v_m, x^v_m; \ldots; t^v_1, x^v_1; I^{t^v_1}_{\mathrm{wb}}(v), J^{t^v_1}_{\mathrm{w}}(v); I^{t^v_1}_{\mathrm{pb}}(v), J^{t^v_1}(v)),$$

where the superscript v in t^v_i and x^v_i are used to refer to the vertex v. So, $|L(v)| = t^v_m$. Note that only x^v_1 can be a "$\perp$" sign, which means that neither v is an t^v_1-branching vertex in $T^{[v]} - \bigcup_{i=2}^{m} V(T^{[x^v_i]})$ nor it has an t^v_1-branching descendant in $T^{[v]} - \bigcup_{i=2}^{m} V(T^{[x^v_i]})$. We call the first pair (t^v_m, x^v_m) an *item* associated with $T^{[v]}$, and call each pair (t^v_i, x^v_i), $1 \le i < m$, an *item* associated with subtree $T^{[v]} - \bigcup_{j=i+1}^{m} V(T^{[x^v_j]})$, where t^v_i is called the *key* of the item and x^v_i is the *attribute*.


v
y
x
w
x_1
y_2
x_2
q_1
q_2
p_1
p_2
z_1
z_2
k
k
$L(v) = (k, v; 0,0; 0,0)$
$L(y) = (k, \perp; 1,2; 0,0)$
$L(x) = (k, \perp; 1,1; 0,0)$
$L(w) = (k, \perp; 1,0; 0,0)$
$L(y_2) = (k, \perp; 0,0; 1,0)$
$L(x_2) = L(x_1) = (k, \perp; 0,0; 1,0)$
$L(q_1) = L(q_2) = (k, \perp; 0,0; 0,2)$
$L(p_1) = L(p_2) = (k, \perp; 0,0; 0,1)$
$L(z_1) = L(z_2) = (k, \perp; 0,0; 0,0)$

(a)


v
y
x
w
x_1
x_2
q_1
q_2
p_1
p_2
z_1
z_2
k
k
$L(v) = (k+1, \perp; 0,0; 1,0)$
$L(y) = (k+1, \perp; 0,0; 0,2)$
$L(x) = (k+1, \perp; 0,0; 0,1)$
$L(w) = (k+1, \perp; 0,0; 0,0)$
$L(x_2) = (k, x_2; 0,0; 0,0)$
$L(x_1) = (k, x_1; 0,0; 0,0)$
$L(q_1) = L(q_2) = (k, \perp; 1,2; 0,0)$
$L(p_1) = L(p_2) = (k, \perp; 1,1; 0,0)$
$L(z_1) = L(z_2) = (k, \perp; 1,0; 0,0)$

(b)

Figure 1: Labels of vertices

We explain labels with the following example.

Example 2.5. Consider the tree in Figure 1(a). Suppose that v is the root and $L(z_1) = L(z_2) = (k, \perp; 0, 0; 0, 0)$. Then $J^k(p_1) = J^k(p_2) = 1$ and $J^k(q_1) = J^k(q_2) = 2$. So $I^k_{\mathrm{pb}}(x_1) = I^k_{\mathrm{pb}}(x_2) = 1$ and both x_1 and x_2 are k-pre-branching vertices with label $(k, \perp; 0, 0; 1, 0)$. Note that y_2 is also a k-pre-branching vertex with the same label. The vertex w is a k-weakly-branching vertex since there are exactly two components in the forest $T^{[w]} - w$ such that each of the two components is rooted at a k-pre-branching vertex. The vertex v is a k-branching vertex as it has a k-weakly-branching child y and $J^k_{\mathrm{w}}(y) = 2$.

In Figure 1(b), suppose that v is the root and $L(z_1) = L(z_2) = (k, \perp; 1, 0; 0, 0)$. Then x_1 and x_2 are k-branching vertices with labels $(k, x_1; 0, 0; 0, 0)$ and $(k, x_2; 0, 0; 0, 0)$ respectively. As w has two k-branching children x_1 and x_2, the label of w is $(k+1, \perp; 0, 0; 0, 0)$. Hence, v is a $(k+1)$-pre-branching vertex with label $(k+1, \perp; 0, 0; 1, 0)$.

