An Explanation of Lengauer-Tarjan Dominators Algorithm

Jayadev Misra
Dept. of Computer Science
The University of Texas
Austin, Texas, 78712

May 2, 2023
Revised May 25, 2023

1 Introduction

This note explains the Lengauer-Tarjan dominators algorithm given in "A Fast Algorithm for Finding Dominators in a Flowgraph" [2], henceforth referred to as the "original paper". The algorithm is ingenious; though relatively simple to state, its underlying proofs are fairly involved. I have tried to present the proofs in a more structured fashion in this note.

2 Depth-first search

Given is a graph G with a specific node r from which every other node is reachable. Let T be a depth-first tree (or *dfs-tree*) rooted at r. Each node in the dfs-tree is identified by its preorder number, starting with $r = 1$. In the example dfs-tree in Figure 1 (page 2), taken from the original paper, the tree edges in the right figure are shown as solid arrows and the non-tree edges as dashed arrows.

2.1 Notation

Notation for graphs The notations for edges and paths in G and T are summarized in Table 1 (page 2). "Any path", with a $*$ symbol, is a path of length zero or more, so its end nodes may be identical. A "non-zero path", with a + symbol, is a path of length one or more, so its end nodes are different. A path may have an optional label which appears over the path, shown as p and q in the following table. All paths are simple.

In proofs, I use edges and paths as logical propositions, such as $x \overset{*p}{\rightsquigarrow} y$ to denote the proposition "there is a path p from node x to node y". Thus, $x \overset{*}{\rightarrow} y$ means that y is a descendant of x (and x an ancestor of y), and $x \overset{+}{\rightarrow} y$ for y is

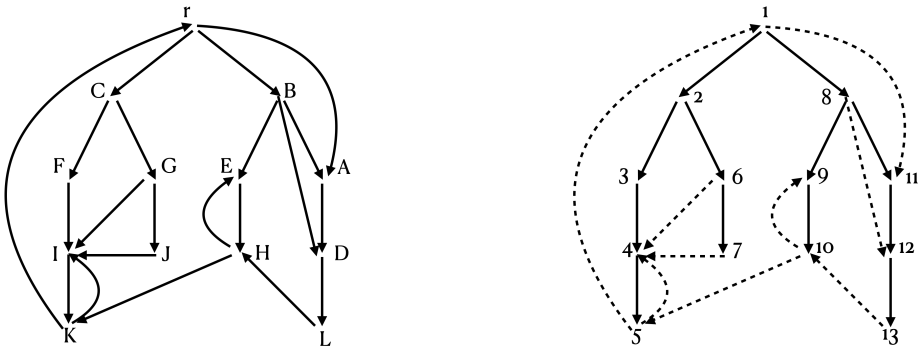

Figure 1: A graph and its dfs-tree; the non-tree edges are dashed arrows

	edge	any path	non-zero path	
graph	$x \rightsquigarrow y$	$x \overset{*p}{\rightsquigarrow} y$	$x \overset{+q}{\rightsquigarrow} y$	
tree	$x \rightarrow y$	$x \overset{*p}{\rightarrow} y$	$x \overset{+q}{\rightarrow} y$	

Table 1: Terminology for graph and tree paths and edges

a proper descendant of x (and x a proper ancestor of y). Because of preorder numbers assigned to nodes, $(x \overset{*}{\rightarrow} y) \;\Rightarrow\; (x \leq y)$, and $(x \overset{+}{\rightarrow} y) \;\Rightarrow\; (x < y)$. Observe that for any w, $r \overset{*}{\rightsquigarrow} w$ and $r \overset{*}{\rightarrow} w$.

Write $v \in x \overset{*p}{\rightsquigarrow} y$ to denote that v is a node on path p, possibly x or y.

