# 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)

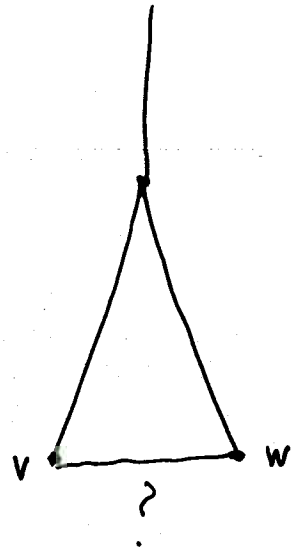
Postvisit order = postorder on tree (finishing order)

v an ancestor of w iff $pre(v) \leq pre(w)$ & $post(v) \geq post(w)$

Undirected graph: direct edges along search direction:

each nontree 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 $pre(v) > pre(w)$ (& $post(v) > post(w)$)



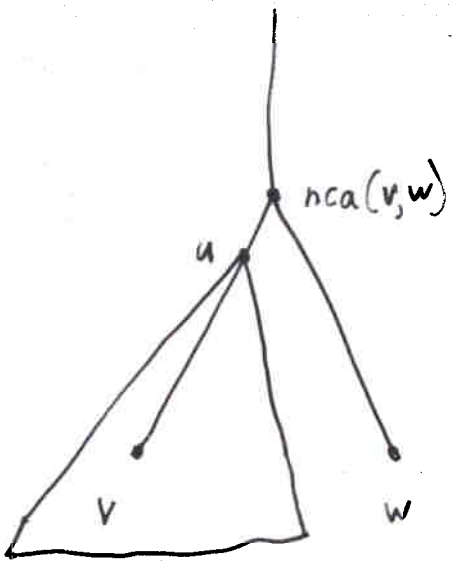

would be traversed from v
before w is reached,
hence a tree edge


(same argument)

Path lemma:

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

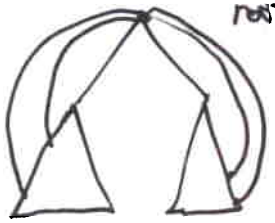
digraph: Any path from v to w with pre(v) < pre(w) (or post(v) < post(w)) contains a common ancestor of v and w



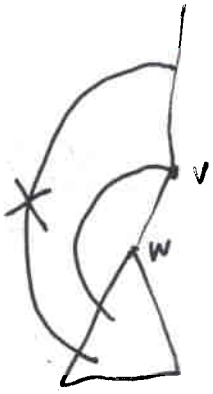

common ancestor = first vertex on the path that is not a descendant of u

(digraph) = first vertex on the path that is ≥ u in postorder

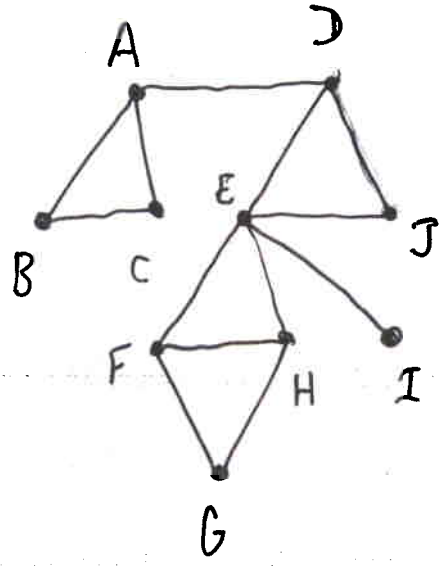
Cut vertex:



root with ≥ children
(no cross edges)




non root with child w with no edges from a descendant of w to a proper ancestor of v (iff low(w) ≥ v)


A
D
E
B
C
J
F
H
I
G

Bipolar order: edges directed so acyclic,
s = only source
t = only sink

Bipolar order = topological order with one source, one sink

G has a bipolar order iff $G \cup \{(s,t)\}$ 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

   $\bar{s}, t$

   if sign(low(v)) = +, insert v after p(v); sign(p(v)) = -
   else insert v before p(v); sign(p(v)) = +

Dominators: Digraph with start vertex s.

