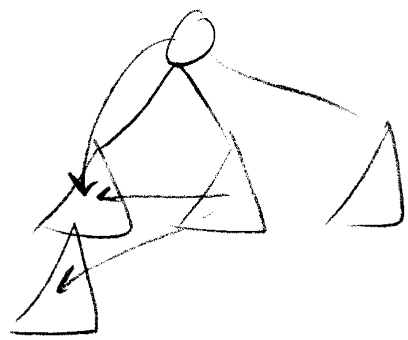
GRAPH → DFST → Superimpose all edges
Eliminate Back-Edges



We're only sure of ~~the~~ correct (ultimate) answer at Z if when we apply equation, answer was ultimate answer at all preds

Get answer there first

Topological

Pre Order, Right → Left

Tree	OK
CHORD	OK
CROSS	OK

W/ Back Edges - can't get it all in one pass

Analysis of simple dom

$\forall Y$ dom(Y) initially N $\qquad O(N^2)$

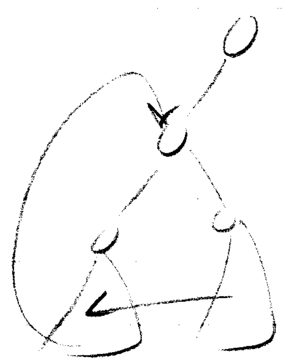
Each node can change N times, since $\cap$ only removes info

If every node changes every edge is touched

$$O(N^2 E) \text{ time}$$

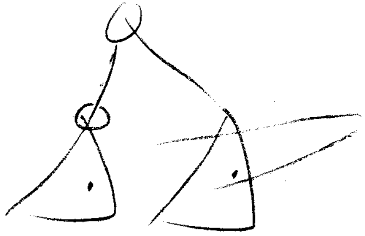
to compute $O(N^2)$ INFO

One remark: Look at DFST



Each dominator of Y must be an ancestor of Y

Why?



can't dominate Y because there's a path of tree edges to Y that avoids any non ancestor

Could initialize dom(Y) = ancestors (Y) and save time

Maybe DFST can be of even more use ...

Look at dom(Y) for Y 1 2 11 12

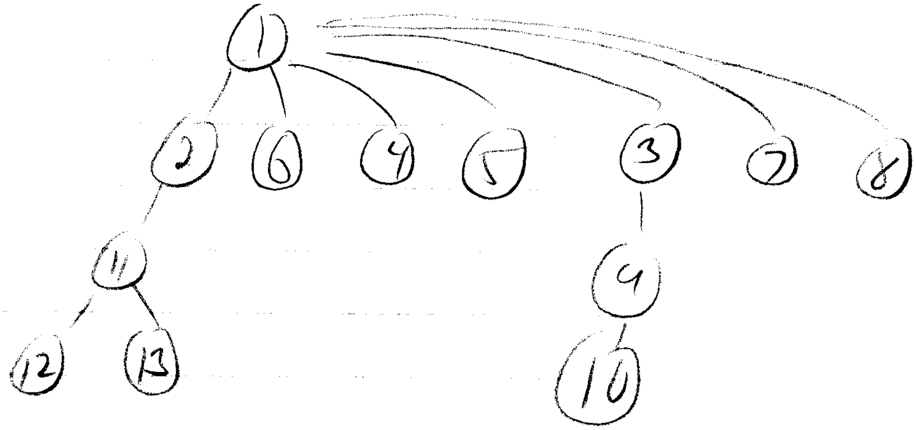
Y	dom(Y)
1	1
2	1 2
11	1 2 11
12	1 2 11 12

Seems redundant

In fact

$$\left.\begin{array}{l} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{array}\right\} \text{dom takes } O(n^2) \text{ space to store}$$

Look at dom Tree (IDOM)



to get dom() just go up tree

The above (tree) takes $O(N)$ storage

Can we compute IDOM faster?

