COS 423 Lecture 16
Dominators in Digraphs

Flowgraph: A directed graph with a start vertex s such that every vertex is reachable from s

Vertex v *dominates* vertex w if $v \neq w$ and v is on every path from s to w.

Domination is *anti-symmetric*: if v dominates w, then w does not dominate v.

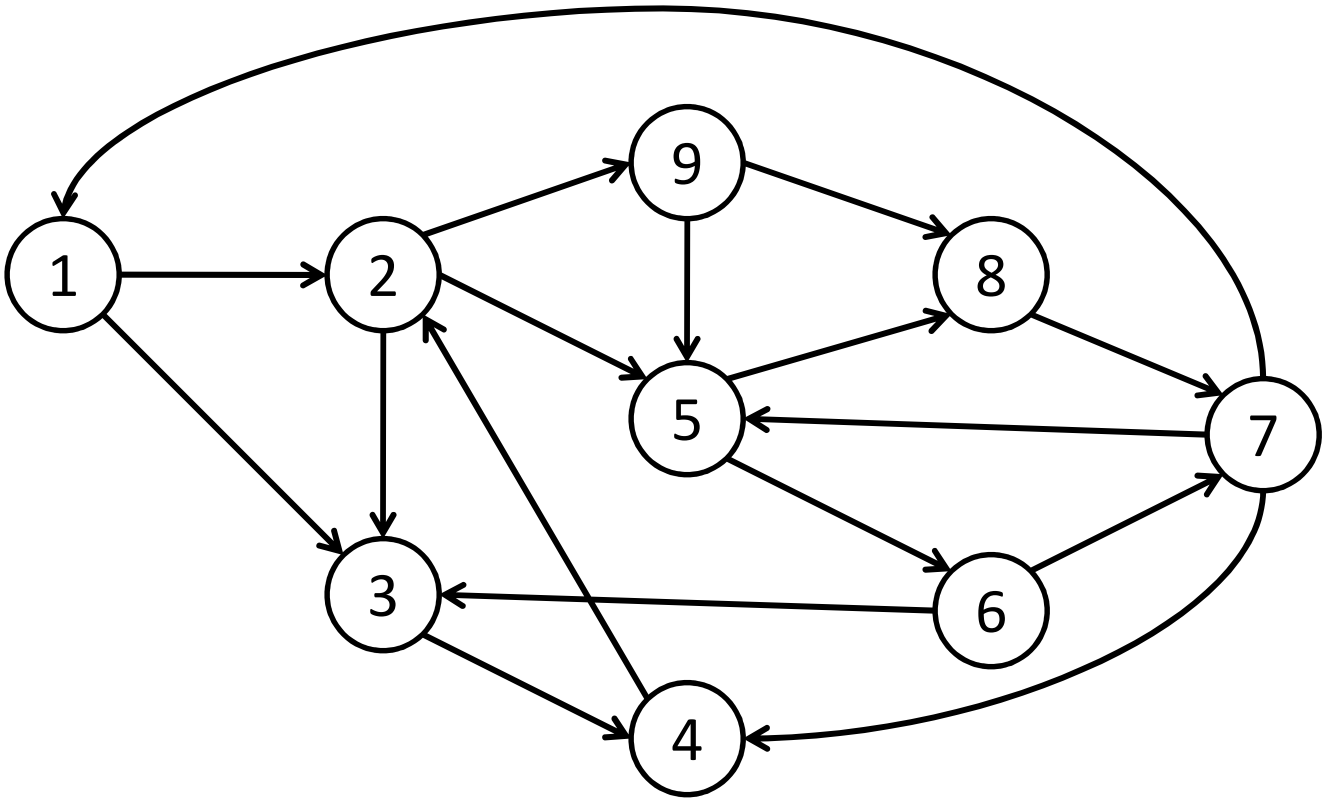
Domination is *transitive*: if u dominates v and v dominates w, then u dominates w.

Domination is *complete*: if both u and v dominate w, then either u dominates v or v dominates w

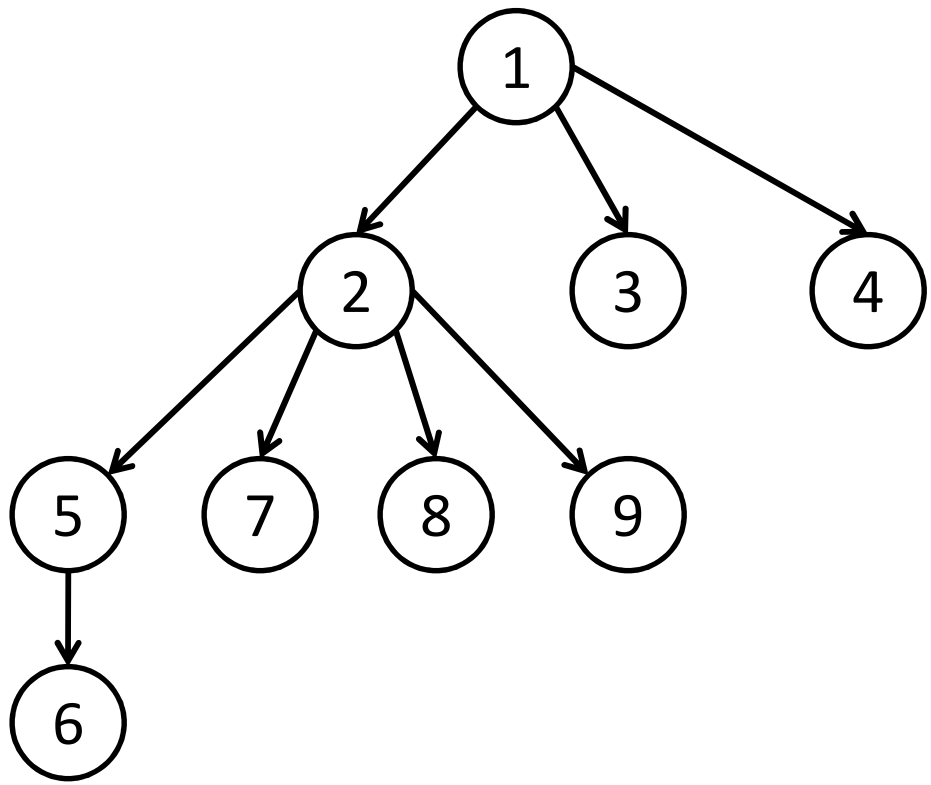
Anti-symmetry, transitivity, and completeness imply that the dominators of any vertex w are totally ordered by domination. Thus there is a vertex v called the *immediate dominator* of w, denoted by *idom*(*w*), that dominates *w* and is dominated by all other dominators of *w*.

