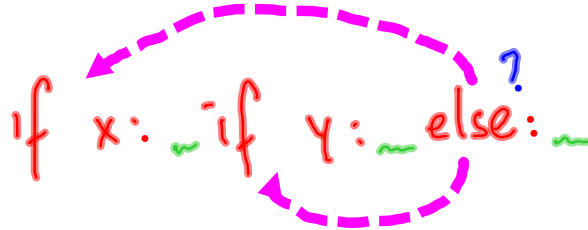


Trees

Record structure of expressions

$6 + 3 * 8$ BODMAS $\rightarrow 6 + 24 = 30$

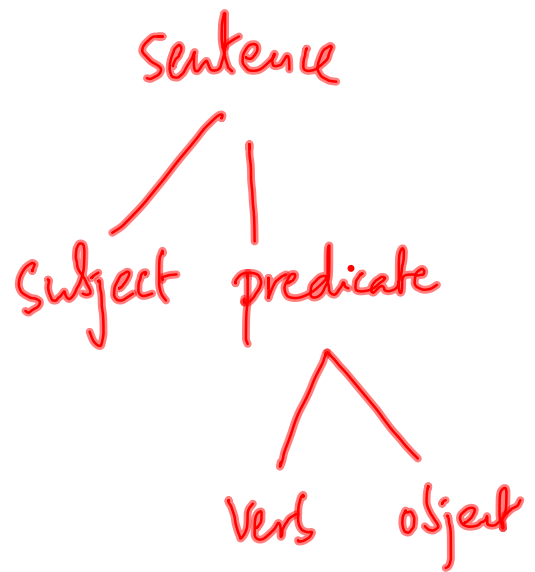
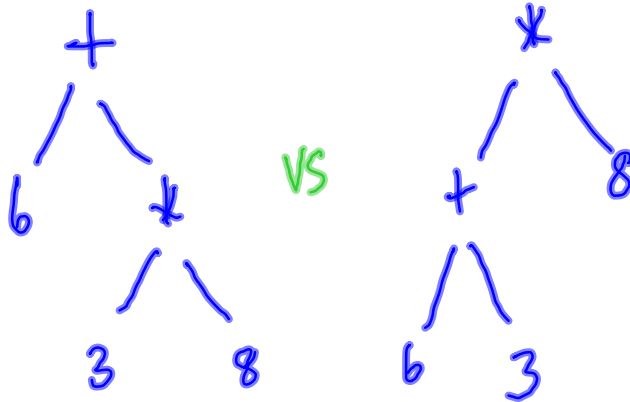
if x: ~ if y: ~ else: ~


Avoid ambiguity \rightarrow Parentheses

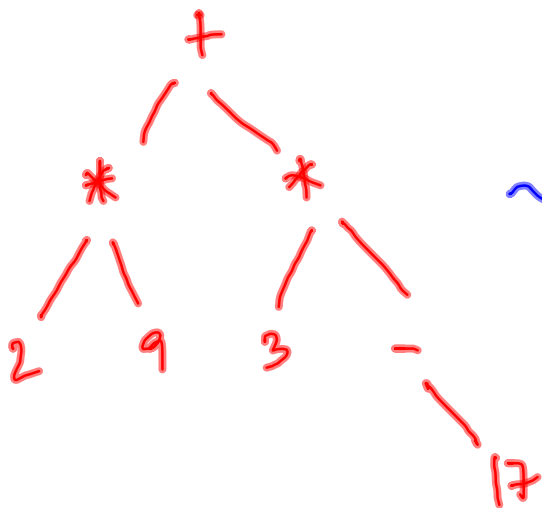
$6 + (3 * 8)$ vs $(6 + 3) * 8$

Better representation

"Parse tree"



Have a parse tree \rightarrow recover the string



$$\underbrace{2 * 9}_{\text{left}} + \underbrace{3 * -17}_{\text{right}}$$

$$x = (b ? v1 : v2)$$

```
if b
  x = v1
else
  x = v2
```

Tree \rightarrow String

left subtree \rightarrow String

Root

right subtree \rightarrow String

Inorder
traversal
of the
tree

Other traversals are possible

Other traversals

root

left subtree

right subtree

left subtree

right subtree

root



+ * 2 9 * 3 - 17

Preorder

2 9 * 3 17 - * +

Postorder

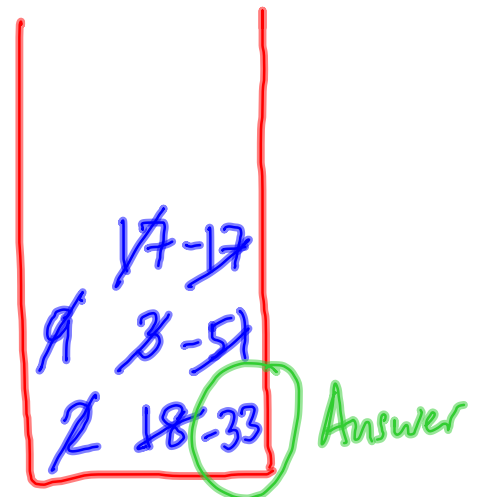
29 * 3 17 ÷ * + ↙ negation, not subtraction

