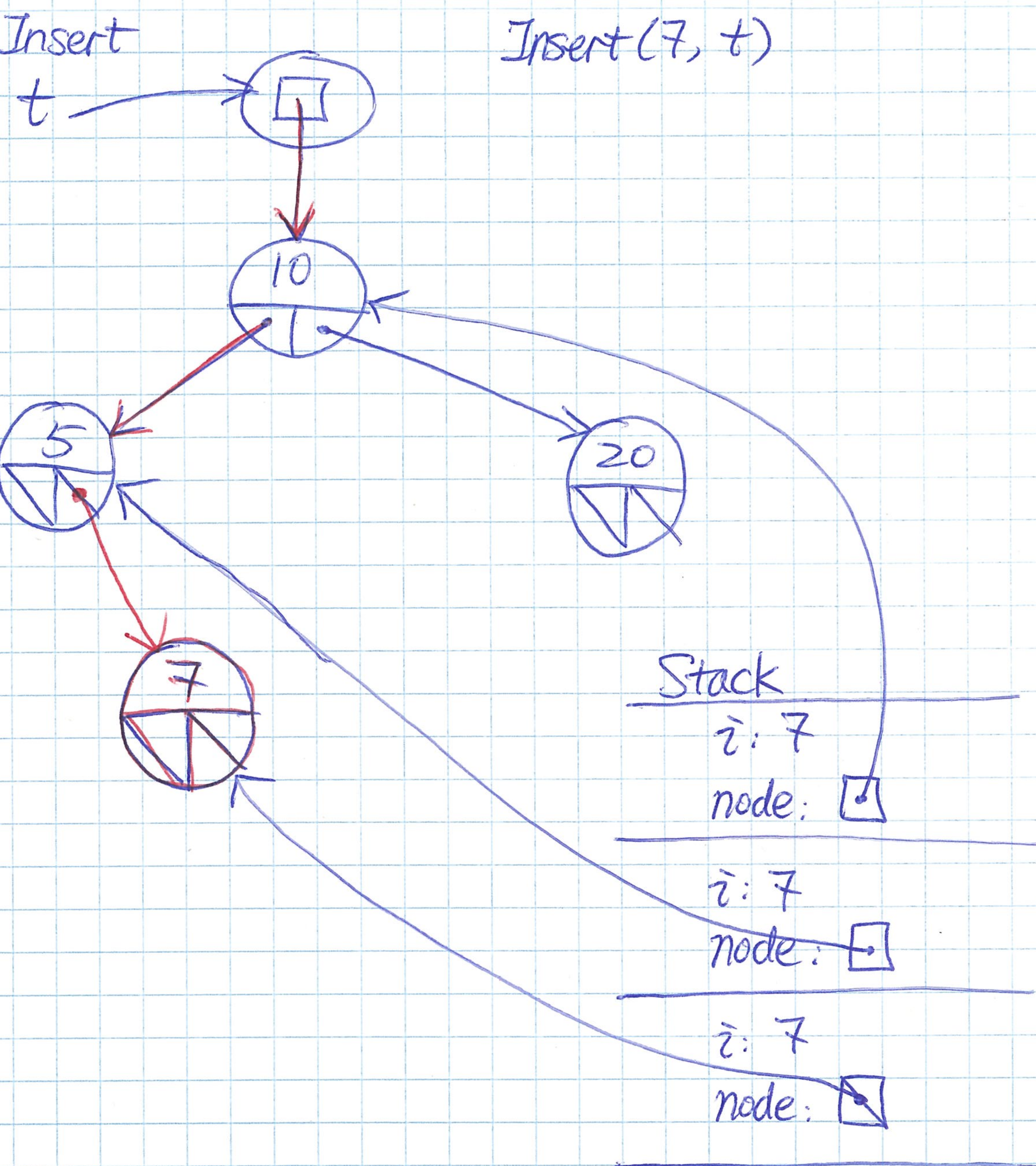
Recursive Insert



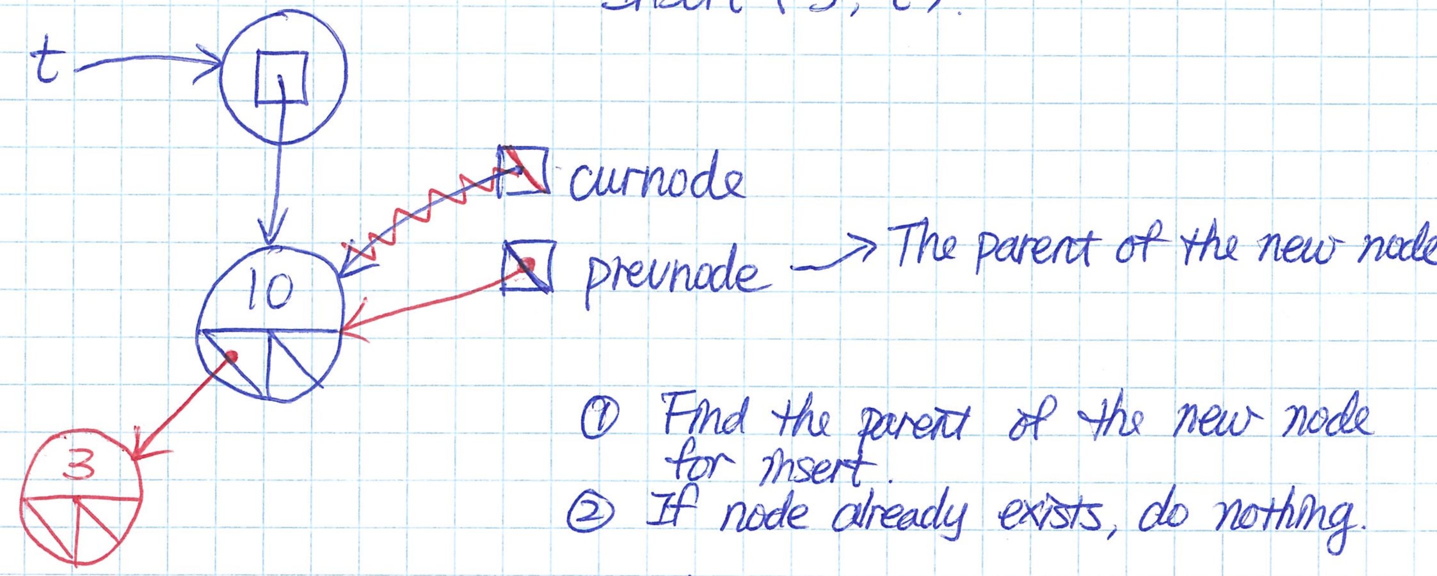
We always return the parent node of the subtree.

Why?

- Most of times it doesn't change.
