Lengauer-Tarjan Example

October 16, 2012
**Processing** \( w_{13} \)

Computing \( sdom(13) \) by calling \( eval \) for each predecessor.
Initially \( sdom(13) = 13 \).
Inspecting predecessor \( v = 0 \).
Finding ancestor of \( v = 0 \) with least semi-dominator. Initially \( u = v \).
\( eval \) returns \( u = 0 \).
Predecessor \( v = 0 \) gave \( u = eval(0) = 0 \). Found a better candidate for the semi-dominator: \( sdom(13) = 0 \).
Linking \( w = 13 \) and \( parent(w) = 0 \)
Since \( sdom(13) = parent(13) = 0 \) we know \( idom(13) = sdom(13) = 0 \).
Emptying bucket of 0, which already was empty.

**Processing** \( w_{12} \)

Computing \( sdom(12) \) by calling \( eval \) for each predecessor.
Initially \( sdom(12) = 12 \).
Inspecting predecessor \( v = 9 \).
Finding ancestor of \( v = 9 \) with least semi-dominator. Initially \( u = v \).
\( eval \) returns \( u = 9 \).
Predecessor \( v = 9 \) gave \( u = eval(9) = 9 \). Found a better candidate for the semi-dominator: \( sdom(12) = 9 \).
Linking \( w = 12 \) and \( parent(w) = 9 \)
Since \( sdom(12) = parent(12) = 9 \) we know \( idom(12) = sdom(12) = 9 \).
Emptying bucket of 9, which already was empty.

**Processing** \( w_{11} \)

Computing \( sdom(11) \) by calling \( eval \) for each predecessor.
Initially \( sdom(11) = 11 \).
Inspecting predecessor \( v = 10 \).
Finding ancestor of \( v = 10 \) with least semi-dominator. Initially \( u = v \).
\( eval \) returns \( u = 10 \).
Predecessor \( v = 10 \) gave \( u = eval(10) = 10 \). Found a better candidate for the semi-dominator: \( sdom(11) = 10 \).
Linking \( w = 11 \) and \( parent(w) = 10 \)
Since \( sdom(11) = parent(11) = 10 \) we know \( idom(11) = sdom(11) = 10 \).
Emptying bucket of 10, which already was empty.

**Processing** \( w_{10} \)

Computing \( sdom(10) \) by calling \( eval \) for each predecessor.
Initially \( sdom(10) = 10 \).
Inspecting predecessor \( v = 9 \).
Finding ancestor of \( v = 9 \) with least semi-dominator. Initially \( u = v \).
\textit{eval} returns \( u = 9 \).
Predecessor \( v = 9 \) gave \( u = \text{eval}(9) = 9 \). Found a better candidate for the semi-dominator: \( sdom(10) = 9 \).
Linking \( w = 10 \) and \( parent(w) = 9 \)
Since \( sdom(10) = parent(10) = 9 \) we know \( idom(10) = sdom(10) = 9 \).
Emptying bucket of 9, which already was empty.

\textbf{Processing} \( w_9 \)
Computing \( sdom(9) \) by calling \textit{eval} for each predecessor.
Initially \( sdom(9) = 9 \).
Inspecting predecessor \( v = 1 \).
Finding ancestor of \( v = 1 \) with least semi-dominator. Initially \( u = v \).
\textit{eval} returns \( u = 1 \).
Predecessor \( v = 1 \) gave \( u = \text{eval}(1) = 1 \). Found a better candidate for the semi-dominator: \( sdom(9) = 1 \).
Linking \( w = 9 \) and \( parent(w) = 1 \)
Since \( sdom(9) = parent(9) = 1 \) we know \( idom(9) = sdom(9) = 1 \).
Emptying bucket of 1, which already was empty.

\textbf{Processing} \( w_8 \)
Computing \( sdom(8) \) by calling \textit{eval} for each predecessor.
Initially \( sdom(8) = 8 \).
Inspecting predecessor \( v = 7 \).
Finding ancestor of \( v = 7 \) with least semi-dominator. Initially \( u = v \).
\textit{eval} returns \( u = 7 \).
Predecessor \( v = 7 \) gave \( u = \text{eval}(7) = 7 \). Found a better candidate for the semi-dominator: \( sdom(8) = 7 \).
Inspecting predecessor \( v = 11 \).
Finding ancestor of \( v = 11 \) with least semi-dominator. Initially \( u = v \).
\( v = 11 \) has another ancestor pointer to follow.
\( v = 10 \) has another ancestor pointer to follow.
\textit{eval} found ancestor with smaller semi-dominator: \( u = 10 \).
\( v = 9 \) has another ancestor pointer to follow.
\textit{eval} found ancestor with smaller semi-dominator: \( u = 9 \).
\textit{eval} returns \( u = 9 \).
Predecessor \( v = 11 \) gave \( u = \text{eval}(11) = 9 \). Found a better candidate for the semi-dominator: \( sdom(8) = 1 \).
Linking \( w = 8 \) and \( parent(w) = 7 \)
Adding \( w = 8 \) to bucket of \( sdom(8) = 1 \).
Emptying bucket of 7, which already was empty.
**Processing** $w_7$

