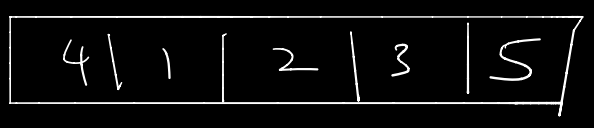
# Trees.

- Say we want to find a value quickly, and insert it.

Current Candidates

(1) unsorted array / list

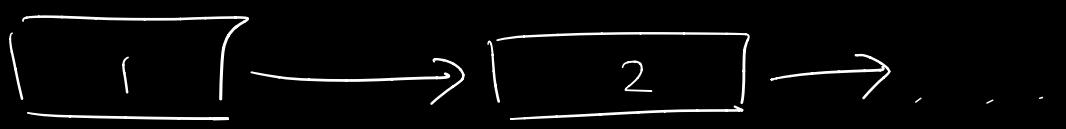



- O(n) time req to find value
- linearly scales w input size
- O(n) insertion.

(2) sorted array

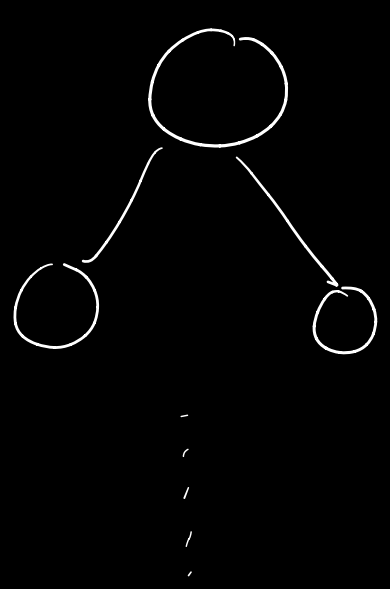

- O(log n) for finding value
- O(1) for finding insert pt
- O(n) for insertion. because need to shift arr

## ③ Linked list




- O(n) for finding value
- O(1) for insert

## ④ heap



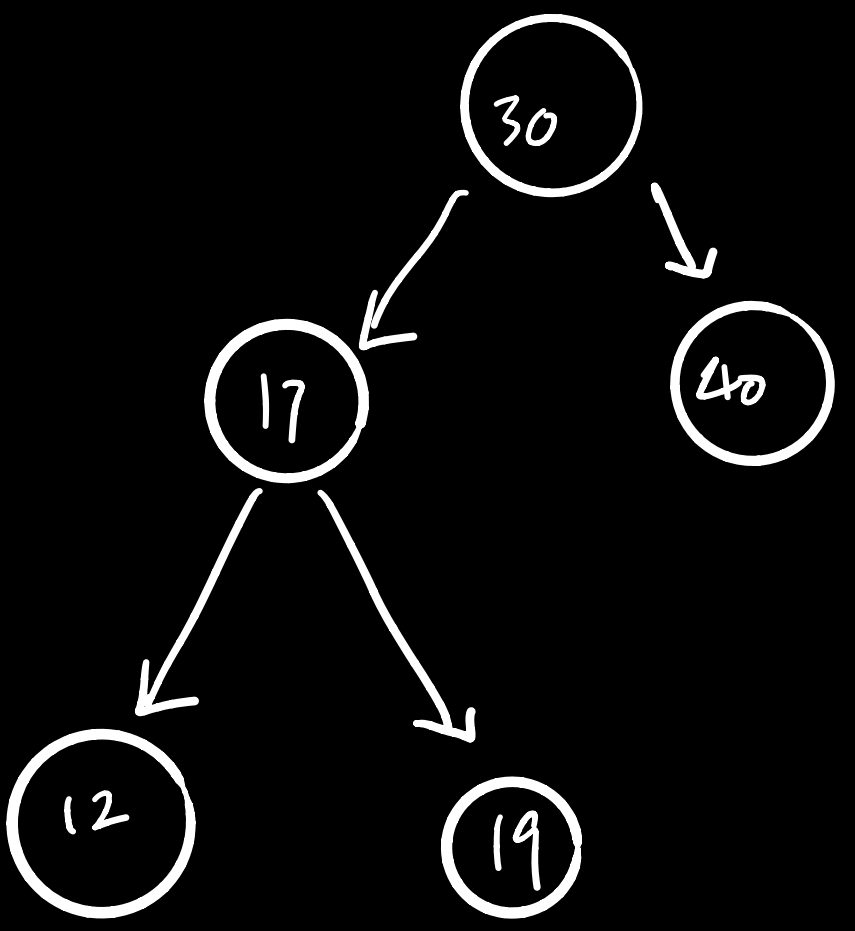
- O(lgn) insert
- O(n) search for value

What can we use instead?

- ☐ fast insert → log n
- ☐ fast search → log n

} ... how?

# Binary Search Trees!




- each node store 3 info
  - val
  - left child
  - right child

- All nodes obey one invariant.