- It does change when we perform the insertion.

Iterative Insert

Insert (3, t).

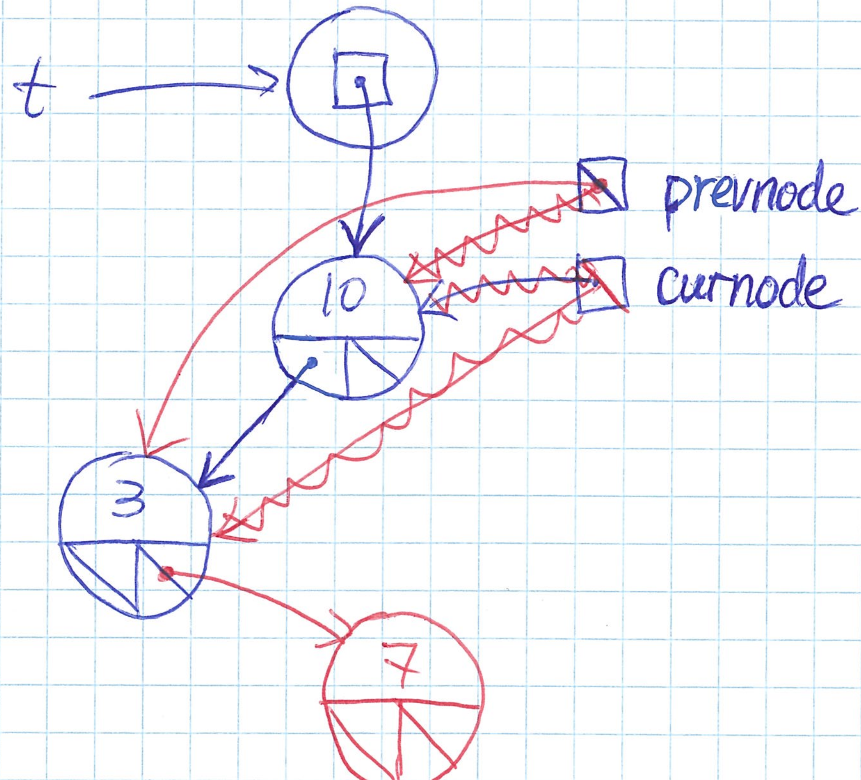


① Find the parent of the new node for insert.