Yes: Lengauer-Tarjan alg $O(E\,\alpha(N,E))$

Approach:

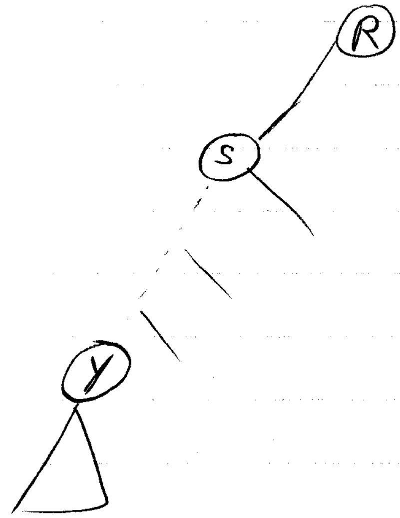
1) G transformed to G'
   Same nodes as G, nodes have same dominators, but G' has no cross or back edges

2) Find IDOM for a graph free of cross and back edges

1) Elimination of Cross + Back Edges

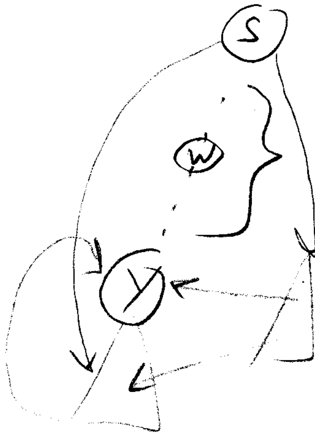
Recall that doms(Y) must be ancestors of Y in (any) DFST

So IDOM(Y) is also some (proper) ancestor of Y - closest ancestor that dominates Y



Lemma 2

$S = IDOM(Y) \implies$ must be some path $p: S \xrightarrow{+} Y$ such that S + Y are the only ancestors of Y



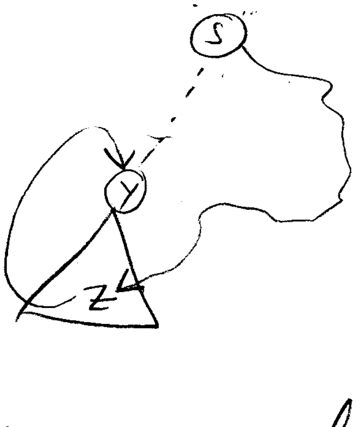
s plays "keep-away" from these ancestors of y

Otherwise, cannot hit Y without including some W and so S either doesn't dominate Y at all or else W is closer

We'll call s "sneaky" – how does it do it?

Tree: By def
Chord: By edge (s, Y)

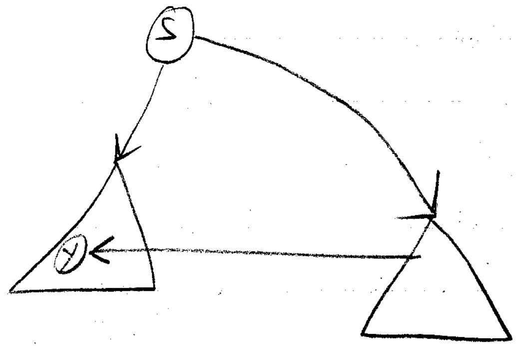
BACK:



$s \xrightarrow{+} Z \xrightarrow[BACK]{} Y$

Summarize by (s, Y) edge (forward)

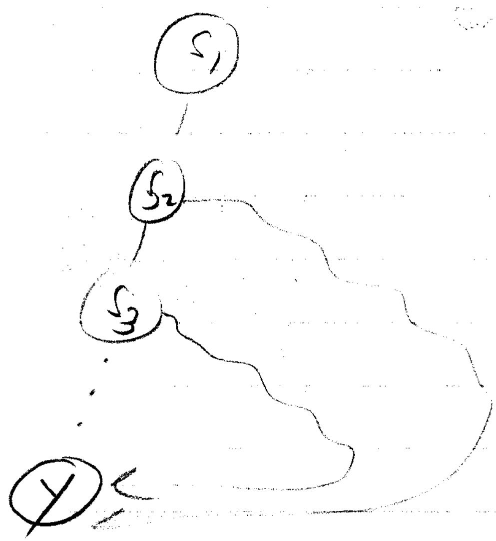
Same story w/ cross edge



Summarize w/ forward edge (S, Y)

SHOW Alg[5]