> for all nodes $x$, if $y$ in the left subtree of $x$, $x.val \geq y.val$.

Operations ① find

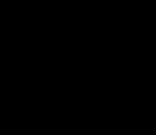
② insert

Example.

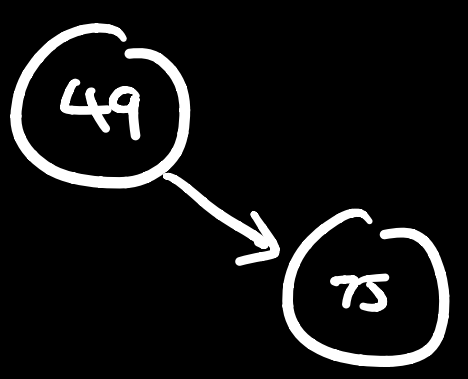
(1) Init

None

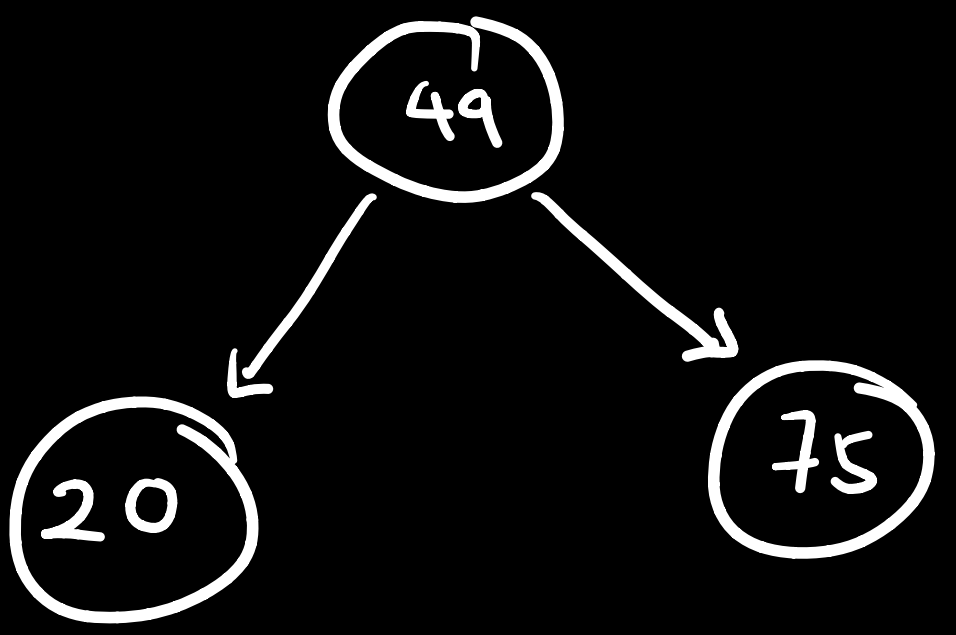
(2) Insert 49



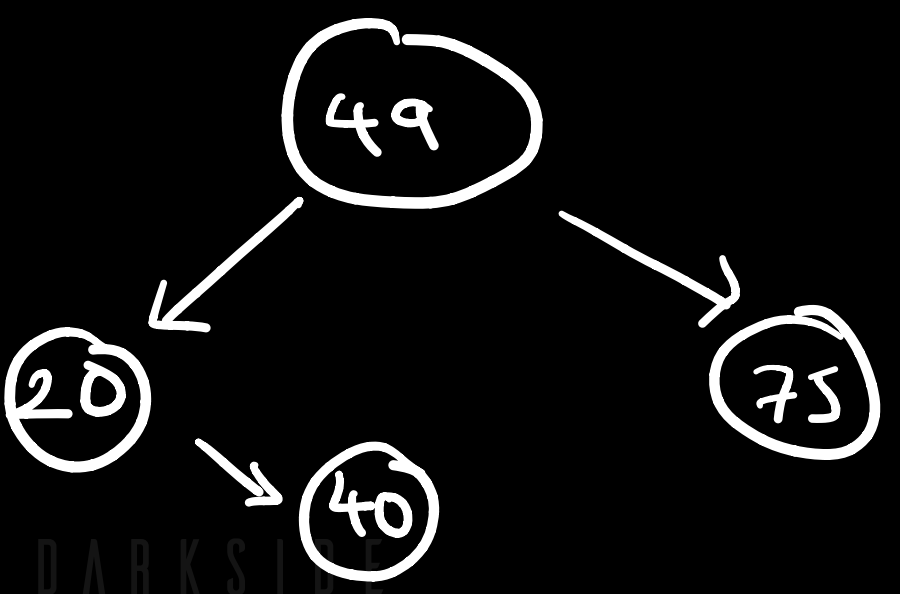

(3) Insert 75




(4) Insert 20




(5) Insert 40



JOIN THE DARKSIDE