② If node already exists, do nothing.

③ When inserting, figure out which child should the new node be.

Insert (7, t).



Summary of runtimes. n is total # of items.

	sorted array	BST	Balanced BST
insert	$O(n)$	$O(n)$	$O(\log n)$
search	$O(\log n)$	$O(n)$	$O(\log n)$

sorted array:

search: use binary search $O(\log n)$.

insert: shift all elements $O(n)$.

BST: Depends on what the tree looks like.

Advantages of BST:

- can search in $O(\log n)$
- insert/remove in $O(\log n)$

$\}$ if balanced.

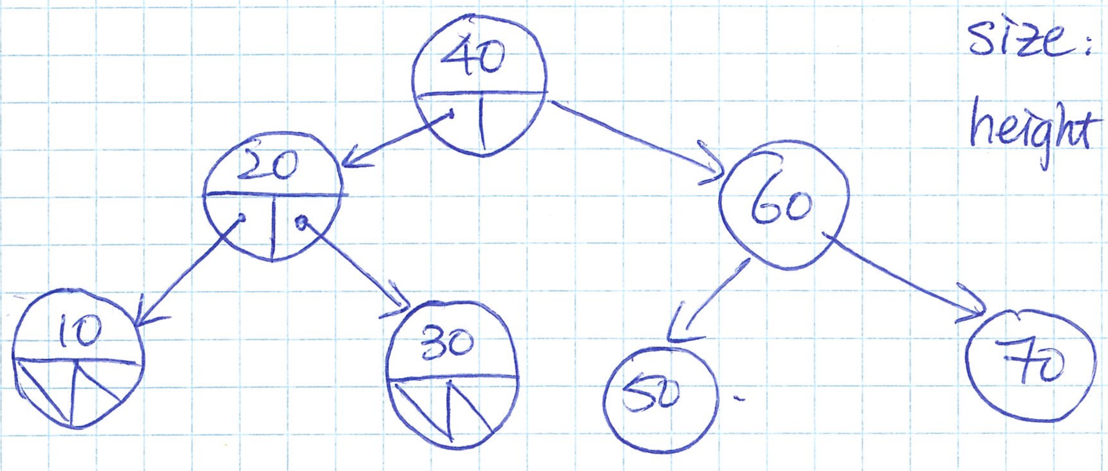
Disadvantages of BST:

- tricky to rebalance.
- if unbalanced, a linked list.

  search, insert, delete are all $O(n)$.

Possible to implement a self-rebalancing tree. AVL tree. DEMO.

Tree A [40, 20, 60, 10, 30, 50, 70]



size: 7

height: 3 $\approx \log_2 7$

Balanced tree. short & fat.