Imagine a competitive game of keep-away

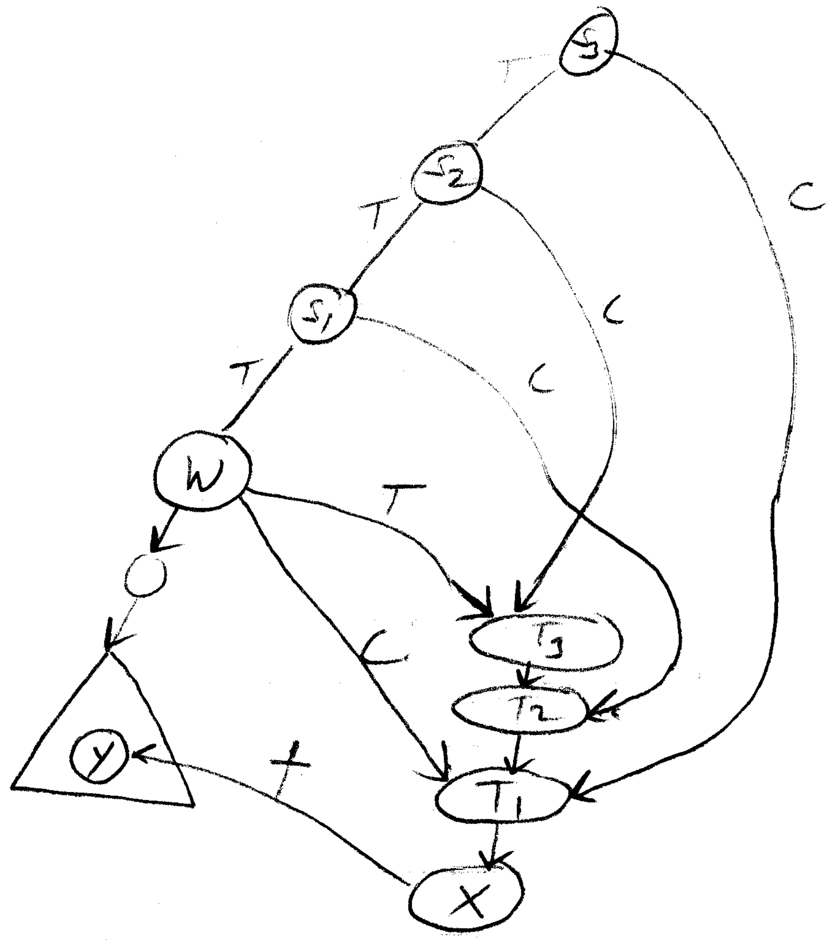


Multiple ancestors of Y try to be sneaky

Only the <u>sneakiest</u> of ancestors has any hope of dominating Y (and even then it may not succeed)

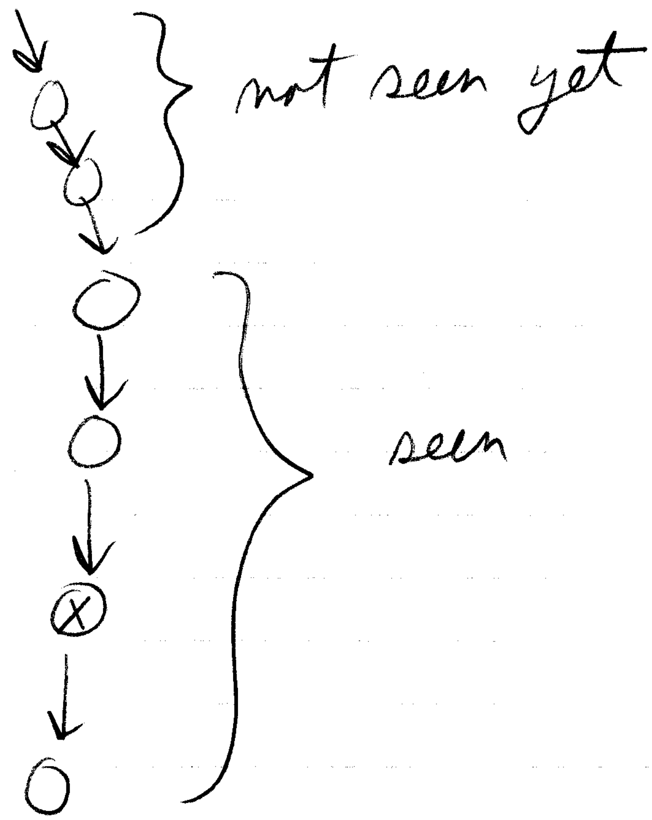
2/3/92

This sneakiest ancestor of Y
is defined as sdom(Y)
(semidominator (Ralf)
(may dominate, may not)