Notation for logical formulae and proofs A quantified formula is of the form $(\otimes x : q(x) : e(x))$, where $\otimes$ is a commutative and associative binary operator that can be applied to pairs of values in the given domain; see Misra[3], Sections 2.6.2 and 2.6.5. The value of the formula is $\otimes\{e(x) \mid q(x)\}$. That is, apply $\otimes$ in arbitrary order over all the values in the set $\{e(x) \mid q(x)\}$. Logical connective $\forall$ and the arithmetic operator min are used in this note for $\otimes$. So, $(\forall x : x \in u \overset{+}{\rightarrow} v : x > w)$ asserts that all proper descendants of u up to and including v are greater than w. And, $(\min x : x \in u \overset{+}{\rightarrow} v : f(v))$ is the smallest value of $f(x)$ over all x where x is in $u \overset{+}{\rightarrow} v$.

Proofs are written in a stylized fashion; see Misra[3], Section 2.7.2. □

Lemma 1 (Lemma 1 in the original paper) Given $v \overset{*p}{\rightsquigarrow} w$ where $v \leq w$, there

is a node in p that is a common ancestor of v and w. □

Observe that (1) if $v = w$, or v is the smallest node in p, v is the common ancestor of v and w, and (2) if $v < w$, the common ancestor is a proper ancestor of w.

3 Dominators

Dominator v of w, $w \neq r$, is a node other than w that appears on every path $r \overset{+}{\rightsquigarrow} w$. Therefore, r is a dominator of every node w, $w \neq r$. The dominators of the nodes in Figure 1 are shown in a dominator tree in Figure 2 where the ancestors of a node are its dominators..

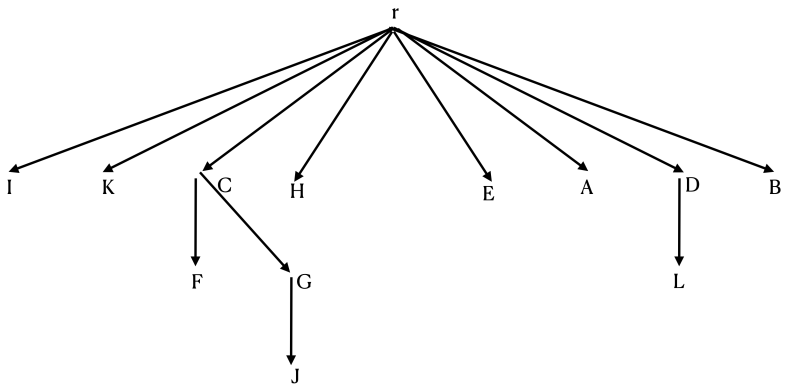

Figure 2: Dominator tree

Proposition 1 The dominators of any node w, $w \neq r$, occur in the same order on every path $r \overset{+}{\rightsquigarrow} w$.

Proof: Proof is by contradiction. Consider two different dominators, x and y, of w. We may assume that both x and y are different from r because r occurs as the first node in any path $r \overset{+}{\rightsquigarrow} w$. Suppose x and y occur in different orders in two paths, as shown below.

$$
\begin{array}{ll}
 & r \overset{+p}{\rightsquigarrow} x \overset{+p'}{\rightsquigarrow} y \overset{+p''}{\rightsquigarrow} w,\ r \overset{+q}{\rightsquigarrow} y \overset{+q'}{\rightsquigarrow} x \overset{+q''}{\rightsquigarrow} w \\
\Rightarrow & \{y \notin p,\ y \notin q''\} \\
 & r \overset{+p}{\rightsquigarrow} x \overset{+q''}{\rightsquigarrow} w,\ y \notin p,\ y \notin q'' \\
\Rightarrow & \{y \notin r \overset{+pq''}{\rightsquigarrow} w, \text{ and } y \text{ is a dominator of } w\} \\
 & \mathit{false}
\end{array}
$$

□

The *immediate dominator* of w, $w \neq r$, written as $id(w)$, is the last dominator along every path $r \overset{+}{\rightsquigarrow} w$. Note that each node w, $w \neq r$, has an immediate dominator, because each node has r as a dominator. And, the immediate dominator of a node is unique.

Proposition 2 Given that v is a dominator of w, $v \overset{*}{\rightarrow} id(w)$.