Evaluation strategy

left to right

- See a number, put it on stack
- See an operation, operate on top of stack, replace with answer

STACK



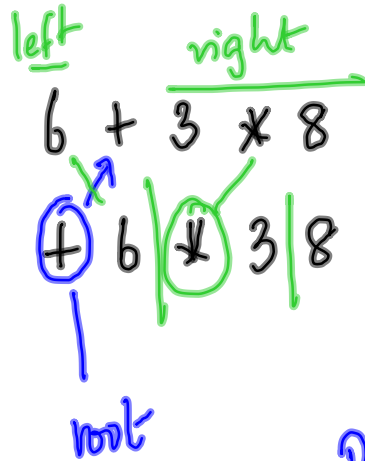
Postorder expressions can be evaluated unambiguously on a stack

Till ? HP calculators expected input in postorder

Inverse problem:

Reconstruct tree from its traversal

One traversal clearly is not enough



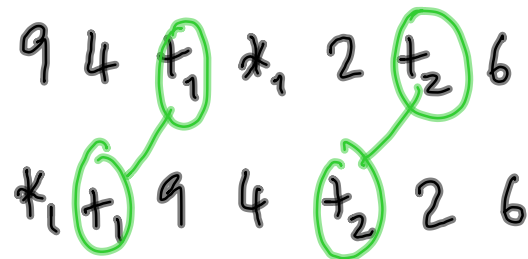
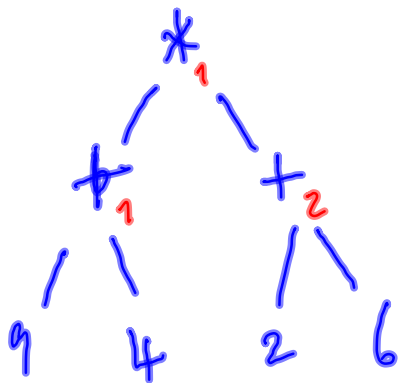
inorder }
 preorder } → reconstruct?

Preorder identifies root position
in inorder

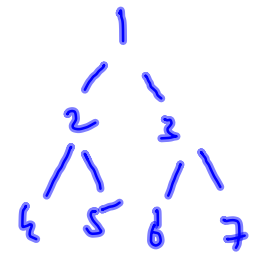
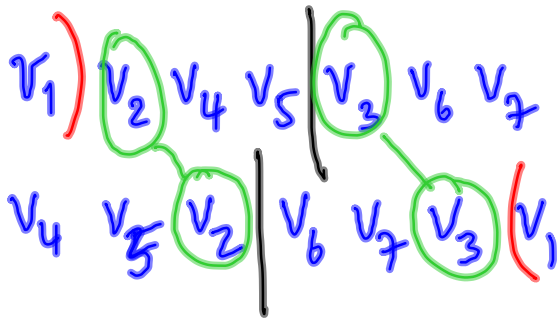
Inorder gives length of left/right
segments

Inorder + Postorder — similar, mirror image argument

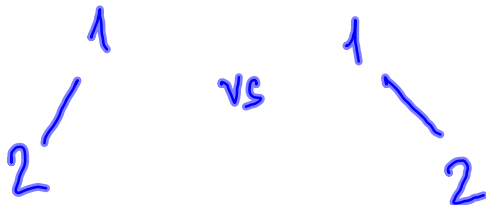
Need to unambiguously name nodes



What about reconstructing
 from preorder + postorder?



What about

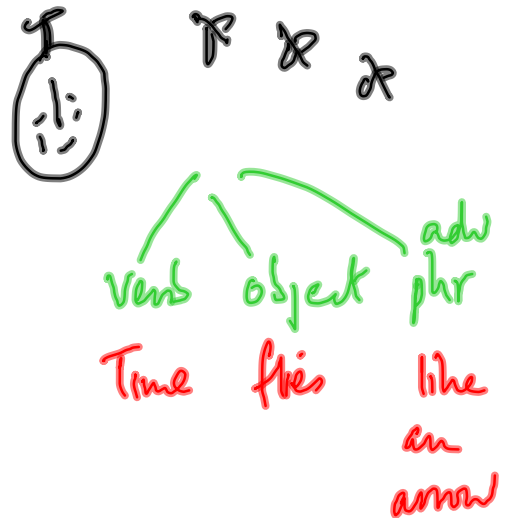
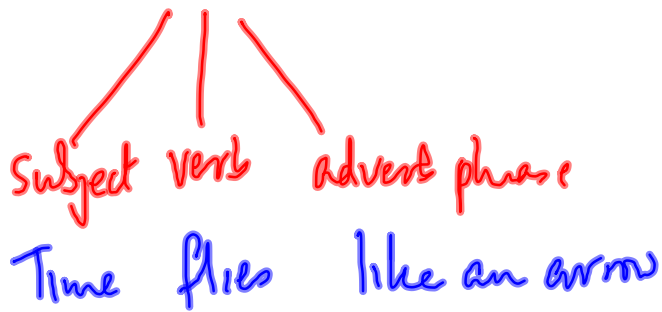


Preorder: 1 2 _ 1 _ 2

Postorder: 2 - 1 - 2 1

Natural language:

Time flies like an arrow.