Definition 2.6. Let $T^{[u]}$ be a tree with root u whose children are $v_1, \ldots, v_d$. Suppose $c_1^*(T^{[u]} - u) = k \geq 1$. The counters $\#^k_{\mathrm{pb}}(T^{[u]} - u)$, $\#^k_{\mathrm{wb}}(T^{[u]} - u)$, $\#^k_{\mathrm{c}}(T^{[u]} - u)$, $h^k(T^{[u]} - u)$ and $h^k_{\mathrm{w}}(T^{[u]} - u)$ are defined as follows:

$$\#^k_{\mathrm{pb}}(T^{[u]} - u) = \sum_{j=1}^{d} I^k_{\mathrm{pb}}(v_j),$$

$$\#^k_{\mathrm{wb}}(T^{[u]} - u) = \sum_{j=1}^{d} I^k_{\mathrm{wb}}(v_j),$$

$$\begin{aligned} \#^k_{\mathrm{c}}(T^{[u]} - u) &= \left|\left\{ j \;\middle|\; \mathsf{c}_1(T^{[v_j]}) = k \text{ for } j \in \{1, \ldots, d\} \right\}\right|, \\ h^k(T^{[u]} - u) &= \max\left\{ J^k(v_j) \;\middle|\; j \in \{1, \ldots, d\} \right\}, \\ h^k_{\mathrm{w}}(T^{[u]} - u) &= \max\left\{ J^k_{\mathrm{w}}(v_j) \;\middle|\; j \in \{1, \ldots, d\} \right\}. \end{aligned}$$

3 Algorithms

```
function COMPUTE-LABEL($T^{[u]}$)
Input: A tree $T^{[u]}$ with children $v_1, \ldots, v_d$. Let $k = c_1^*(T^{[u]} - u)$ and for $1 \le j \le d$,
$L_{T^{[v_j]}}(v_j) = (t^{v_j}, \perp; I_{\mathrm{wb}}^{t^{v_j}}(v_j), J_{\mathrm{w}}^{t^{v_j}}(v_j); I_{\mathrm{pb}}^{t^{v_j}}(v_j), J^{t^{v_j}}(v_j))$.
Output: The label $L_{T^{[u]}}(u)$ whose first component is $\mathtt{c}_1(T^{[u]})$.
1: if $\#_{\mathrm{wb}}^k(T^{[u]} - u) > 1$ then
2:     return $L_{T^{[u]}}(u) = (k+1, \perp; 0, 0; 0, 0)$.
3: else if $\#_{\mathrm{wb}}^k(T^{[u]} - u) = 1$ and $\#_{\mathrm{pb}}^k(T^{[u]} - u) \ge 1$ then
4:     return $L_{T^{[u]}}(u) = (k+1, \perp; 0, 0; 0, 0)$.
5: else if $\#_{\mathrm{wb}}^k(T^{[u]} - u) = 1$, $\#_{\mathrm{pb}}^k(T^{[u]} - u) = 0$ and $\#_{\mathrm{c}}^k(T^{[u]} - u) \ge 2$ then
6:     if $h_{\mathrm{w}}^k(T^{[u]} - u) = 2$ then
7:         return $L_{T^{[u]}}(u) = (k+1, \perp; 0, 0; 0, 0)$.
8:     else if $h_{\mathrm{w}}^k(T^{[u]} - u) = 1$ and $h^k(T^{[u]} - u) \ge 1$ then
9:         return $L_{T^{[u]}}(u) = (k+1, \perp; 0, 0; 0, 0)$
10:    else if $h_{\mathrm{w}}^k(T^{[u]} - u) = 1$ and $h^k(T^{[u]} - u) = 0$ then
11:        return $L_{T^{[u]}}(u) = (k, \perp; 1, 2; 0, 0)$
12:    else if $h_{\mathrm{w}}^k(T^{[u]} - u) = 0$ and $h^k(T^{[u]} - u) = 2$ then
13:        return $L_{T^{[u]}}(u) = (k+1, \perp; 0, 0; 0, 0)$
14:    else if $h_{\mathrm{w}}^k(T^{[u]} - u) = 0$ and $h^k(T^{[u]} - u) \le 1$ then
15:        return $L_{T^{[u]}}(u) = (k, \perp; 1, 1; 0, 0)$
16: else if $\#_{\mathrm{wb}}^k(T^{[u]} - u) = 1$ and $\#_{\mathrm{c}}^k(T^{[u]} - u) = 1$ then
17:    if $h_{\mathrm{w}}^k(T^{[u]} - u) = 2$ then
18:        return $L_{T^{[u]}}(u) = (k, u; 0, 0; 0, 0)$
19:    else if $h_{\mathrm{w}}^k(T^{[u]} - u) = 1$ then
20:        return $L_{T^{[u]}}(u) = (k, \perp; 1, 2; 0, 0)$
21:    else if $h_{\mathrm{w}}^k(T^{[u]} - u) = 0$ then
22:        return $L_{T^{[u]}}(u) = (k, \perp; 1, 1; 0, 0)$
23: else if $\#_{\mathrm{wb}}^k(T^{[u]} - u) = 0$ then
24:    if $\#_{\mathrm{pb}}^k(T^{[u]} - u) \ge 3$ then
25:        return $L_{T^{[u]}}(u) = (k+1, \perp; 0, 0; 0, 0)$
26:    else if $\#_{\mathrm{pb}}^k(T^{[u]} - u) = 2$ then
27:        return $L_{T^{[u]}}(u) = (k, \perp; 1, 0; 0, 0)$.
28:    else if $\#_{\mathrm{pb}}^k(T^{[u]} - u) = 1$ then
29:        return $L_{T^{[u]}}(u) = (k, \perp; 0, 0; 1, 0)$.
30:    else if $\#_{\mathrm{pb}}^k(T^{[u]} - u) = 0$ then
31:        if $h^k(T^{[u]} - u) = 2$ then
32:            return $L_{T^{[u]}}(u) = (k, \perp; 0, 0; 1, 0)$
33:        else if $h^k(T^{[u]} - u) = 1$ then
34:            return $L_{T^{[u]}}(u) = (k, \perp; 0, 0; 0, 2)$
35:        else if $h^k(T^{[u]} - u) = 0$ then
36:            return $L_{T^{[u]}}(u) = (k, \perp; 0, 0; 0, 1)$
```