The immediate dominators define a tree *D* rooted at *r* such that *idom*(*w*) is the parent of *w* in *D*. Vertex *v* dominates *w* iff *v* is a proper ancestor of *w* in *D*.

Flow graph



Dominator tree



Goal: given a flowgraph $G = (V, E, s)$, find its dominator tree

Applications

Global code optimization: Movement of code to a dominating program block to reduce redundant computation

Circuit testing: Identification of pairs of equivalent line faults.

Theoretical biology: food web analysis

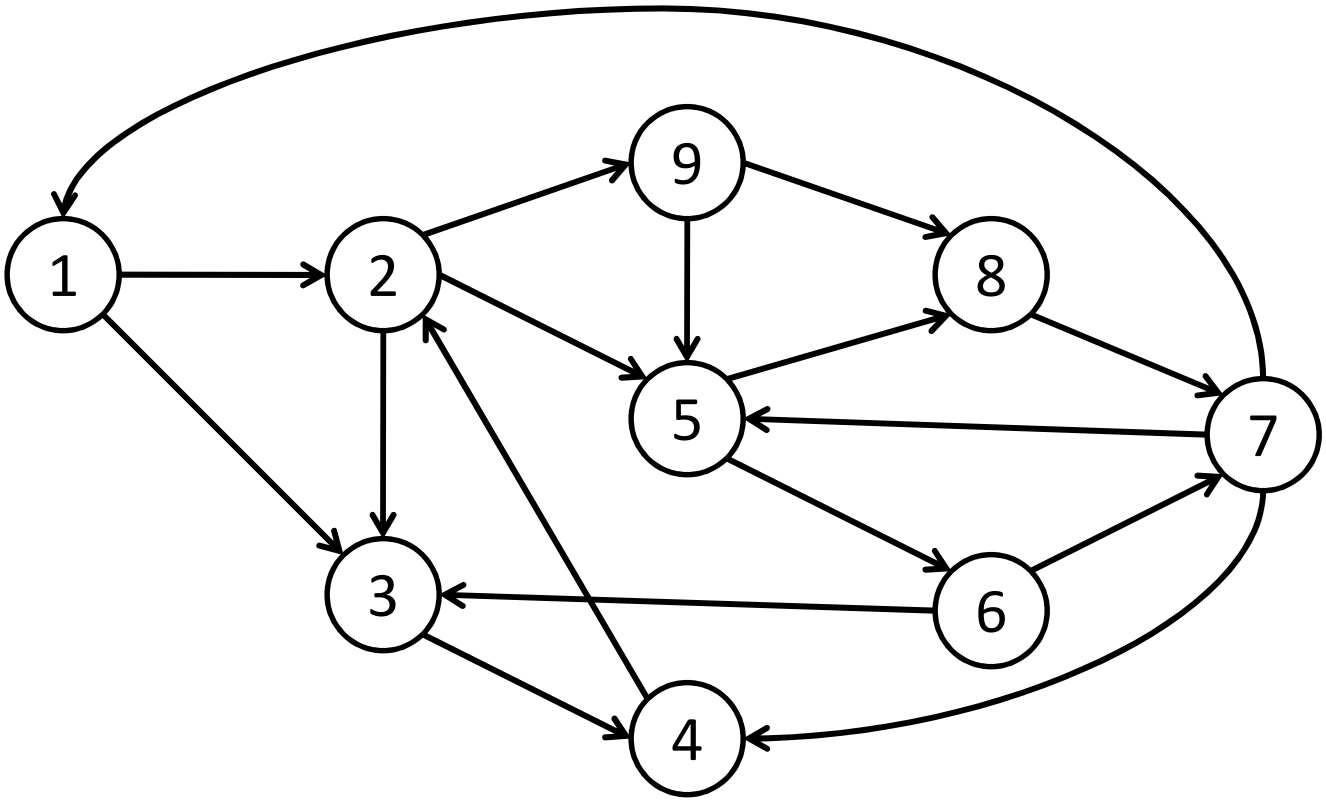
We assume $n > 1$. Since $m \geq n - 1$, $n = O(m)$ and $m > 0$.

Naïve algorithm: For each vertex $v \neq s$, delete v and find all vertices still reachable from s. Vertex v dominates all unreached vertices.