Proof: Along the tree path $r \overset{*p}{\rightarrow} w$

$$\begin{array}{ll} & v \text{ and } id(w) \text{ are both dominators of } w \\ \Rightarrow & \{v \in p,\ id(w) \in p;\ id(w) \text{ is the last dominator in } p\} \\ & v \overset{*}{\rightarrow} id(w) \end{array}$$

□

For the graph in Figure 1 (page 2), the immediate dominators are given by the parent of each node in the dominator tree in Figure 2. In general for x an ancestor of y in the dfs-tree, $id(x)$ is not necessarily an ancestor of $id(y)$ in the dfs-tree. Absence of such a monotonicity property makes it harder to compute the immediate dominators.

4 Semidominator

The dominators of w are some of its proper ancestors, as shown in Proposition 2 (page 5). However, as the paragraph preceding this section shows, there is no easy way to determine the immediate dominators easily from the dfs-tree. The original paper introduces *semidominator* to solve the problem.

To motivate the notion of semidominator, consider a path $v \overset{+q}{\rightsquigarrow} w$ in which all internal nodes are greater than w. I claim that any dominator u of w is an ancestor of v. To see this, consider the path $r \overset{*p}{\rightarrow} v \overset{+q}{\rightsquigarrow} w$. If u is not an ancestor of v, then (1) $u \notin p$, and (2) $u \neq w$ because u is a dominator of w, and u, being an ancestor of w, is smaller than w, so it is not in q. Then $u \notin r \overset{*p}{\rightarrow} v \overset{+q}{\rightsquigarrow} w$, contradicting that u is a dominator of w. If we choose v to be the smallest possible node such that in $v \overset{+q}{\rightsquigarrow} w$ all internal nodes are greater than w, then we can eliminate all nodes larger than v as dominators of w. Such a v is the semidominator of w.

Node v is a *semidominator-candidate* of w, or *sd-candidate* for w, if there is a path $v \overset{+}{\rightsquigarrow} w$ in which every internal node is greater than w; call such a path a *sd-path*. Note that a sd-candidate of w is different from w. Since $parent(w) \rightarrow w$ has no internal node it satisfies the condition for sd-path vacuously, so $parent(w)$ is a sd-candidate for w.

The smallest semidominator-candidate of w is the *semidominator* of w, written as $sd(w)$. Every node w, except r, has a semidominator because $parent(w)$ is a sd-candidate. The semidominator is unique because nodes have distinct numbers. The sd-path from $sd(w)$ to w may not be unique.

In Figure 1 (page 2), $sd(3) = 2$ because of the edge from its parent. And $sd(4) = 1$ because of the path $1 \rightsquigarrow 8 \rightsquigarrow 9 \rightsquigarrow 10 \rightsquigarrow 5 \rightsquigarrow 4$, though $2 \rightsquigarrow 6 \rightsquigarrow 4$

is another sd-path to 4. The path from $sd(x)$ to any node x is not necessarily unique; the path $1 \rightsquigarrow 11 \rightsquigarrow 12 \rightsquigarrow 13 \rightsquigarrow 10 \rightsquigarrow 5 \rightsquigarrow 4$ is another sd-path from 1 to 4.

Lemma 2 (Lemma 4 in the original paper) For every w, $w \neq r$, there is a tree-path $r \xrightarrow{*} id(w) \xrightarrow{*} sd(w) \xrightarrow{+} w$.