W - common ancestor of X + Y, uses
(X, Y) to be sneaky to Y

But ancestors between W + X
allow S_1 S_2 S_3 to be sneaky to
Y, also using (X, Y) edge
S_3 is sneakiest



EVAL(X) : returns seen ancestor of X w/ sneakiest sdom

We go backwards — over DFN

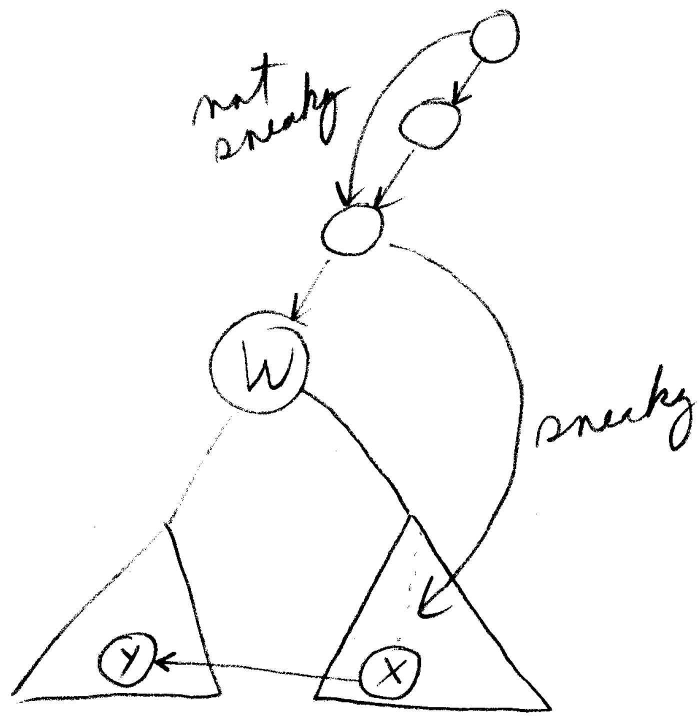
On example, haven't seen W yet

Or anything between W and Y

EVAL(T_3) is T_3
Eval(T_2) is T_3
Eval(T_1) is T_1
Eval(X) is T_1

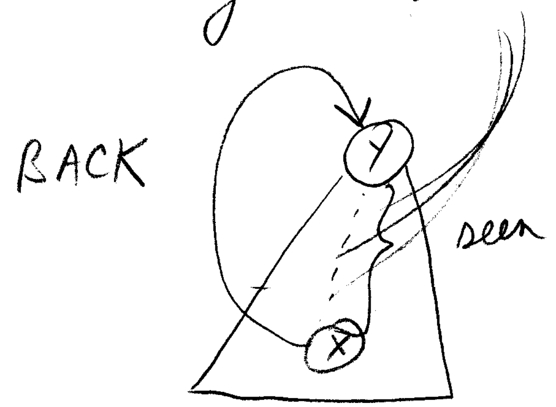
Initially sdom(X) = dfn(X)

After Eval(X), sneakiest via X is sdom(eval(X))

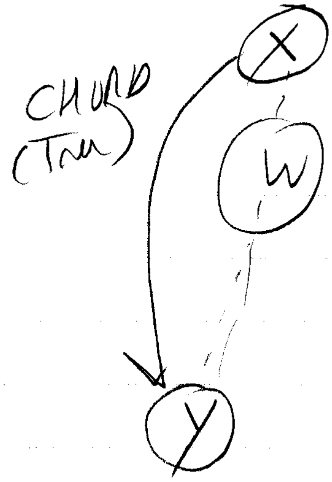


Let's look at EVAL(X) for any pred X of Y

We just saw CROSS

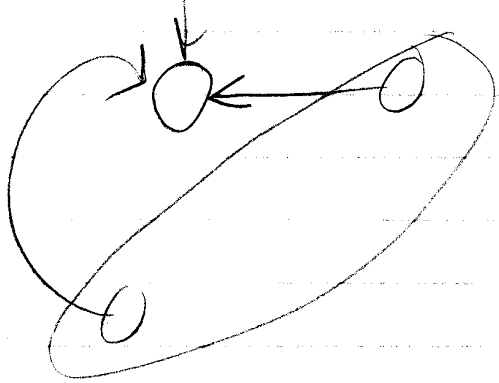


sneak to nodes between Y and X



must hit Y if they fall short, they're not sneaky

So use dfn( )