Algorithm 1 Computing the copnumber of a tree

Input: A tree T with n vertices, where $n \geq 12$.

Output: The label of the root of T.

```
1: Pick a vertex of $T$ as its root.
2: Sort the vertices of $T$ to a list $u_1, \dots, u_n$ such that every vertex is before its parent in the list. For each vertex that has no child, set its label as $(1, \perp; 0, 0; 0, 0)$.
3: If the root $u_n$ has obtained a label $L_{T^{[u_n]}}(u_n)$, then output this label; otherwise, let $u$ be the first unlabeled vertex in the list currently. Run Steps 4 to 11 to compute $L_{T^{[u]}}(u)$.
4: Let $v_j$, $1 \leq j \leq d$, be all children of $u$ with labels $L_{T^{[v_j]}}(v_j)$. Let $I_\perp$ be the subset of children whose label contains $\perp$ and let $I_b$ be the subset of children whose label does not contain $\perp$.
5: Construct $T_1^{[u]}$, where $T_1^{[u]} = T^{[u]} - \bigcup_{y \in I_b} V(T^{[x_1^y]}) - \bigcup_{y \in I_\perp} V(T^{[x_2^y]})$. For $v \in V(T_1) - \{u\}$, $L_{T_1^{[v]}}(v)$ is obtained by deleting all components except the last six components from $L_{T^{[v]}}(v)$.
6: If $u$ is the only vertex in $T_1^{[u]}$, then $L_{T_1^{[u]}}(u) = (1, \perp; 0, 0; 0, 0)$; otherwise, call COMPUTE-LABEL$(T_1^{[u]})$. Let $k = \mathsf{c}_1(T_1^{[u]})$.
7: For $1 \leq j \leq d$, if $v_j \in I_\perp$, let $L_j$ be a list obtained from $L_{T^{[v_j]}}(v_j)$ by deleting the last six components and items whose key is less than $k$; if $v_j \in I_b$, let $L_j$ be a list obtained from $L_{T^{[v_j]}}(v_j)$ by deleting the last four components and items whose key is less than $k$. Let $L_{d+1}$ be a list containing only the first item of $L_{T_1^{[u]}}(u)$.
8: If no key in $L_1, \dots, L_d, L_{d+1}$ is repeated, then $L_{T^{[u]}}(u) \leftarrow L_{T_1^{[u]}}(u)$ and insert the items of $L_1, \dots, L_d$ into $L_{T^{[u]}}(u)$. Go to Step 3.
9: Find the largest repeated key $k^*$ in the lists $L_1, \dots, L_d, L_{d+1}$.
10: Let $K = (k_1, \dots, k_\ell)$ be a list containing the distinct keys from $L_1, \dots, L_{d+1}$ satisfying that the keys in $K$ are decreasing and are greater than or equal to $k^*$.
11: Find the smallest index $h$ in $K$, where $1 \leq h \leq \ell$, such that $k_h = k_{h+1} + 1 = \cdots = k_\ell + (\ell - h)$. Update $K \leftarrow (k_1, \dots, k_{h-1}, k'_h)$ where $k'_h = k_h + 1$. Create a list $X = (Q_1, \dots, Q_{h-1}, Q_h)$, where $Q_i = (k_i, x_i)$, $1 \leq i \leq h - 1$, is an item with key $k_i$ and attribute $x_i$ (note that $x_i$, $1 \leq i \leq h-1$, is a $k_i$-branching vertex in some subtree) and $Q_h = (k'_h, \perp)$. Insert $(0, 0; 0, 0)$ at the end of $X$. Set $L_{T^{[u]}}(u) \leftarrow X$. Go to Step 3.
```