Proof:

$parent(w)$ is a sd-candidate for w; so $sd(w) \leq parent(w) < w$

$\Rightarrow$ {take sd-path $sd(w) \overset{+q}{\rightsquigarrow} w$; apply Lemma 1 (page 2)}
$sd(w) \overset{*}{\rightsquigarrow} v \overset{*}{\rightsquigarrow} w$, where v is a common ancestor of $sd(w)$ and w

$\Rightarrow$ {as an ancestor of w, $v \leq w$, but internal nodes of q are $> w$}
$sd(w)$ is a proper ancestor of w

$\Rightarrow$ {$id(w)$, a dominator of w, is on the path $r \xrightarrow{+} w$}
$id(w)$ is a proper ancestor of w

$\Rightarrow$ {the path $r \xrightarrow{*} sd(w) \overset{+q}{\rightsquigarrow} w$ includes the dominator $id(w)$.
$id(w)$ is a proper ancestor of w, so $id(w) < w$.
Then $id(w)$ is not an internal node in the sd-path q}
$r \xrightarrow{*} id(w) \xrightarrow{*} sd(w) \xrightarrow{+} w$ □

Note that $id(w)$ and $sd(w)$ may be identical.

Basis for computing the semidominators Computation of $sd(w)$ for all w, $w \neq r$, is essential to computing the dominators. Let $sd_1(w)$ be the minimum sd-candidate for w over paths that have no internal node, and $sd_2(w)$ the minimum sd-candidate over sd-paths that have some internal node. So, $sd(w) = \min(sd_1(w),\ sd_2(w))$.

Note that $sd_1(w)$ is always defined because $parent(w) \xrightarrow{+} w$ is a sd-path with no internal node. However, $sd_2(w)$ may not be defined because there may be no sd-path to w with an internal node; then $sd_2(w)$ is ∞.

Proposition 3 $sd_1(w)$ is the smallest predecessor of w, i.e.
$sd_1(w) = (\min v : v \rightsquigarrow w : v)$.

Proof: Any sd-path to w that has no internal node is of the form $v \rightsquigarrow w$. From the definition of $sd_1(w)$, the result follows. □

Observe that $parent(w) \rightarrow w$, so $sd_1(w) \leq parent(w) < w$.

Proposition 4 $sd_2(w)$ is the minimum $sd(u)$ over all u, where $u > w$ and for some v, $u \xrightarrow{*} v \rightsquigarrow w$, i.e.
$sd_2(w) = (\min u, v : u > w,\ u \xrightarrow{*} v \rightsquigarrow w : sd(u))$.

Proof: Assume that there is a sd-path with some internal node; if there is no such path, according to the formal definition of min, $sd_2(w) = \infty$. Let u be the minimum internal node on a sd-path $sd_2(w) \overset{+p}{\rightsquigarrow} u \overset{*q}{\rightsquigarrow} w$. I show that $sd_2(w) = sd(u)$.

$sd_2(w) \overset{+p}{\rightsquigarrow} u \overset{*q}{\rightsquigarrow} w$

$\Rightarrow$ {internal nodes of p are $> u$,
because u is the minimum internal node in pq}
$sd_2(w) \overset{+p}{\rightsquigarrow} u$ is a sd-path to u, so $sd(u) \leq sd_2(w)$

$\Rightarrow$ {all internal nodes on $sd(u) \overset{+}{\rightsquigarrow} u$ are $> u$, hence $> w$}
$sd(u) \overset{+}{\rightsquigarrow} u \overset{*q}{\rightsquigarrow} w$ is a sd-path to w, so $sd_2(w) \leq sd(u)$

$\Rightarrow$ {from $sd(u) \leq sd_2(w)$ and $sd_2(w) \leq sd(u)$}
$sd_2(w) = sd(u)$

We can give a stronger characterization of the appropriate u. Let v be the penultimate node in $u \overset{+p}{\rightsquigarrow} w$, so the path is of the form $u \overset{*q}{\rightsquigarrow} v \rightsquigarrow w$ (u and v may be identical). Applying Lemma 1 (page 2)to $u \overset{*q}{\rightsquigarrow} v$, there is a common ancestor of u and v in p. Since u is the smallest node in q, u is an ancestor of v . So,

$sd_2(w) = (\min u, v : u > w,\ u \overset{*}{\rightarrow} v \rightsquigarrow w : sd(u))$. □

5 Basic results for immediate dominators

Notation Let $\overline{w}$ be a proper descendant of $sd(w)$ and ancestor of w, i.e. $sd(w) \overset{+}{\rightarrow} \overline{w} \overset{*}{\rightarrow} w$, such that $sd(\overline{w})$ has the minimum value among all such nodes, so $sd(\overline{w}) = (\min u : sd(w) \overset{+}{\rightarrow} u \overset{*}{\rightarrow} w : sd(u))$.