Back to storing values for retrieval

Heap — insert, deletemax

In general, want to support

Insert

Delete

Lookup

Updated dynamically

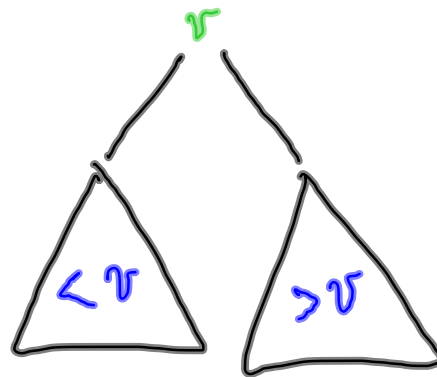
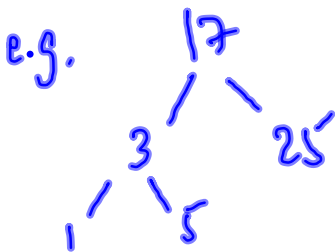
works well if sorted

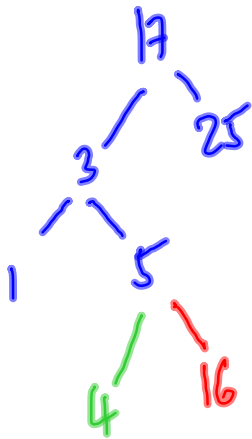
Search tree

Stores distinct values, one per node

Invariant:

At any node





Add 16 ?

Add 4 ?

Search & Insert are related

Search:

Start at root & follow
one path

If value already
exists, do
nothing

Insert:

Follow search path, &
insert where you expect it

Suppose we insert 7, 6, 5, 4, 3, 2, 1



Effectively a list

Search & insert follow exactly one path

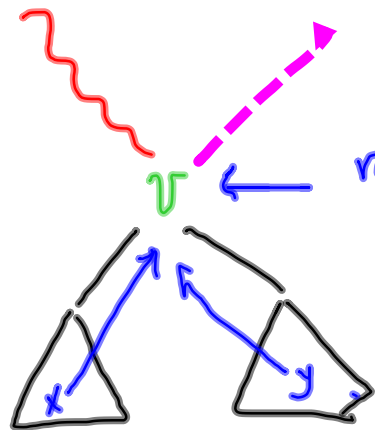
from root to leaf

But this path could be very long!

What about delete?

If v exists, delete it

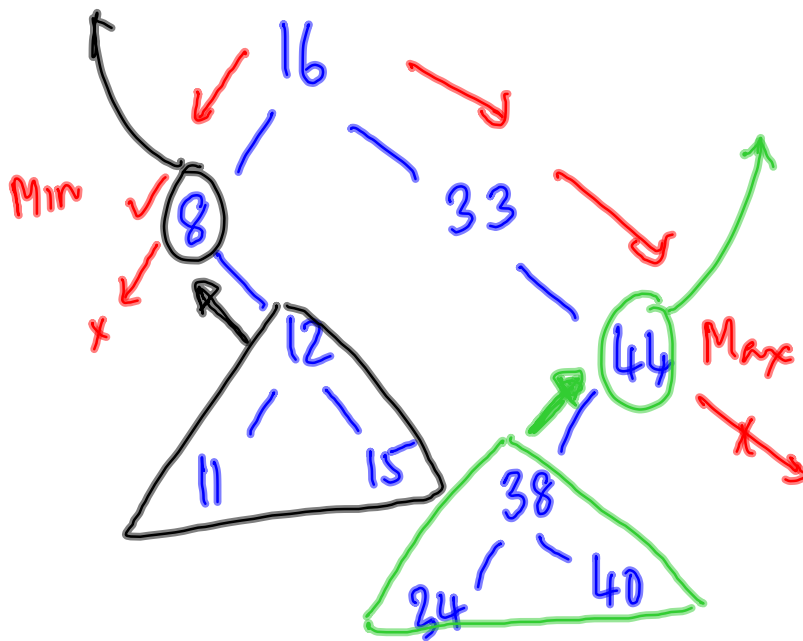
Find v — search



replace with a value
that is consistent
with subtrees

Promote max value
in left or min
value in right

Keep going left — min value
Keep going right — max value



Delete

— Deletemax / Deletemin