4 Examples

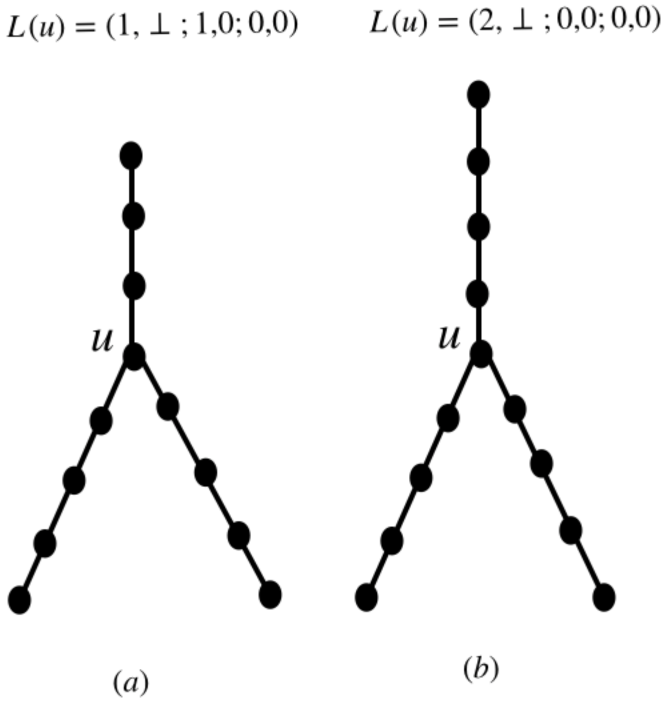

$L(u) = (1, \perp ; 1,0; 0,0)$
$L(u) = (2, \perp ; 0,0; 0,0)$
u
u
(a)
(b)

Figure 2: (a) A tree $T^{[u]}$ with $L_{T^{[u]}}(u) = (1, \perp; 1, 0; 0, 0)$. (b) A tree $T^{[u]}$ with $L_{T^{[u]}}(u) = (2, \perp; 0, 0; 0, 0)$.

Example 4.1. Test Algorithm 1 on the tree in Figure 2(a).

Input: The tree in Figure 2(a).

Output: If you pick the vertex u as the root of T in Step 1, then Algorithm 1 should output $L_{T^{[u]}}(u) = (1, \perp; 1, 0; 0, 0)$.

Example 4.2. Test Algorithm 1 on the tree in Figure 2(b).

Input: The tree in Figure 2(b).

Output: If you pick the vertex u as the root of T in Step 1, then Algorithm 1 should output $L_{T^{[u]}}(u) = (2, \perp; 0, 0; 0, 0)$.