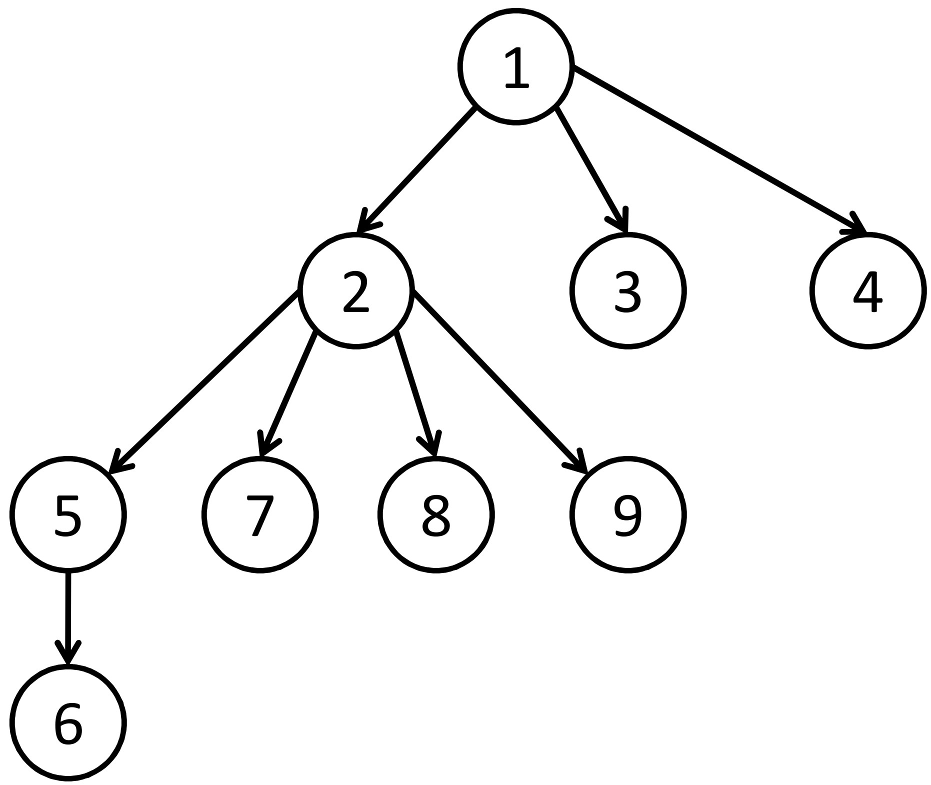
Running time = $O(nm)$

Delete 2: 5, 6, 7,8, 9 unreachable

Delete 5: 6 unreachable



Dominator tree



Tree update algorithm (less naïve but no faster):

Let D be any spanning tree rooted at s

$p(x)$ = parent of x

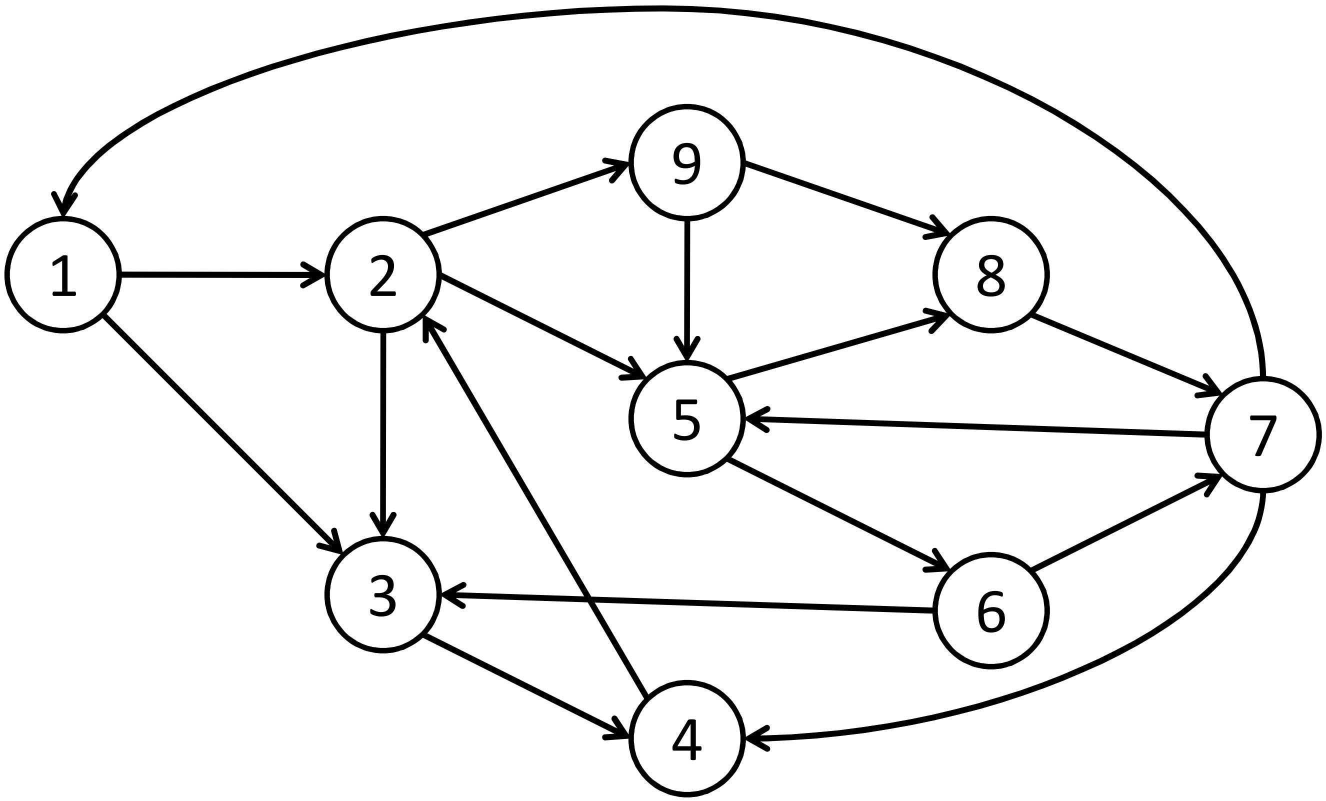
$nca(v, w)$ = nearest common ancestor of v, w