In case several nodes satisfy this definition of $\overline{w}$, pick any one arbitrarily. In Figure 1 (page 2) $sd(\overline{3})$ can be either 1 or 3 because both of these nodes are proper descendants of $sd(3) = 0$ and ancestors of 3, and $sd(1) = sd(3) = 0$. □

At this stage we have w and $\overline{w}$, and their id and sd values. Determining the order of these six quantities in a tree path is the crux of the paper. We have $sd(\overline{w}) \overset{*}{\rightarrow} sd(w) \overset{+}{\rightarrow} \overline{w} \overset{*}{\rightarrow} w$. We know $id(w) \overset{*}{\rightarrow} sd(w)$ and $id(\overline{w}) \overset{*}{\rightarrow} sd(\overline{w})$. But we don't know the relationship between $\{id(\overline{w}), id(w)\}$ or $\{id(w), sd(\overline{w})\}$. The next two lemmas help fix the orders.

Regard a pair of nodes $id(v)$ and v as a pair of matching parentheses. Parentheses lemma shows that all parentheses pairs on any tree path are properly nested. That is, there is no tree path of the form $id(v) \overset{+}{\rightarrow} id(w) \overset{*}{\rightarrow} v \overset{*}{\rightarrow} w$.

Lemma 3 (Parentheses lemma; Lemma 5 in the original paper)
$id(w) \overset{+}{\rightarrow} v \overset{+}{\rightarrow} w \ \equiv\ id(w) \overset{*}{\rightarrow} id(v) \overset{+}{\rightarrow} w$.

Proof: I prove one side of the equivalence:
$id(w) \overset{+}{\rightarrow} v \overset{+}{\rightarrow} w \ \Rightarrow\ id(w) \overset{*}{\rightarrow} id(v) \overset{+}{\rightarrow} w$.
The other side, $id(w) \overset{+}{\rightarrow} id(v) \overset{+}{\rightarrow} w \ \Rightarrow\ id(w) \overset{*}{\rightarrow} v \overset{+}{\rightarrow} w$, follows by symmetry by switching the roles of v and w in the following proof.

Proof is by contradiction.

$id(v) \xrightarrow{+} id(w) \xrightarrow{*} v \xrightarrow{*q} w$

$\Rightarrow$ $\{id(v)$ is v's immediate dominator; so $id(w)$ is not a dominator of $v\}$

there is $r \overset{*p}{\rightsquigarrow} v$ that does not include $id(w)$

$\Rightarrow$ $\{v \xrightarrow{*q} w$ does not include $id(w)\}$

$r \overset{*p}{\rightsquigarrow} v \xrightarrow{*q} w$ does not include $id(w)$

$\Rightarrow$ $\{id(w)$ is a dominator of $w\}$

false □

Since $id(v) \xrightarrow{+} v$, the right side of the equivalence can be strengthened to $id(w) \xrightarrow{*} id(v) \xrightarrow{+} v \xrightarrow{*} w$.

Lemma 4 Let v be an ancestor of w and for all y, where $v \xrightarrow{+} y \xrightarrow{*} w$, $sd(y) \geq v$. Then $v \xrightarrow{*} id(w)$.

Proof: I prove the stronger conclusion that v is a dominator of w, from which $v \xrightarrow{*} id(w)$ follows by applying Proposition 2 (page 4).

I show the contrapositive, that if v is not a dominator of w then there is a y, where $v \xrightarrow{+} y \xrightarrow{*} w$, such that $sd(y) < v$.

Since v is not a dominator of w, there is a path $r \overset{*p}{\rightsquigarrow} w$ that does not include v. Let y be the closest proper descendant of v that is on p; such a descendant exists because w, a proper descendant of v, is on p. I prove $sd(y) < v$.