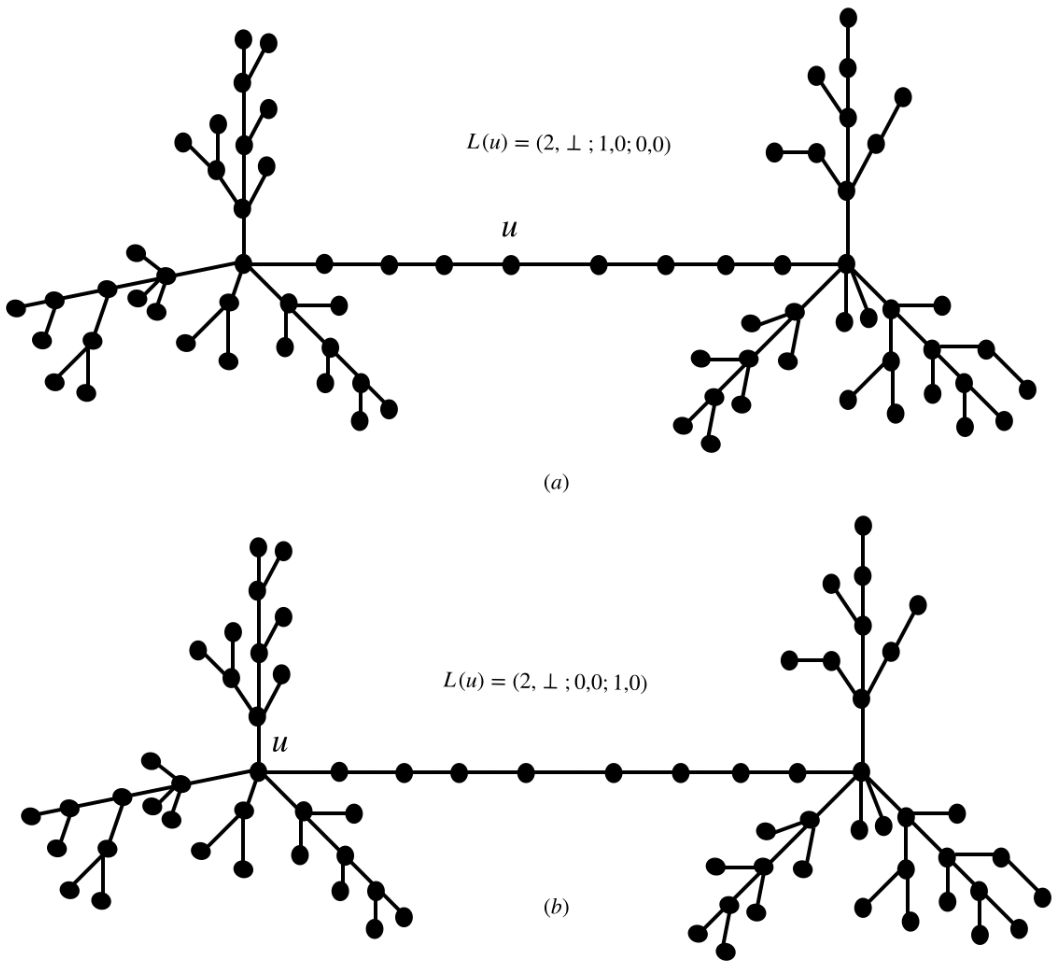

$L(u) = (2, \perp ; 1,0; 0,0)$
u
(a)
$L(u) = (2, \perp ; 0,0; 1,0)$
u
(b)

Figure 3: (a) A tree $T^{[u]}$ with $L_{T^{[u]}}(u) = (2, \perp; 1, 0; 0, 0)$. (b) A tree $T^{[u]}$ with $L_{T^{[u]}}(u) = (2, \perp; 0, 0; 1, 0)$.

Example 4.3. Test Algorithm 1 on the tree in Figure 3(a).

Input: The tree in Figure 3(a).

Output: If you pick the vertex u as the root of T in Step 1, then Algorithm 1 should output $L_{T^{[u]}}(u) = (2, \perp; 1, 0; 0, 0)$.

Example 4.4. Test Algorithm 1 on the tree in Figure 3(b).