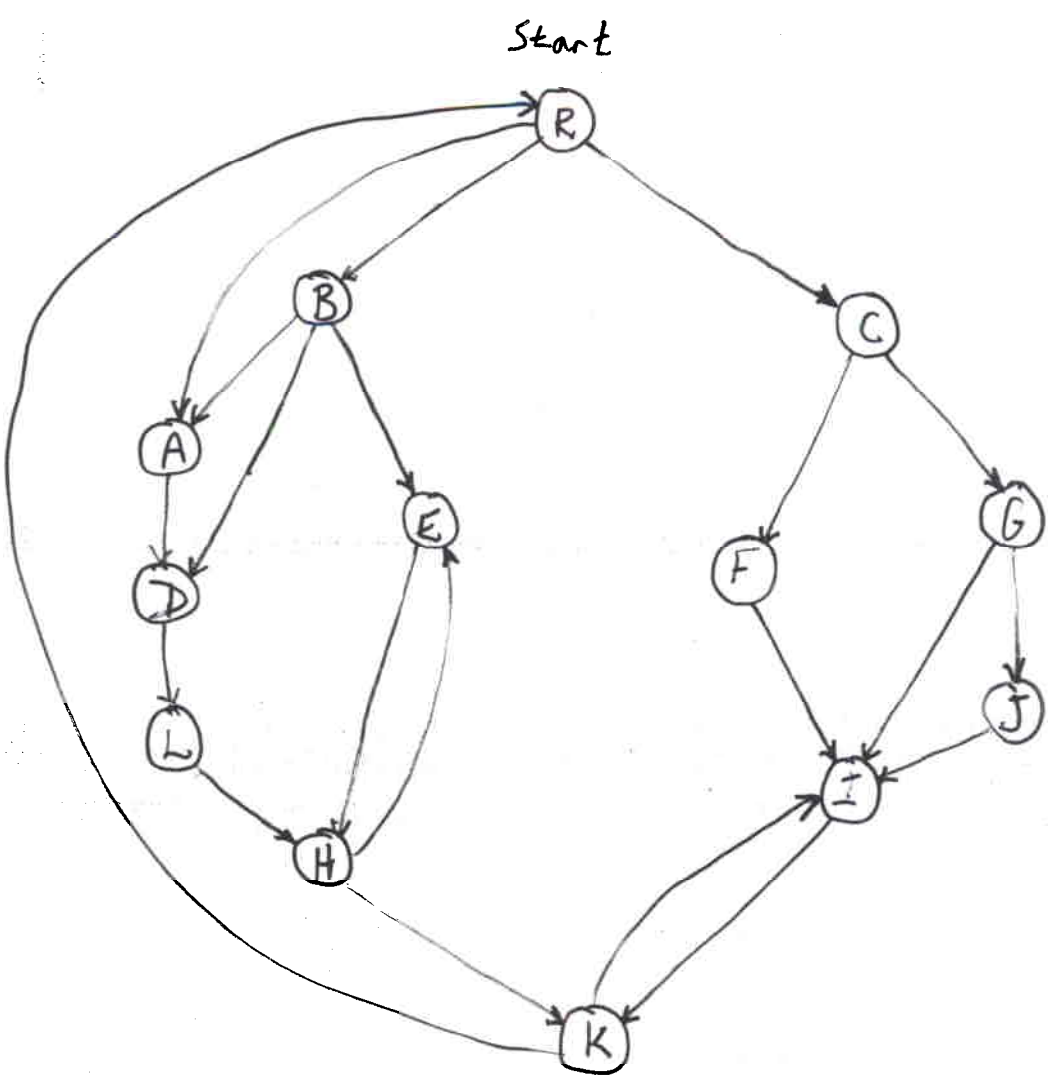
v dominates w if all paths from s to w contain v.

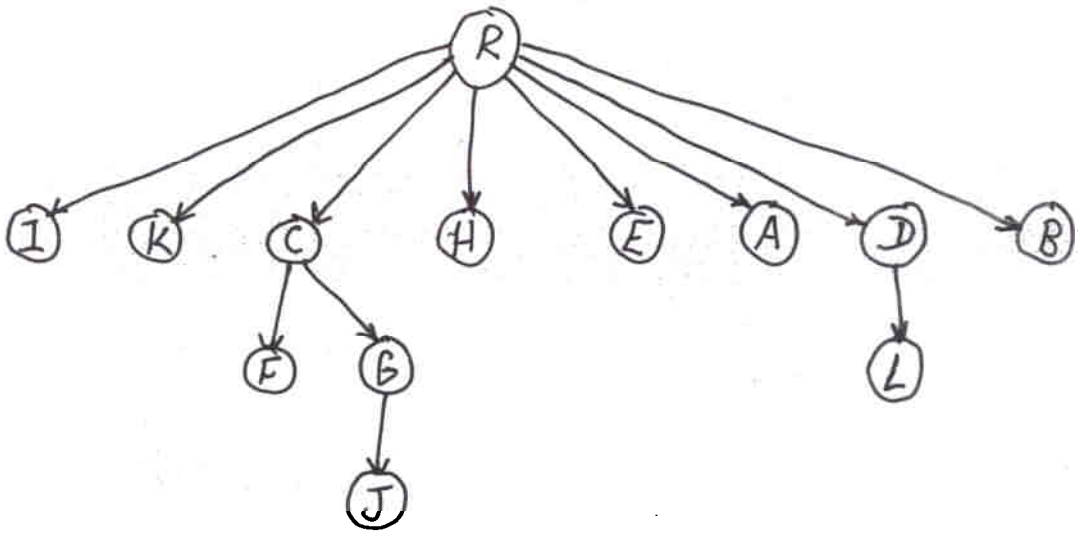
v immediately dominates w if all dominators of w except v also dominate v.

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.