Computing $sdom(7)$ by calling `$eval` for each predecessor.
Initially $sdom(7) = 7$.
Inspecting predecessor $v = 3$.
Finding ancestor of $v = 3$ with least semi-dominator. Initially $u = v$.
$eval$ returns $u = 3$.
Predecessor $v = 3$ gave $u = eval(3) = 3$. Found a better candidate for the semi-dominator: $sdom(7) = 3$.
Inspecting predecessor $v = 12$.
Finding ancestor of $v = 12$ with least semi-dominator. Initially $u = v$.
$v = 12$ has another ancestor pointer to follow.
$v = 9$ has another ancestor pointer to follow.
$eval$ found ancestor with smaller semi-dominator: $u = 9$.
$eval$ returns $u = 9$.
Predecessor $v = 12$ gave $u = eval(12) = 9$. Found a better candidate for the semi-dominator: $sdom(7) = 1$.
Linking $w = 7$ and $parent(w) = 3$
Adding $w = 7$ to bucket of $sdom(7) = 1$.
Emptying bucket of 3, which already was empty.

**Processing** $w_6$

Computing $sdom(6)$ by calling `$eval` for each predecessor.
Initially $sdom(6) = 6$.
Inspecting predecessor $v = 5$.
Finding ancestor of $v = 5$ with least semi-dominator. Initially $u = v$.
$eval$ returns $u = 5$.
Predecessor $v = 5$ gave $u = eval(5) = 5$. Found a better candidate for the semi-dominator: $sdom(6) = 5$.
Inspecting predecessor $v = 8$.
Finding ancestor of $v = 8$ with least semi-dominator. Initially $u = v$.
$v = 8$ has another ancestor pointer to follow.
$v = 7$ has another ancestor pointer to follow.
$eval$ returns $u = 8$.
Predecessor $v = 8$ gave $u = eval(8) = 8$. Found a better candidate for the semi-dominator: $sdom(6) = 1$.
Linking $w = 6$ and $parent(w) = 5$
Adding $w = 6$ to bucket of $sdom(6) = 1$.
Emptying bucket of 5, which already was empty.
**Processing** $w_5$

Comparing $sdom(5)$ by calling $eval$ for each predecessor.
Initially $sdom(5) = 5$.
Inspecting predecessor $v = 4$.
Finding ancestor of $v = 4$ with least semi-dominator. Initially $u = v$.
$eval$ returns $u = 4$.
Predecessor $v = 4$ gave $u = eval(4) = 4$. Found a better candidate for the semi-dominator: $sdom(5) = 4$.
Linking $w = 5$ and $parent(w) = 4$
Since $sdom(5) = parent(5) = 4$ we know $idom(5) = sdom(5) = 4$.
Emptying bucket of 4, which already was empty.

**Processing** $w_4$

Comparing $sdom(4)$ by calling $eval$ for each predecessor.
Initially $sdom(4) = 4$.
Inspecting predecessor $v = 3$.
Finding ancestor of $v = 3$ with least semi-dominator. Initially $u = v$.
$eval$ returns $u = 3$.
Predecessor $v = 3$ gave $u = eval(3) = 3$. Found a better candidate for the semi-dominator: $sdom(4) = 3$.
Inspecting predecessor $v = 13$.
Finding ancestor of $v = 13$ with least semi-dominator. Initially $u = v$.
$v = 13$ has another ancestor pointer to follow.
$eval$ returns $u = 13$.
Predecessor $v = 13$ gave $u = eval(13) = 13$. Found a better candidate for the semi-dominator: $sdom(4) = 0$.
Linking $w = 4$ and $parent(w) = 3$
Adding $w = 4$ to bucket of $sdom(4) = 0$.
Emptying bucket of 3, which already was empty.