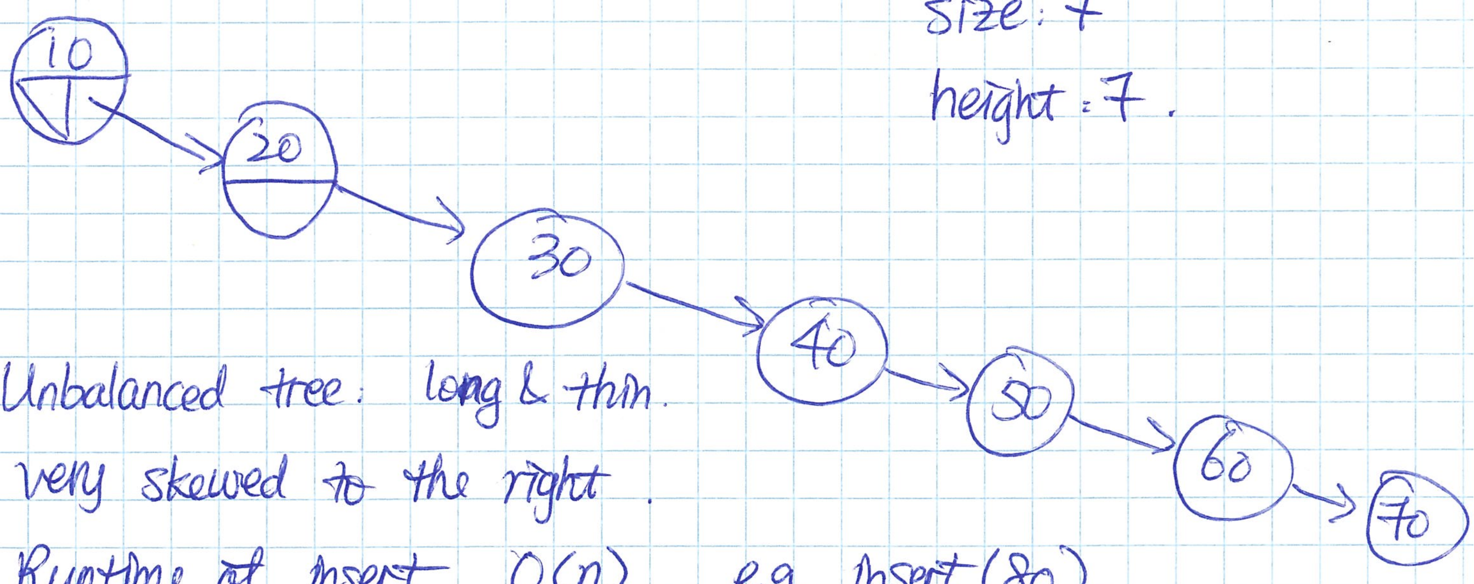
For every node, sizes of 2 subtrees are similar.

Runtime of insert $O(\log n)$. e.g. insert(55)
go thru 3 nodes.

Tree B [10, 20, 30, 40, 50, 60, 70]

size: 7

height: 7.



Unbalanced tree: long & thin.

very skewed to the right.

Runtime of insert $O(n)$. e.g. insert(80)
go thru 7 nodes.

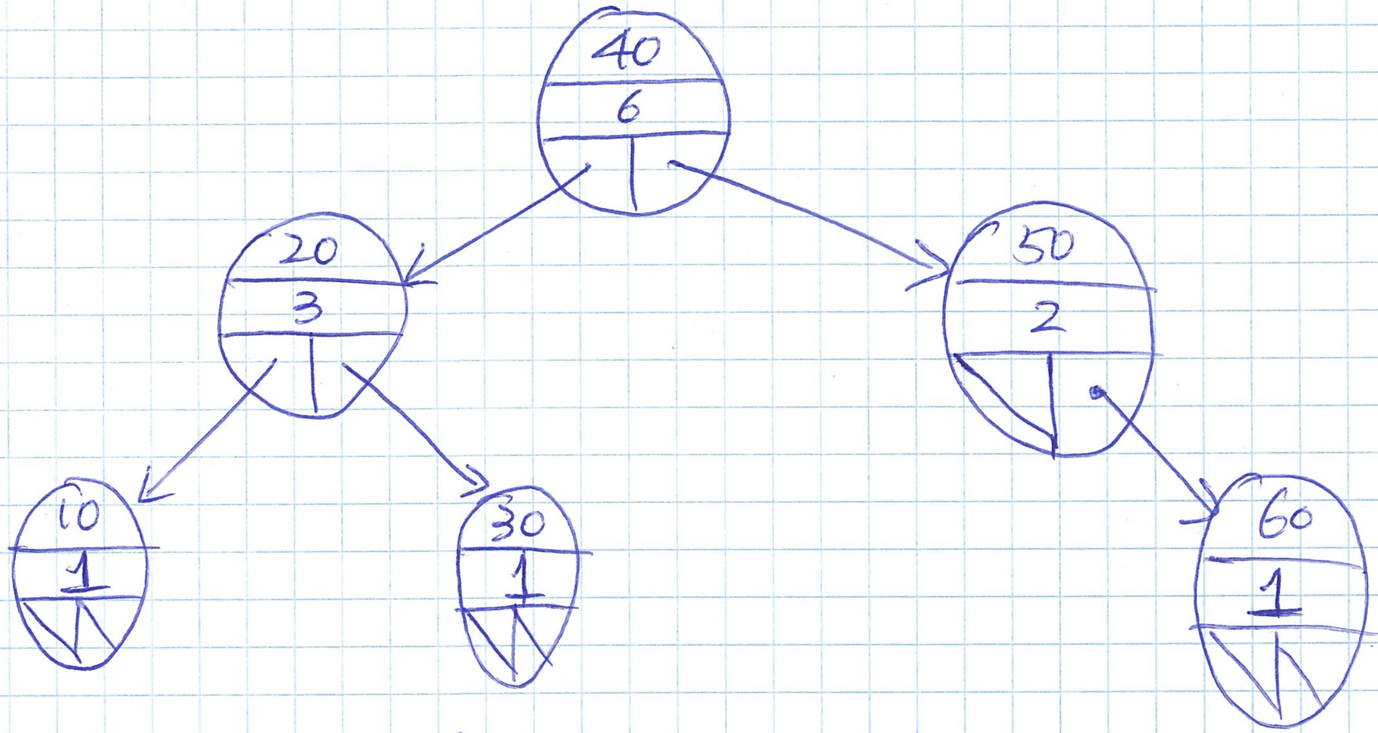
Size node augmentation.