already know sdom up to common ancestor

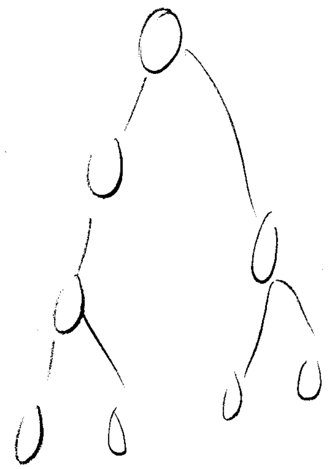
after looking at each pred, we get sneakiest ANCESTOR

PS: Inductive

BASE: Node N – only back edge to itself
chord edges use dfn

INDUCT: If know answer n+1 – N
can decide at node n

New Graph



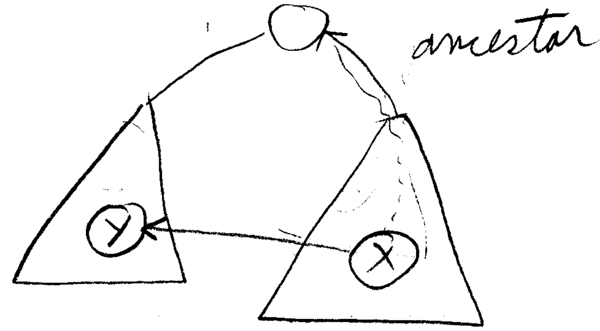
DFST
+ all edges
(sdom(Y), Y)

sdom(Y) always some proper ancestor, so new graph has only forward edges

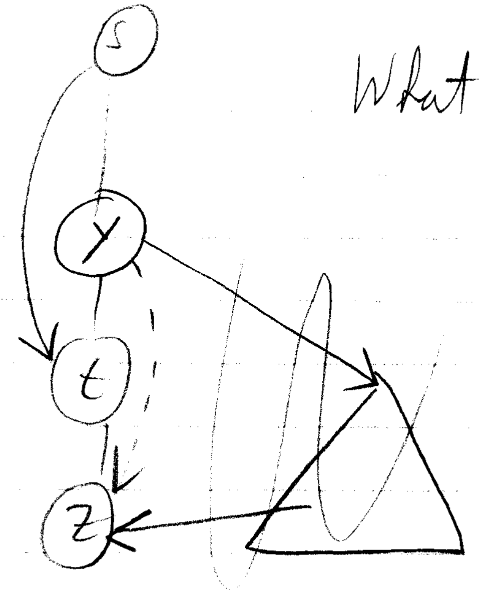
Example FIG 2.22
2.23

Let's look at alg 6

EVAL



s.dead(X) = list of nodes semidominated by X — use later



What is sdom()?

Y sdom of Z, but doesn't dominate

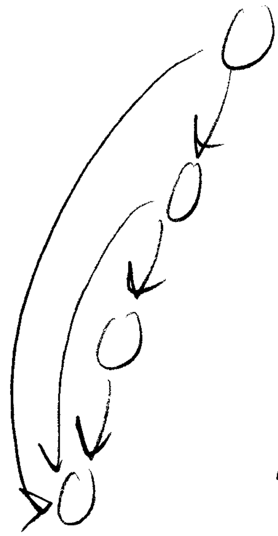
If Y dominated Z, it would be the immediate dominator

All paths to Z include Y
Y can skip all nodes between itself & Z

Now start w/ graph of only forward edges

Maybe we have this to begin with, or maybe we get it by eliminating cross & back edges