If for every arc (v, w), $nca(v, w)$ is either w or $p(w)$, stop. Otherwise, choose an arc (v, w) such that $u = nca(v, w)$ is neither w nor $p(w)$, replace $p(w)$ by u, and repeat.

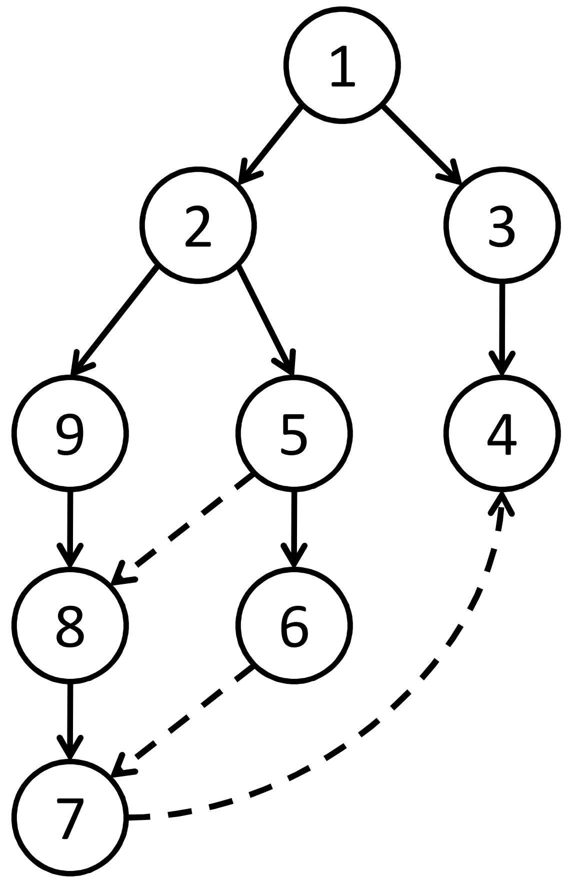
Can represent D with just parent pointers. Each test of an arc takes O(n) time, each update takes O(1) time and reduces the depth of at least one node by at least 1 → O(n^3m) time

Can reduce time to O(n^2m) by careful choice of arcs to test: fast in practice on small graphs

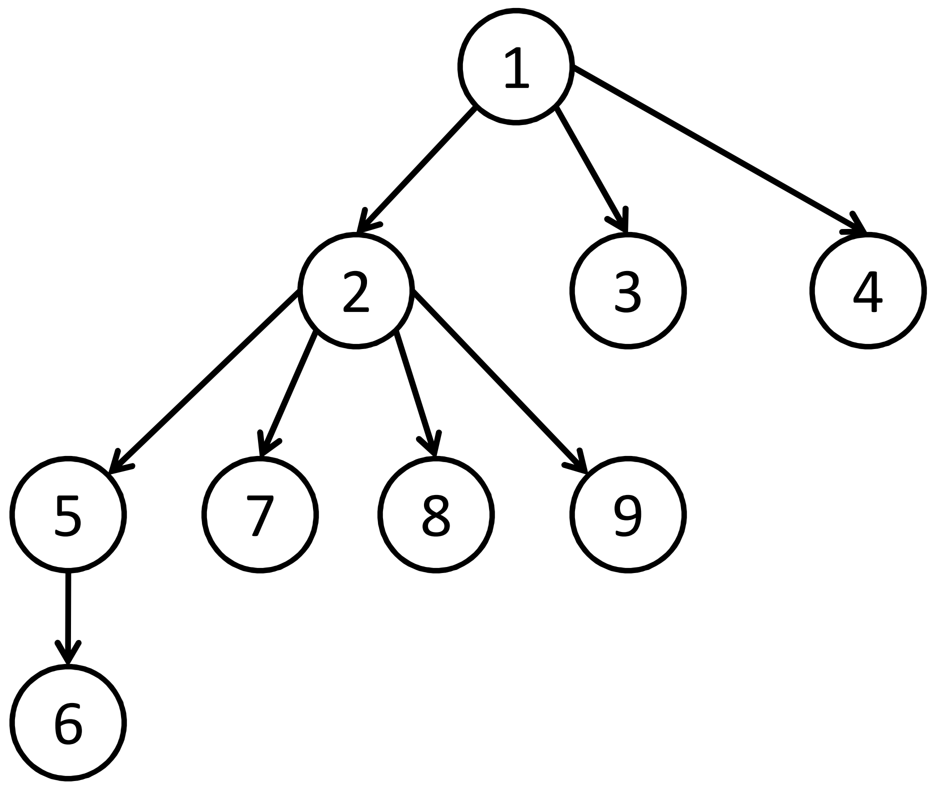
Can reduce time to O(nm) by careful choice of arcs to test and representation of D by child sets as well as parent pointers


1
2
3
4
5
6
7
8
9

BFS tree



Dominator tree



Finding dominators faster?