Let x be the closest node preceding y on p such that $x < y$; such a x exists because for a preceding node r, $r \leq v < y$. In the subpath $x \overset{*q}{\rightsquigarrow} y$ all internal nodes are greater than y.

$x \overset{*q}{\rightsquigarrow} y$ is a sd-path, $x < y$

$\Rightarrow$ {definition of semidominator}

$sd(y) \leq x$, $x \overset{*q}{\rightsquigarrow} y$ is a sd-path, $x < y$

$\Rightarrow$ {apply Lemma 1 (page 2)on q}

$sd(y) \leq x$, x is a proper ancestor of y

$\Rightarrow$ {y is the closest proper descendant of v on p,
$(v \notin p, x \in p) \Rightarrow x \neq v$}

$sd(y) \leq x$, x is a proper ancestor of v

$\Rightarrow$ $\{sd(y) \leq x < v\}$

$sd(y) < v$ □

Theorem 1 (Theorem 3 in the paper) $id(w) = id(\overline{w})$.

Proof: For any y:

$y \in id(\overline{w}) \xrightarrow{*} sd(\overline{w}) \xrightarrow{*} sd(w) \xrightarrow{+} \overline{w} \xrightarrow{*} w$

$\Rightarrow$ {divide the given path into two overlapping parts}

$y \in id(\overline{w}) \xrightarrow{+} \overline{w}$ or $y \in sd(w) \xrightarrow{+} w$

$\Rightarrow$ {apply Lemma 3 (page 6) on the first term}

$id(\overline{w}) \xrightarrow{*} id(y) \xrightarrow{+} \overline{w}$ or $y \in sd(w) \xrightarrow{+} w$

$\Rightarrow$ {in the first term: $id(\overline{w}) \xrightarrow{*} id(y)$. So, $sd(y) \geq id(y) \geq id(\overline{w})$}

$sd(y) \geq id(\overline{w})$ or $y \in sd(w) \xrightarrow{+} w$

$\Rightarrow$ {definition of $sd(\overline{w})$ on the second term: $sd(y) \geq sd(\overline{w}) \geq id(\overline{w})$}

$sd(y) \geq id(\overline{w})$

$\Rightarrow$ {$sd(y) \geq id(\overline{w})$ for all y in $id(\overline{w}) \xrightarrow{+} w$; apply Lemma 4 (page 7)}

$id(\overline{w}) \xrightarrow{*} id(w)$

$\Rightarrow$ {$id(w) \xrightarrow{*} sd(w) \xrightarrow{+} \overline{w} \xrightarrow{+} w$; so $id(w) \xrightarrow{+} \overline{w} \xrightarrow{+} w$. Apply Lemma 3}

$id(\overline{w}) \xrightarrow{*} id(w)$, $id(w) \xrightarrow{*} id(\overline{w}) \xrightarrow{+} w$

$\Rightarrow$ {from $id(\overline{w}) \xrightarrow{*} id(w)$ and $id(w) \xrightarrow{*} id(\overline{w})$}

$id(w) = id(\overline{w})$ □

Theorem 1 provides a way of computing the immediate dominators in increasing order of node numbers: for every w set $id(w) := id(\overline{w})$. Since $\overline{w} \leq w$, $id(\overline{w})$ has been computed earlier if $\overline{w} \neq w$. The case $w = \overline{w}$ is addressed in the next theorem.

Theorem 2 (Theorem 2 in the paper)
If $w = \overline{w}$ then $id(w) = sd(w)$.

Proof: From the definition of $sd(\overline{w})$,

every y, where $sd(w) \xrightarrow{+} y \xrightarrow{*} w$, has $sd(y) \geq sd(\overline{w})$

$\Rightarrow$ {$w = \overline{w}$ implies $sd(w) = sd(\overline{w})$}

every y, where $sd(w) \xrightarrow{+} y \xrightarrow{*} w$, has $sd(y) \geq sd(w)$

$\Rightarrow$ {apply Lemma 4 (page 7) using $sd(w)$ for v}

$sd(w) \xrightarrow{*} id(w)$

$\Rightarrow$ {Lemma 2 (page 5): $id(w) \xrightarrow{*} sd(w)$}

$id(w) = sd(w)$ □

6 Algorithm

6.1 Basis of the algorithm

The algorithm is based on the following two corollaries. Corollary 1 is used to compute all the sd values first, followed by the computations of immediate dominators using Corollary 2.