As before, need only sneakiest

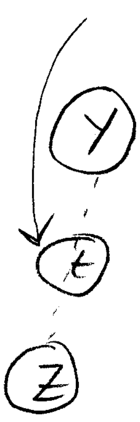


sdom(Y) = lowest numbered predecessor

2 passes

1) See if sdom(Y) dominates Y if so, set answer otherwise defer

2) Handle deferrals



look at what's semidominated by Y

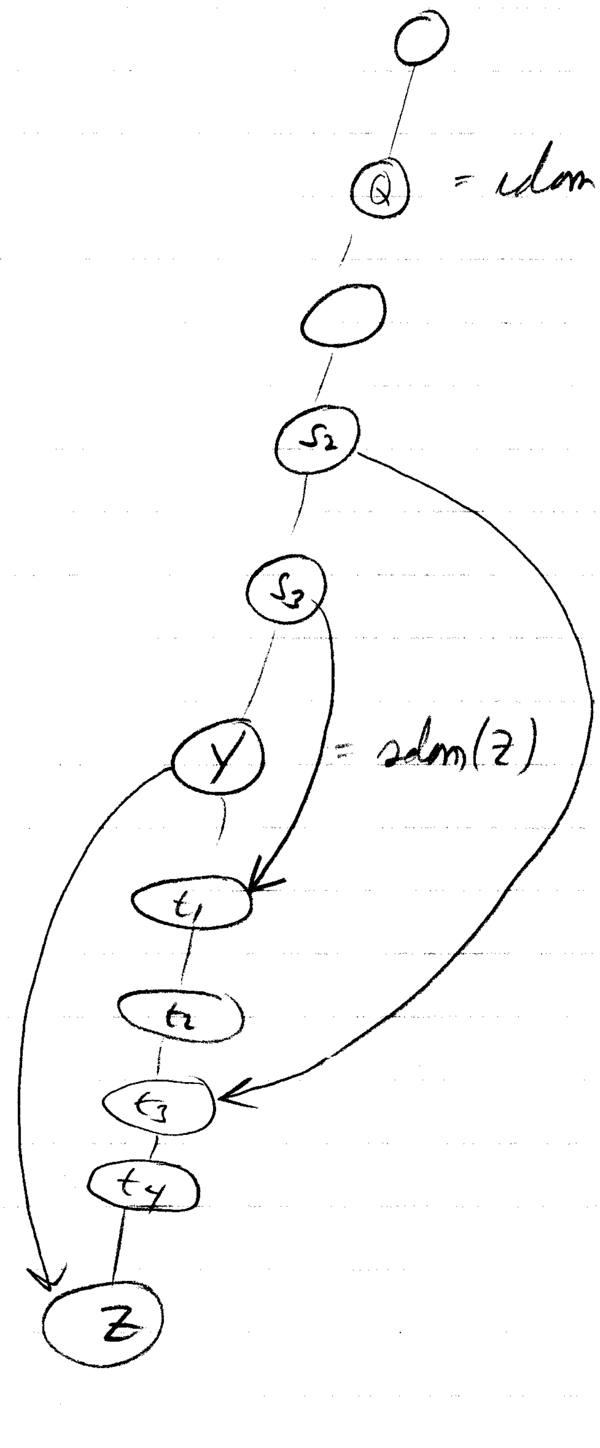
Z is

If we do EVAL(Z), we'll see if any node leap-frogs over Y

If nobody does then we have answer

$$Y = sdom(Z) = idom(Z)$$

Otherwise



Sneakiest leap-frogger $S_2 \to t_3$

Then whoever immediately dominates t_3 also immediately dominates Z

Proof

Suppose $Q = idom(t_3)$ doesn't dominate Z

$\exists$ path Root $\xrightarrow{+} Z$ that avoids Q

We know Q is an ancestor of S_2

Some node above Q must avoid Q + head for Z

1) Can't hit Z directly or it would have semidominated Z

2) Can't hit any node between Q and Y - if it did then Q doesn't dominate t_3

3) Can't hit Y for same reason

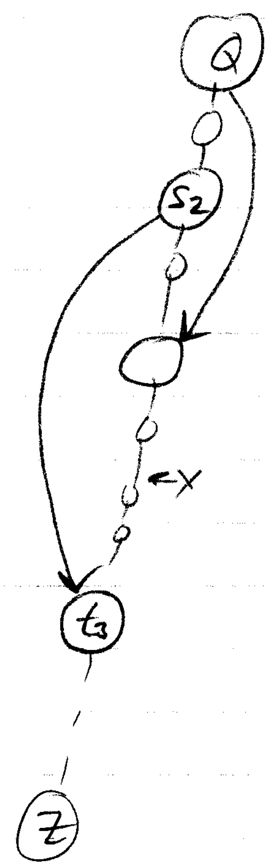
4) Can't hit any node between Y + t_3 on t_3 for same reason

5) Could only hit some node between t_3 + Z, like t_4

But if it does, then
S_2 isn't sneakiest leapfrogger

Therefore idom(t_3) dominates Z

How do we know it immediately dominates Z?