O(m) is possible

O($m\alpha(n, \lceil m/n \rceil)$) is practical but a little complicated

Here: an O($m\lg n$)-time algorithm that uses DFS + finding minima on paths in the DFS tree

We need a better way to characterize immediate dominators

Do a DFS to form a DFS tree T rooted at s. Let $p(v)$ be the parent of v in T, $nca(v, w)$ the nearest common ancestor of v, w. Order the vertices in preorder.

Let $sdom(v)$, the *semi-dominator* of v, be the smallest vertex u such that there is a path from u to v all of whose vertices except u are no smaller than v

Let $v \neq s$.

$idom(v)$ is a proper ancestor of v in T

Since $p(v)$ is a candidate for $sdom(v)$, $sdom(v) <_{pre} v$

Let P be a path from $sdom(v)$ to v all of whose vertices excluding $sdom(v)$ are no smaller than v

$sdom(v)$ is a proper ancestor of v by the preorder lemma (Lecture 14)

P avoids all ancestors of v that are not ancestors of $sdom(v)$; thus $idom(v)$ is an ancestor of $sdom(v)$

Let $rdom(v)$, the *relative dominator* of v, be a vertex $x \neq sdom(v)$ on the path in T from $sdom(v)$ to v such that $sdom(x)$ is minimum (break a tie arbitrarily)

Dominators Lemma: If $\mathrm{rdom}(v) = v$, then $idom(v) = sdom(v)$. Also, $idom(v) = idom(rdom(v))$

Proof: Suppose $sdom(v)$ does dominate v. Let P be a path from s to v that avoids $sdom(v)$, let x be the last vertex on P less than $sdom(v)$, and let y be the minimum vertex after x on P. Then x is a candidate for $sdom(y)$, so $sdom(y) <_{pre} sdom(v) <_{pre} y$. But y is an ancestor of v by the preorder lemma, which implies that y is a candidate for $rdom(v)$. Since $sdom(y) <_{pre} sdom(v)$, $rdom(v) \neq v$. This gives the first part of the lemma.

Proof (cont.): A path from *s* to *rdom*(*v*) can be extended to *v* by adding the tree path from *rdom*(*v*) to *v*. It follows that no proper descendant of *idom*(*rdom*(*v*)) dominates *v*. Suppose *idom*(*rdom*(*v*)) does not dominate *v*. Let *P* be a path from *s* to *v* that avoids *idom*(*rdom*(*v*)), let *x* be the last vertex on *P* less than *idom*(*rdom*(*v*)), and let *y* be the minimum vertex after *x* on *P*. Then *x* is a candidate for *sdom*(*y*), so $sdom(y) <_{pre} idom(rdom(v)) <_{pre} y$. But *y* is an ancestor of *v* by the preorder lemma.

Proof(cont.): If y were an ancestor of $rdom(v)$, then $idom(rdom(v))$ would not dominate $rdom(v)$; thus y is a proper descendant of $rdom(v)$. But then y is a candidate for $rdom(v)$, which implies $sdom(rdom(v)) \leq_{pre} sdom(y) <_{pre} idom(rdom(v))$, and again $idom(rdom(v))$ cannot dominate $rdom(v)$, a contradiction. This gives the second part of the lemma.

Dominators algorithm

Compute *sdom*(v) for every vertex $v \neq s$.