Corollary 1 (Theorem 4 in the original paper)
For any w, $w \neq r$, $sd(w) = \min(sd_1(w),\ sd_2(w))$ where:
$sd_1(w) = (\min v : v \rightsquigarrow w : v)$, and
$sd_2(w) = (\min u, v : u > w,\ u \xrightarrow{*} v \rightsquigarrow w : sd(u))$.

Proof: From Propositions 3 (page 5) and 4 (page 5). □

Corollary 2 (Corollary 1 in the original paper)

$$id(w) = \begin{cases} sd(w) & \text{if } w = \overline{w} \\ id(\overline{w}) & \text{otherwise} \end{cases}$$

where $\overline{w} \in sd(w) \xrightarrow{+} w$, and $\overline{w}$ has the minimum sd value among all such nodes.

Proof: From Theorem 1 (page 7) and Theorem 2 (page 8). □

6.2 Outline of the algorithm

I present an abstract version of the algorithm that is almost a direct transliteration of Corollaries 1 and 2. Many of the optimizing steps given in the original paper do not appear in this version. I retain the description in terms of node numbers rather than node names which is used in the original version. The output of the algorithm, however, has to use the node names in the original graph description. The mapping from node numbers to names can be implemented by computing array $vertex$ during the depth-first search where $vertex[i]$ is the name of the node with preorder number i.

I use arrays corresponding to sd_1, sd_2, sd, and id; arrays are shown with square brackets, as in $sd[w]$, to distinguish it from the function value $sd(w)$. Each array is indexed over 1 through N corresponding to the node numbers other than r. One additional array, $bucket$, is introduced to simplify the algorithm, where $bucket[k]$ is the set of nodes whose sd value is k, i.e.

$(w \in bucket[k]) \equiv (sd(w) = k)$.

Programming Notation I use a pseudo notation for programming, appealing to the reader's knowledge of imperative programming; see Misra[3], Section 4.4, for more detailed explanation of the notation.

Comments and assertions are within bold braces. Commands that are within double quotes are explained elsewhere in this document.

The only looping construct is **for** , which has the form

for $x \in S$ **do** f **endfor**

for a finite set or sequence S and variable x. This command executes f for each value of x in S, in arbitrary order if S is a set and in the prescribed order for a sequence. The sequences used in the following program are $N..1$ and $1..N$, denoting decreasing and increasing order of values, respectively, of the associated index[1]. A sequence enclosed within braces denotes the set of of values in that sequence.

A command of the form $x :\in S$, where x is a variable and S a non-empty set denotes a non-deterministic assignment in which any value from set S is assigned to x.

[1] This notation differs slightly from the description in Misra[3].

Immediate Dominators

{ Given is a graph G with $N+1$ nodes, and edges denoted as $v \rightsquigarrow w$.
There is a node r such that $r \overset{+}{\rightsquigarrow} w$, for all w, $w \neq r$. **}**

"do a depth-first search of G: each node gets its preorder number; $r = 0$"**;**

for $k \in \{1..N\}$ **do** $bucket[k] := \{\}$ **endfor;**

{ compute sd in decreasing order of nodes **}**
for $w \in N..1$ **do**
  "compute $sd[w]$" **{** expanded in Section 6.3 **}**
  $bucket[sd[w]] := bucket[sd[w]] \cup \{w\}$
endfor ;

{ compute $\overline{w}$ for all w in arbitrary node order of w **}**
for $k \in \{1..N\}$ **do**
  for $w \in bucket[k]$ **do** **{** $sd[w] = k$ **}**
    $sdwbar := (\min v : v \in k \overset{+}{\rightarrow} w : sd[v])$ **;**
    $\overline{w} :\in \{v \mid v \in k \overset{+}{\rightarrow} w,\ sd[v] = sdwbar\}$ **{** $sdwbar = sd[\overline{w}]$ **}**
  endfor
endfor ;