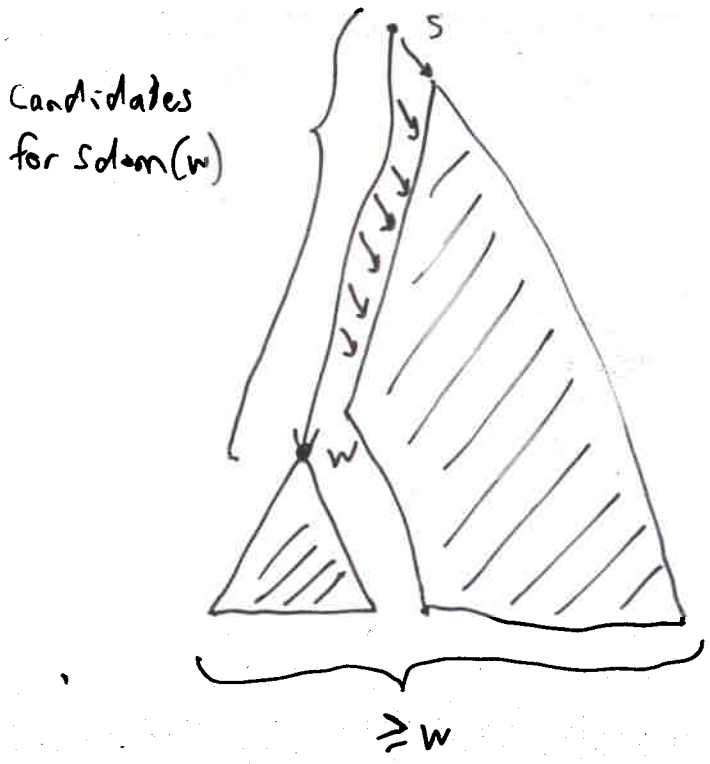

Start
R
B
C
A
E
G
F
D
J
L
I
H
K


R
I
K
C
H
E
A
D
B
F
G
L
J

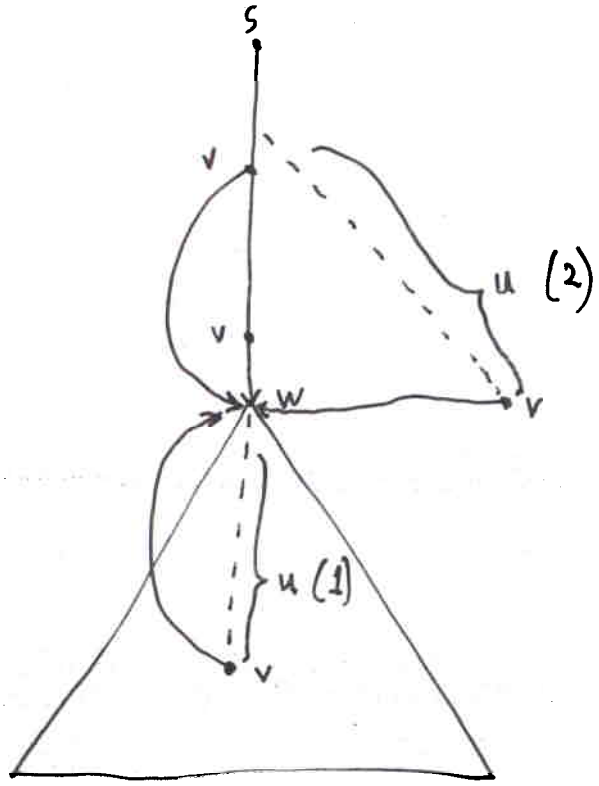
Dominators algorithm uses preorder, r-preorder, evaluation of path functions on trees (extension of set union)

Vertices numbered in preorder

$sdom(w) = \min\{v \mid$ there is a path from $v$ to $w$ containing only vertices $\geq w$ (except $v$)$\}$



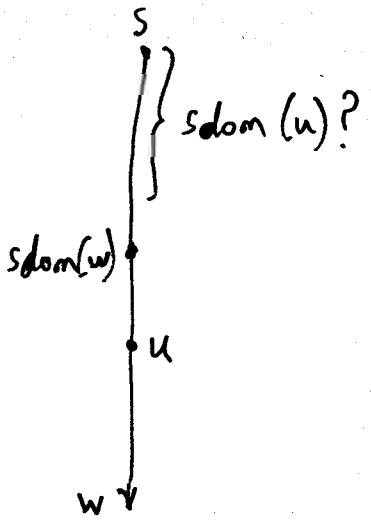

$sdom(w) = \min\{v \mid (v,w)$ with $v < w\} \cup$
$\{sdom(u) \mid u > w$ and $(v,w)$ with $u$ an ancestor of $v\}$




Problem: must compute minima of sdom(u) along tree paths (e.g (1), (2))

$w \neq r$ $u$: sdom($u$) minimum such that $u$ is a proper descendant of sdom($w$) and an ancestor of $w$

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




Also needs computation of mins of sdom along tree paths

Allows computation of idom in preorder

3 passes: preorder (DFS), reverse preorder (sdom), preorder (idom)

Computation of minima along tree paths:
path compression

Trickery to allow union by rank