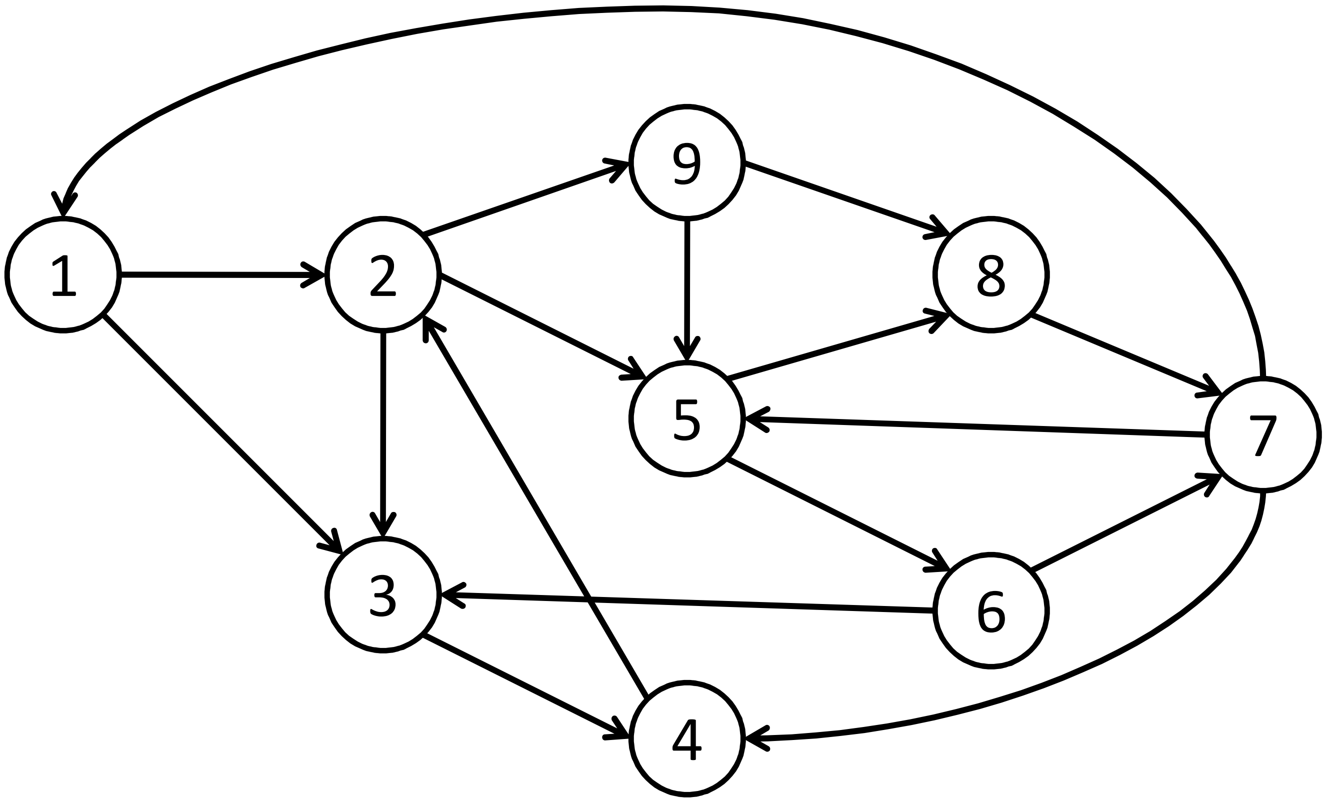
Compute *rdom*(v) for every vertex $v \neq s$.

Set *idom*(s) = null. Visit vertices $v \neq s$ in an order such that $p(v)$ is visited before v, e.g. preorder

visit(v):

if *rdom*(v) = v **then** *idom*(v) $\leftarrow$ *sdom*(v)

else *idom*(v) $\leftarrow$ *idom*(*rdom*(v))