Input: The tree in Figure 3(b).

Output: If you pick the vertex u as the root of T in Step 1, then Algorithm 1 should output $L_{T^{[u]}}(u) = (2, \perp; 0, 0; 1, 0)$.

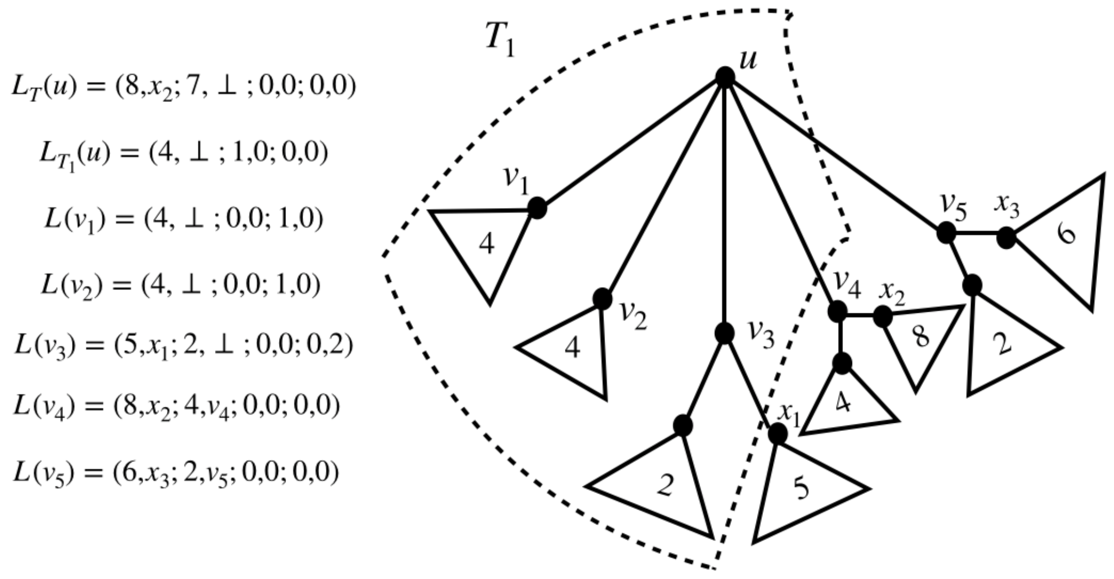

$L_T(u) = (8, x_2; 7, \perp; 0,0; 0,0)$
$L_{T_1}(u) = (4, \perp; 1,0; 0,0)$
$L(v_1) = (4, \perp; 0,0; 1,0)$
$L(v_2) = (4, \perp; 0,0; 1,0)$
$L(v_3) = (5, x_1; 2, \perp; 0,0; 0,2)$
$L(v_4) = (8, x_2; 4, v_4; 0,0; 0,0)$
$L(v_5) = (6, x_3; 2, v_5; 0,0; 0,0)$
T_1
u
v_1
v_2
v_3
v_4
v_5
x_1
x_2
x_3
4
4
2
5
4
8
2
6

Figure 4: A tree $T^{[u]}$ with $L_{T^{[u]}}(u) = (8, x_2; 7, \perp; 0, 0; 0, 0)$.

Example 4.5. Let u be the root of the tree in Figure 4 which has five children v_1, v_2, v_3, v_4, v_5. Consider Step 4 of Algorithm 1. Suppose the labels of v_1, v_2, v_3, v_4, v_5 are those given in the figure. In Step 5, T_1 should be the subtree inside dotted region in the figure. In Step 6, function COMPUTE-LABEL should return $L_{T_1^{[u]}}(u) = (4, \perp; 1, 0; 0, 0)$. After executing Steps 7-11 and then going back to Step 3, the algorithm should output $L_{T^{[u]}}(u) = (8, x_2; 7, \perp; 0, 0; 0, 0)$.

5 Numerical experiments

Use the program you wrote in Assignment 1 to randomly create 10 different trees T_i, $1 \leq i \leq 10$, so that the number vertices in T_i is $30i$. Run Algorithm 1 on each T_i, $1 \leq i \leq 10$, ten times, each time randomly pick a vertex as the root of the tree. So you should submit 100 different output results in these numerical experiments.