Q = idom(t_3) ⟹
From Q we can avoid any node between Q and t_3

No node between Q and t_3 can dominate Z

t_3 can't dominate Z because it's below sdom(Z)

No node between t_3 & Z can dominate Z for the same reason

Q is closest dominator of Z

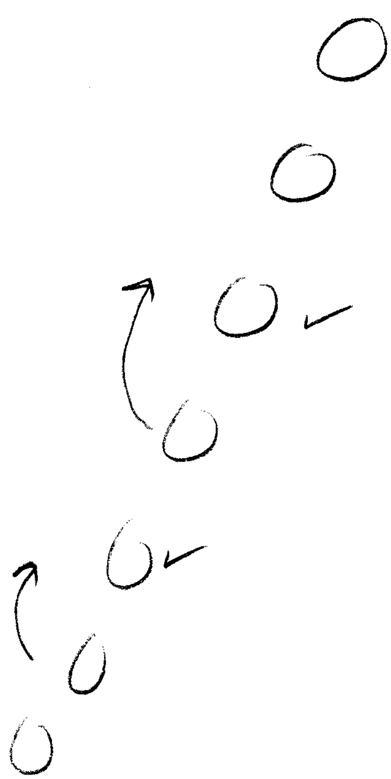
Back to alg

Compute $t = EVAL(z)$

If $sdom(t) = sdom(z)$
then $IDOM(z) = sdom(z)$

else $SAMEDOMAS(z) = t$

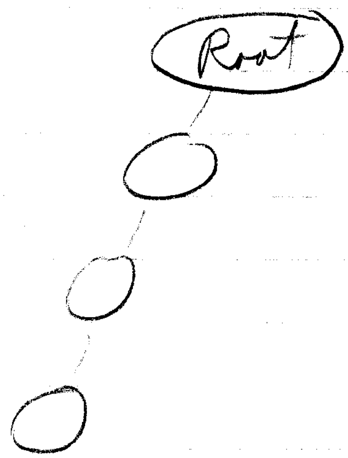
t is always some ancestor of z
(between $sdom(z)$ and z)



Either know answer,
or else it's same
as some ancestor

What should we do?

Final Pass



For n = 2 to #Nodes
  z = vertex(n)
  If idom(z) undefined
    idom(z) = idom(sameDomAs(z))

Thm this works

<u>Proof</u> Inductive

Can't use Root, because it has no immediate dominator

1) All children of Root have known idom = Root because nobody can leapfrog between them + Root

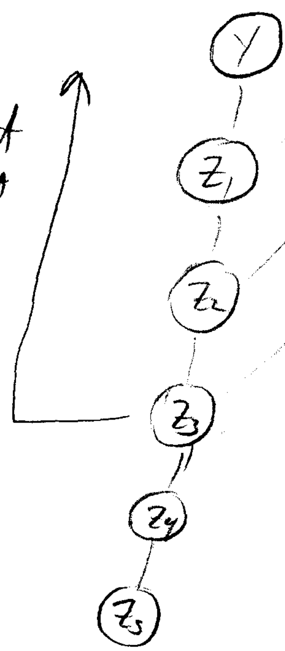
2) If we know

this much

then we can compute idom(Z)

Let's look again at EVAL

find target of sneakiest ancestor z_i w/ min sdom(z_i)

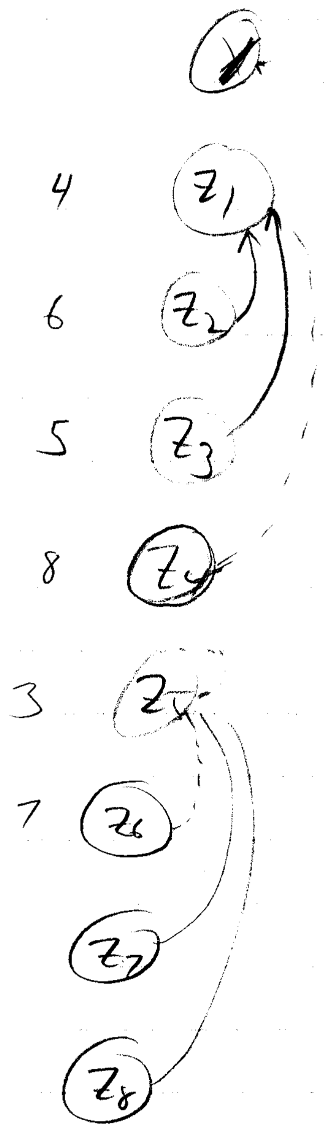


Suppose EVAL(Z_3) is Z_1

Subsequent EVAL(Z_2) gives same answer

If Z_1 answer for all nodes takes quadratic time!

