Properties of Depth-First Search

First edge into each vertex = tree edge

Tree edges define spanning forest; trees rooted at search start vertices

Previsit order = preorder on tree (discovery order)

Post visit order = postorder on tree (finishing order)

\[ v \text{ an ancestor of } w \iff \text{pre}(v) \leq \text{pre}(w) \quad \& \quad \text{post}(v) \geq \text{post}(w) \]

Undirected graph: direct edges along search direction:

- each non-tree edge leads from a descendant to an ancestor

Digraph: \((v, w)\) an edge implies \(v\) and \(w\) are related in depth-first spanning forest

or \(\text{pre}(v) > \text{pre}(w) \quad \& \quad \text{post}(v) > \text{post}(w)\)
would be traversed from $v$ before $w$ is reached, hence a tree edge.
Path lemma:

undigraph: Any path from $v$ to $w$ contains a common ancestor of $v$ and $w$.

digraph: Any path from $v$ to $w$ with $\text{pre}(v) < \text{pre}(w)$ (or $\text{post}(v) < \text{post}(w)$) contains a common ancestor of $v$ and $w$.

Diagram:

- $u$
- $v$
- $w$
- $nca(v,w)$

common ancestor = first vertex on the path that is not a descendant of $u$

(digraph) = first vertex on the path that is $\geq v$ in postorder
Cut vertex:

root with $\geq$ children
(no cross edges)

non-root with child $w$ with no edges from a
descendant of $w$ to a proper ancestor of $v$ (iff $\text{low}(w) \geq v$)
Bipolar order: edges directed so acyclic,
  $s$ is only source
  $t$ is only sink

Bipolar order = topological order with one source, one sink

$G$ has a bipolar order iff $G \cup E(s,t)$ has no cut vertices

Bipolar order algorithm:

1. DFS from $s$, starting along $(s,t)$, compute
   pre, low, parents in DFS tree

2. Visit vertices in preorder, constructing list.
   
   $+$ = after current vertex
   $-$ = before current vertex

   if $\text{sign}(low(v)) = +$, insert $v$ after $p(v)$; $\text{sign}(p(v)) = -$
   else insert $v$ before $p(v)$; $\text{sign}(p(v)) = +$
Dominator: Digraph with start vertex $s$.

$v$ dominates $w$ if all paths from $s$ to $w$ contain $v$.

$v$ immediately dominates $w$ if all dominators of $w$ except $v$ also dominate $v$.

The immediate dominators define a tree rooted at $s$.

Goal: Compute the dominator tree.

Applications: Global code optimization: move code out of an inner loop to a dominator.
Dominator's algorithm uses preorder, r-preorder, evaluation of path functions on trees (extension of set union)

Vertices numbered in preorder

\[ \text{sdom}(w) = \min \{ v | \text{there is a path from } v \text{ to } w \text{ containing only vertices } \geq w \text{ (except } v) \} \]

Candidates for \( \text{sdom}(w) \)

\[ \text{sdom}(w) = \min \{ v / (v,w) \text{ with } v < w \} \cup \{ \text{sdom}(w) | u > w \text{ and } (v,w) \text{ with } u \text{ an ancestor of } v \} \]
Problem: must compute minima of $sdom(n)$ along tree paths (e.g. (1), (2))
w.r.t. $u$: $sdm(u)$ minimum such that $u$ is a proper descendant of $sdm(w)$ and an ancestor of $w$

\[
 idom(w) = \begin{cases} 
 sdom(w) & \text{if } sdom(w) = sdom(u) \\
 idom(u) & \text{otherwise}
\end{cases}
\]

Also needs computation of mins of $sdm$ along tree paths.

Allows computation of $idom$ in preorder.

3 passes: preorder (DFS), reverse preorder ($sdm$), preorder ($idom$)
Computation of minima along tree paths: path compression

Tricky to allow union by rank

```
  10
   /\  
   8  
   /\  
  12  
   /\  
  4   
   /\  
  2   
```