1
2
3
4
5
6
7
8
9

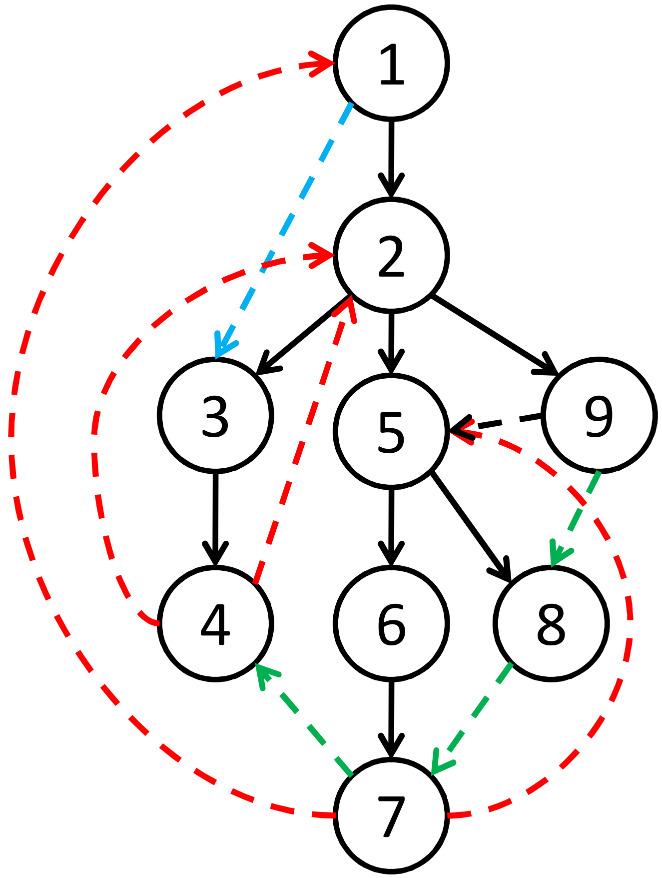
DFS tree and non-tree arcs

tree arcs

forward arcs

cross arcs

back arcs



Semi-dominators

9: 2

8: 2

7: 2

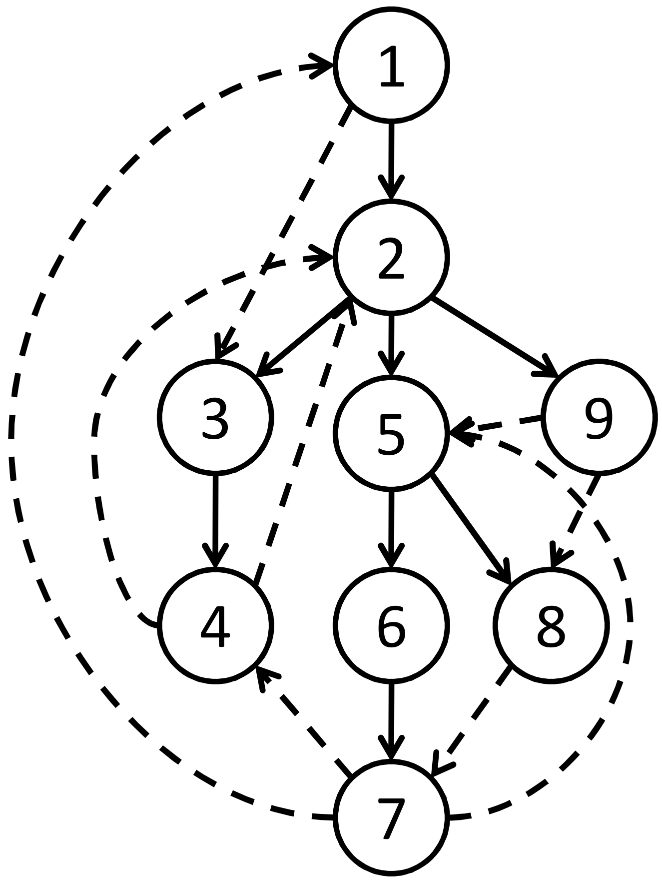
6: 5

5: 2

4: 2

3: 1

2: 1



Relative dominators

9: 2, 9

8: 2, 5

7: 2, 5

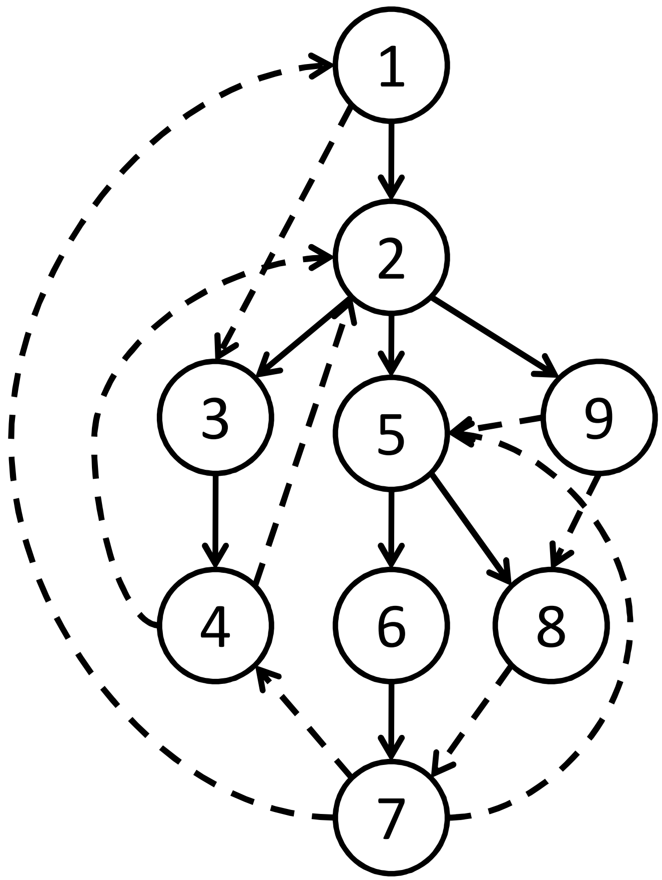
6: 5, 6

5: 2, 5

4: 2, 3

3: 1, 2

2: 1, 2



Immediate dominators

9: 2, 9, 2

8: 2, 5, 2

7: 2, 5, 2

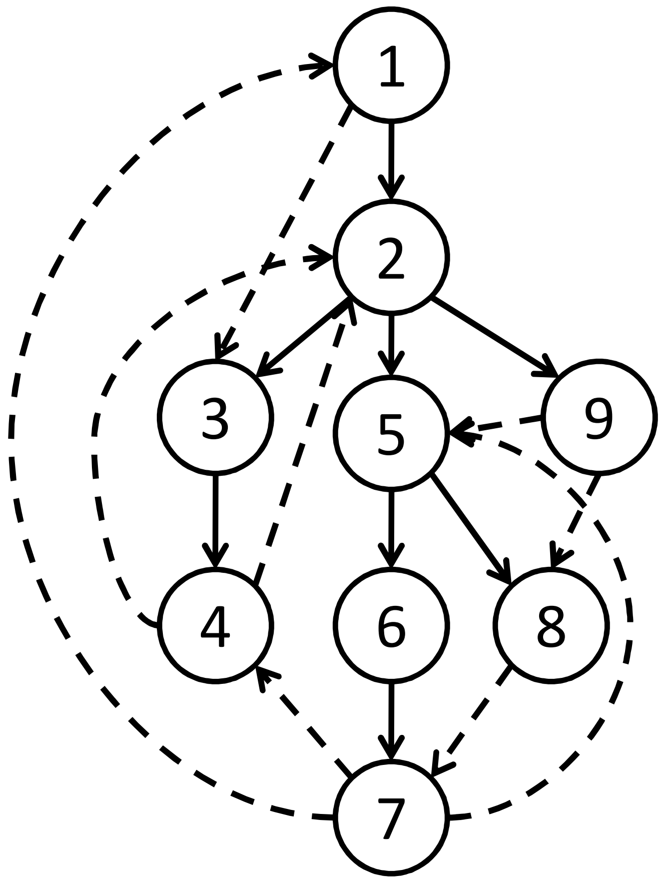
6: 5, 6, 5

5: 2, 5, 2

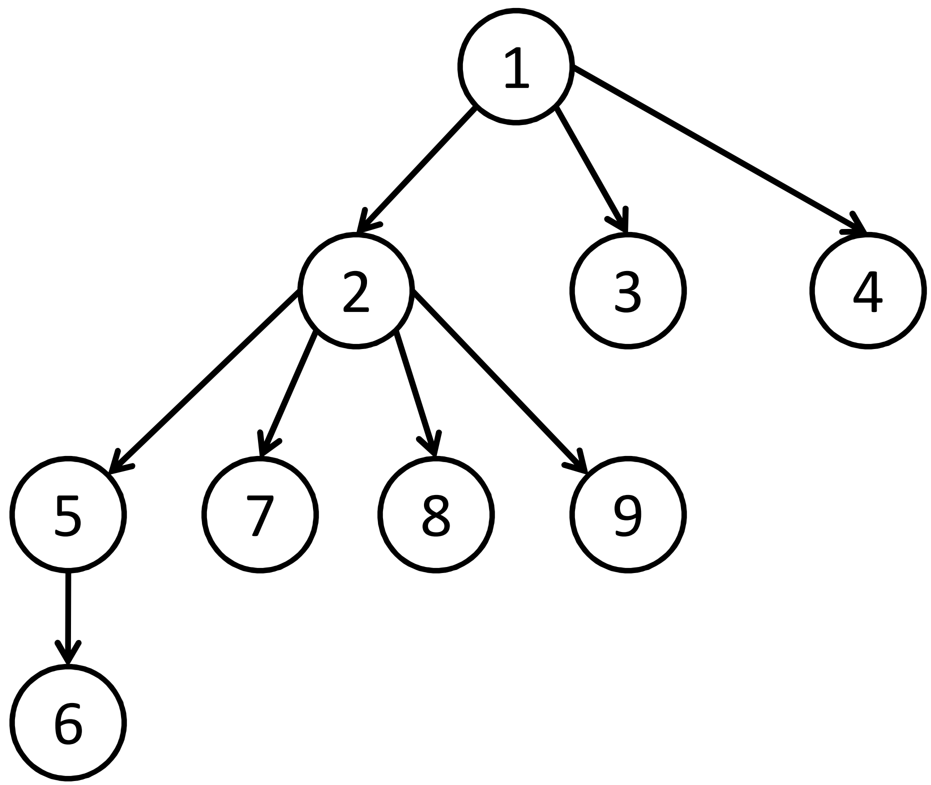
4: 2, 3, 1

3: 1, 2, 1

2: 1, 2, 1



Dominator tree



Correctness: From the dominators lemma; if $rdom(v) \neq v$, then $rdom(v)$ is a proper ancestor of v, hence visited before v

How to compute semi-dominators and relative dominators?

The relative dominators are path-minima on T, with semi-dominators as weights

The computation of semi-dominators can also be done by finding path minima on *T*

Indeed we can compute both semi-dominators and relative dominators in one integrated path minima computation

For an arc (u, v), let $z = nca(u, v)$

If $u = z$, let $r(u, v) = u$.

If $u \neq z$, let $r(u, v)$ be a vertex $x \neq z$ on the path in T from z to u such that $sdom(x)$ is minimum (break a tie arbitrarily)

Lemma: $sdom(v) = \min_{pre}\{r(u, v) \mid (u, v) \in E\}$

Proof: Exercise

This lemma allows us to compute semi-dominators in reverse preorder from path minima of known or previously computed values: If (u, v) is an arc such that u is not an ancestor of v, and $x \neq nca(u, v)$ is on the path in T from $nca(u, v)$ to u, then $x >_{pre} v$, since $x \leq_{pre} v$ implies x is an ancestor of v.