{ compute id in increasing order of nodes **}**
for $w \in 1..N$ **do**
  if $w = \overline{w}$
    then $id[w] := sd[w]$
    else $id[w] := id[\overline{w}]$ **{** $\overline{w} < w$; so, $id[\overline{w}]$ is known **}**
  endif
endfor

Figure 3: Outline of the Immediate Dominators Algorithm

6.3 Refining "compute $sd[w]$"

Compute $sd[w]$ by $sd[w] := \min(sd_1[w], sd_2[w])$. Computation of $sd_1[w]$ is straightforward, by taking the minimum over all predecessors of w that are smaller than w. To compute $sd_2[w]$, we use an additional data structure, F, that consists of a set of disjoint subtrees of the depth-first search tree. Initially, each node is a singleton subtree in F. As the algorithm proceeds, non-trivial parts of T are added to F, so an invariant is $F \subseteq T$.

Define $root(v)$, for any node v, to be the root of the tree in F of which v is a node; if v is itself the root of a tree in F, then $root(v) = v$. An invariant of the computation of $sd_2[w]$ is: sd has been computed for all and only the non-root nodes of F. Since sd is computed in decreasing node order, we can assert that during the computation of $sd_2[w]$: for any predecessor v of w, where $v > w$, the ancestors of v that are larger than w, as required in Proposition 4 (page 5), are exactly the ones on the path $root(v) \xrightarrow{+} v$.

Initially, $F := \{1..N\}$. After $sd(w)$ is computed, the edge $(parent(w), w)$ is added to F to preserve the invariant, so $root(w) \neq w$.

compute $sd[w]$

{ $F \subseteq T,\ (v \leq w)\ \equiv\ (root(v) = v)$ **}**
$sd_1[w], sd_2[w] := \infty, \infty$**;**
for $v \in pred(w)$ **do** **{** $pred(w)$ is the set of predecessors of w in G **}**
  if $v < w$ **then** $sd_1(w) := \min(sd_1(w), v)$
  else **{** $v > w$, so $root(v) \neq v$ from the invariant **}**
    { compute the smallest $sd[u]$, $u \in root(v) \xrightarrow{+} v$. Also see Section 6.4 **}**
    $sd_2[w] := \min(sd_2[w], (\min\ u : u \in root(v) \xrightarrow{+} v : sd[u]))$**;**
  endif
endfor ;
$sd[w] := \min(sd_1[w], sd_2[w])$**;**
$F := F \cup \{(parent(w), w)\}$

Figure 4: Computation of $sd(w)$ for a node w

6.4 Implementing computation of $sd_2[w]$

The data structure F can be efficiently implemented. The two operations on it after initialization are: (1) $root(v)$ that computes the root of v's tree in F, (2) adding an edge $(parent(w), w)$ to F after the computation of $sd[w]$. Disjoint set union implements these two operations extremely efficiently. The original paper suggests two different implementation strategies: (1) using the simple strategy of path compression only, the algorithm runs in $\mathcal{O}(m\ \log N)$ time, where m is the number of edges and N the nodes in G, and (2) using both path compression and union of balanced trees only, the run time becomes nearly linear in m. Alstrup et. al. [1] modify the given algorithm to run in linear time.

Acknowledgement Keshav Pingali encouraged me to study the beautiful algorithm in the original paper and structure the proofs. I am thankful to Doug McIlroy for spotting an error in an earlier draft.

References

[1] Stephen Alstrup, Dov Harel, Peter W Lauridsen, and Mikkel Thorup. Dominators in linear time. *SIAM Journal on Computing*, 28(6):2117–2132, 1999.

[2] Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1(1):121–141, 1979.

[3] Jayadev Misra. *Effective Theories in Programming Practice*. Morgan & Claypool, 2022.