**Processing** $w_3$

Comparing $sdom(3)$ by calling $eval$ for each predecessor.
Initially $sdom(3) = 3$.
Inspecting predecessor $v = 2$.
Finding ancestor of $v = 2$ with least semi-dominator. Initially $u = v$.
$eval$ returns $u = 2$.
Predecessor $v = 2$ gave $u = eval(2) = 2$. Found a better candidate for the semi-dominator: $sdom(3) = 2$.
Linking $w = 3$ and $parent(w) = 2$
Since $sdom(3) = parent(3) = 2$ we know $idom(3) = sdom(3) = 2$.
Emptying bucket of 2, which already was empty.
**Processing** \( w_2 \)

Computing \( sdom(2) \) by calling \( eval \) for each predecessor.
Initially \( sdom(2) = 2 \).
Inspecting predecessor \( v = 1 \).
Finding ancestor of \( v = 1 \) with least semi-dominator. Initially \( u = v \).
\( eval \) returns \( u = 1 \).
Predecessor \( v = 1 \) gave \( u = eval(1) = 1 \). Found a better candidate for the semi-dominator: \( sdom(2) = 1 \).
Linking \( w = 2 \) and \( parent(w) = 1 \)
Since \( sdom(2) = parent(2) = 1 \) we know \( idom(2) = sdom(2) = 1 \).
Emptying bucket of 1.
Inspecting \( v = 1 \).
Finding ancestor of \( v = 6 \) with least semi-dominator. Initially \( u = v \).
\( v = 6 \) has another ancestor pointer to follow.
\( v = 5 \) has another ancestor pointer to follow.
\( v = 4 \) has another ancestor pointer to follow.
\( eval \) found ancestor with smaller semi-dominator: \( u = 4 \).
\( v = 3 \) has another ancestor pointer to follow.
\( v = 2 \) has another ancestor pointer to follow.
\( eval \) returns \( u = 4 \). Found ancestor with smaller semi-dominator: \( u = 4 \).
Inspecting \( v = 6 \).
Finding ancestor of \( v = 7 \) with least semi-dominator. Initially \( u = v \).
\( v = 7 \) has another ancestor pointer to follow.
\( v = 3 \) has another ancestor pointer to follow.
\( v = 2 \) has another ancestor pointer to follow.
\( eval \) returns \( u = 7 \). The semi-dominator is also the immediate dominator.
Inspecting \( v = 7 \).
Finding ancestor of \( v = 8 \) with least semi-dominator. Initially \( u = v \).
\( v = 8 \) has another ancestor pointer to follow.
\( v = 7 \) has another ancestor pointer to follow.
\( v = 3 \) has another ancestor pointer to follow.
\( v = 2 \) has another ancestor pointer to follow.
\( eval \) returns \( u = 8 \). The semi-dominator is also the immediate dominator.

**Processing** \( w_1 \)

Computing \( sdom(1) \) by calling \( eval \) for each predecessor.
Initially \( sdom(1) = 1 \).
Inspecting predecessor \( v = 0 \).
Finding ancestor of \( v = 0 \) with least semi-dominator. Initially \( u = v \).
\( eval \) returns \( u = 0 \).
Predecessor \( v = 0 \) gave \( u = eval(0) = 0 \). Found a better candidate for the semi-dominator: \( sdom(1) = 0 \).
Linking $w = 1$ and $parent(w) = 0$
Since $sdom(1) = parent(1) = 0$ we know $idom(1) = sdom(1) = 0$.
Emptying bucket of 0.
Inspecting $v = 0$.
Finding ancestor of $v = 4$ with least semi-dominator. Initially $u = v$.
$v = 4$ has another ancestor pointer to follow.
$v = 3$ has another ancestor pointer to follow.
$v = 2$ has another ancestor pointer to follow.
$v = 1$ has another ancestor pointer to follow.
$eval$ returns $u = 4$. The semi-dominator is also the immediate dominator.

**Step 4**

Computing final values of $idom$.
Adds $w = 1$ as child of 0.
Adds $w = 2$ as child of 1.
Adds $w = 3$ as child of 2.
Adds $w = 4$ as child of 0.
Adds $w = 5$ as child of 4.
Now we can set $idom(6) = 0$.
Adds $w = 6$ as child of 0.
Adds $w = 7$ as child of 1.
Adds $w = 8$ as child of 1.
Adds $w = 9$ as child of 1.
Adds $w = 10$ as child of 9.
Adds $w = 11$ as child of 10.
Adds $w = 12$ as child of 9.
Adds $w = 13$ as child of 0.