We visit the vertices in reverse preorder, maintaining a compressed version of the part of D visited so far: all $(p(v), v)$ with v visited.

Computation of semi-dominators

for $v \in V$ **do** $a(v) \leftarrow$ null;

for $v \in V - s$ in reverse preorder **do**

$\{sdom(v) \leftarrow \min_{pre}\{sfind(u) \mid (u, v) \in E\};$

$a(v) \leftarrow p(v);\ pmin(v) \leftarrow sdom(v)\}$

$a(v)$: parent of v in compressed forest

$pmin(v)$: path min of v in compressed forest

```
sfind(x):
    if a(x) = null then return x
    else {if a(a(x)) ≠ null then
            {pmin(x) ← min_pre{pmin(x), sfind(a(x)))};
             a(x) ← a(a(x)};
          return pmin(x)}
```

Computation of semi-dominators and relative dominators with optimization

for $v \in V$ **do** $\{a(v) \leftarrow \text{null}; R(v) \leftarrow \{\,\}\};$

for $v \in V - s$ in reverse preorder **do**

$\{$**for** $u \in R(v)$ **do** $rdom(u) \leftarrow sfind(u);$

$sdom(v) \leftarrow \min_{pre}\{sfind(u) \mid (u, v) \in E\};$

$a(v) \leftarrow p(v); pmin(v) \leftarrow sdom(v);$

if $p(v) = sdom(v)$ **then** $rdom(v) \leftarrow v$ **else**

$R(sdom(v)) \leftarrow R(sdom(v)) \cup \{v\}\};$

for $u \in R(s)$ **do** $rdom(u) \leftarrow sfind(u)$

3-pass dominators algorithm

Do a depth-first search. Number vertices in preorder and build DFS tree

Compute *sdom* and *rdom* by visiting the vertices in reverse preorder

Compute *idom* by visiting the vertces in preorder

Running time = $O(m \lg n)$: path compression with naïve linking

Faster Versions

O($m\alpha(n, \lceil m/n \rceil)$): Add linking by rank to the path min data structure (not entirely straightforward)

O(m): Build optimal algorithms for very small subproblems (much less straightforward)