Solution: don't just search for answer
remember solution along the way

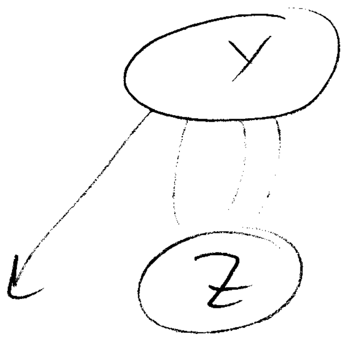


After EVAL(z_3)
PATH Compression

After EVAL(z_6)

After EVAL(z_8)

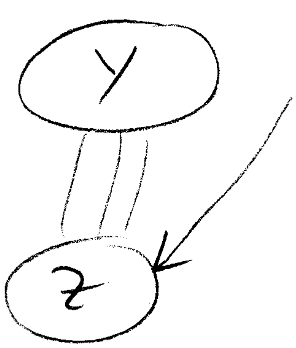
DOMINANCE



Y dom Z   Y is
on all paths from
Root to Z
If you get to Z, you
must have passed
through Y

Where must I come from?

Post Dominance



All paths from
Y to EXIT
include Z

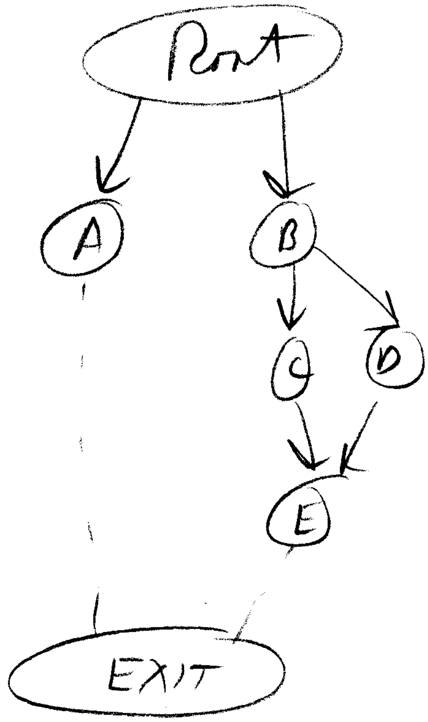
If you hit Y you'll
have to hit Z

Where must I
go to?

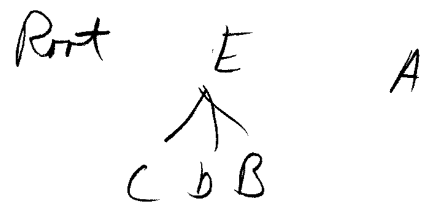
So we have

Z postdom Y
strictly postdom Y
immediately postdom Y

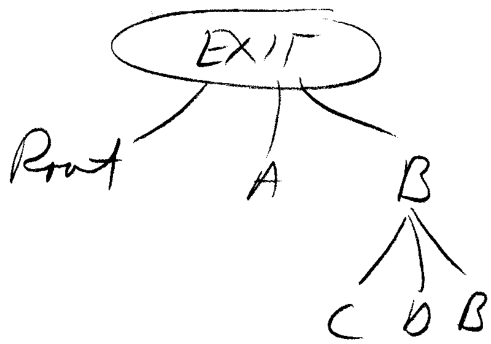
We don't necessarily get a tree



FOREST



Add Exit Node



How to compute

Thm

If Y dominate X in $G^{REVERSE}$

then Y postdominate X in G

Proof

Suppose not ...

There must be a path

$p: X \xrightarrow{+} EXIT$ in G

that avoids Y

Reversing this path for G^R says there's a path from EXIT to X that excludes Y

Then Y couldn't dominate X in G^R

$\square$

Compute by:

1) G^R is reverse of G

2) dominators of G^R are post dominators of G