Insert into the tree what do you need to update?

↪ with myself as the root

- store size of subtree in each node.
  - rebalance a tree.
  - get node given an index. select(k, t)

k ↑ index, starts from 0.



Index $3 =$ # of nodes in left subtree. myself.

Index < 3 left subtree same index.

Index > 3 right subtree. index − me − left_size.

select(2, $\overset{=40}{t}$). left_size = 3. $2 < 3$.

select (2, node 20) left_size = 1 $2 > 1$.

select ($\underset{2-1-1}{\overset{}{0}}$, node 30) left_size = 0. $0 == 0$.

$0 = 2 - 1 - 1$

return 30;

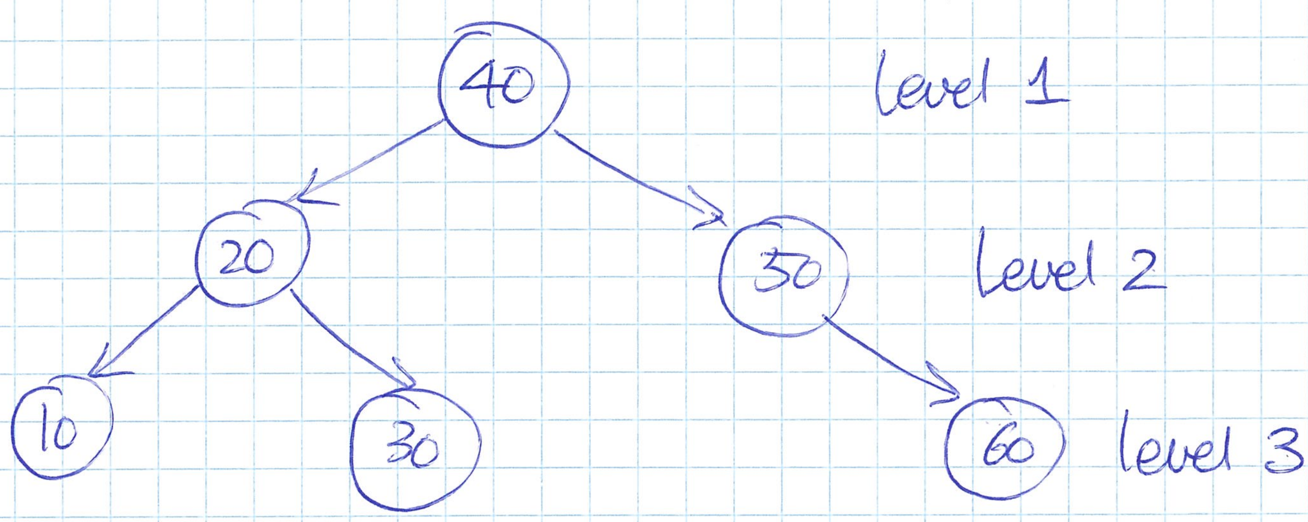
Array-based trees:

e.g. the heap data structure
often a complete tree.

advantages:
- few pointers.
- easily access parent

disadvantages:

- waste space if tree not complete.
- self-balancing can be awkward.



level 1	level 2		level 3			
40	20	50	10	30	–	60

choose a sentinel value.