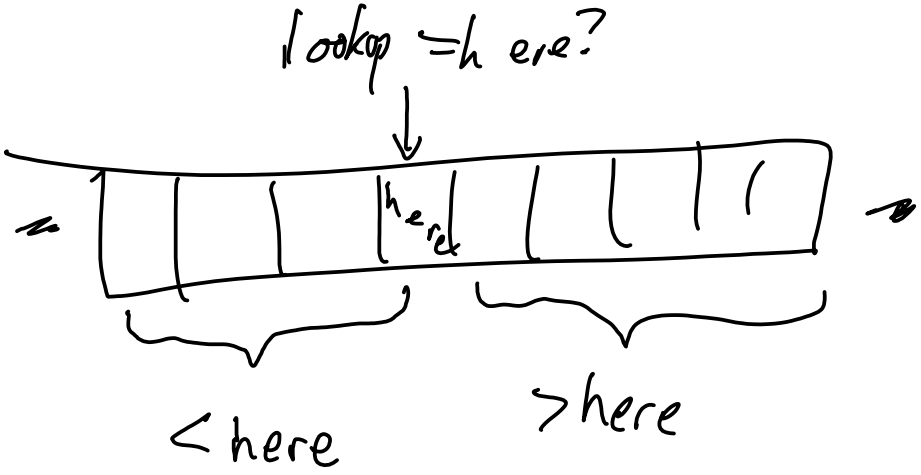
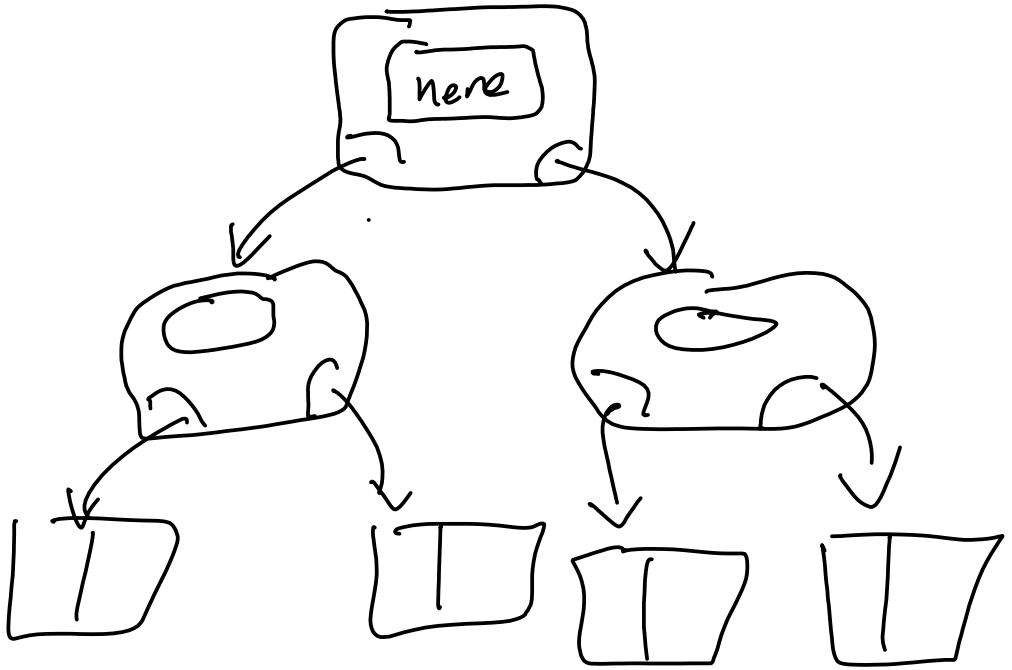
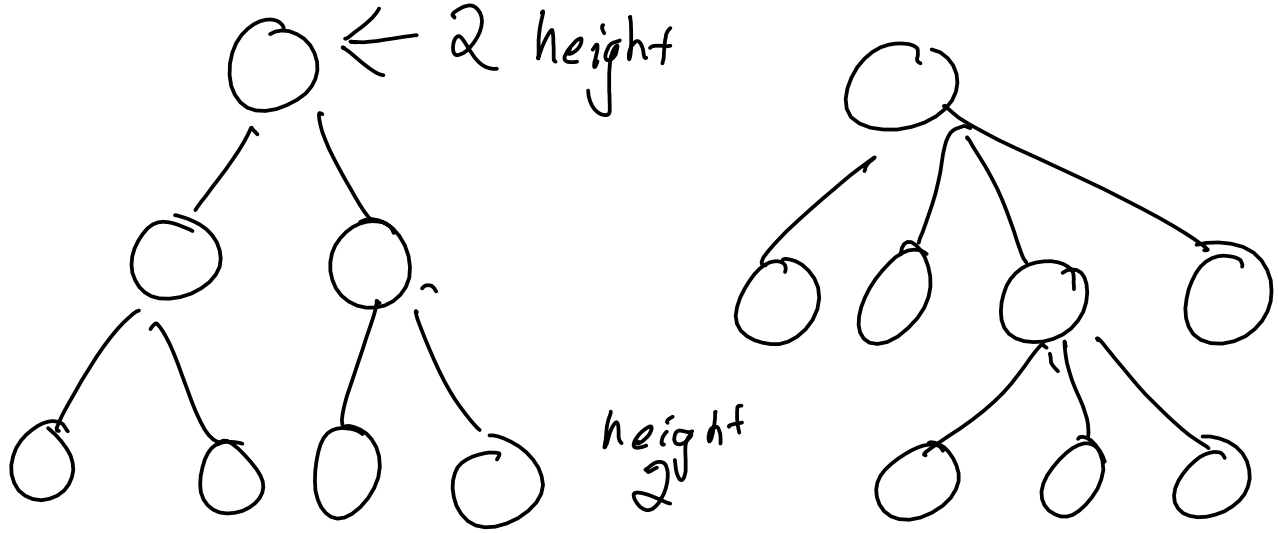
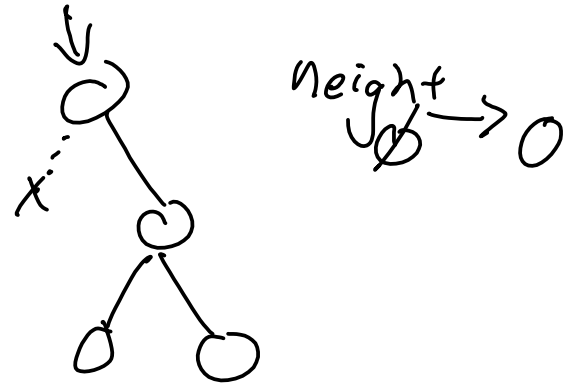

lookup = here?
here
< here
> here


here

Tree

has Nodes





Nodes: data: T

children: Nodes

Tree has "height"

has "size"

Binary Tree

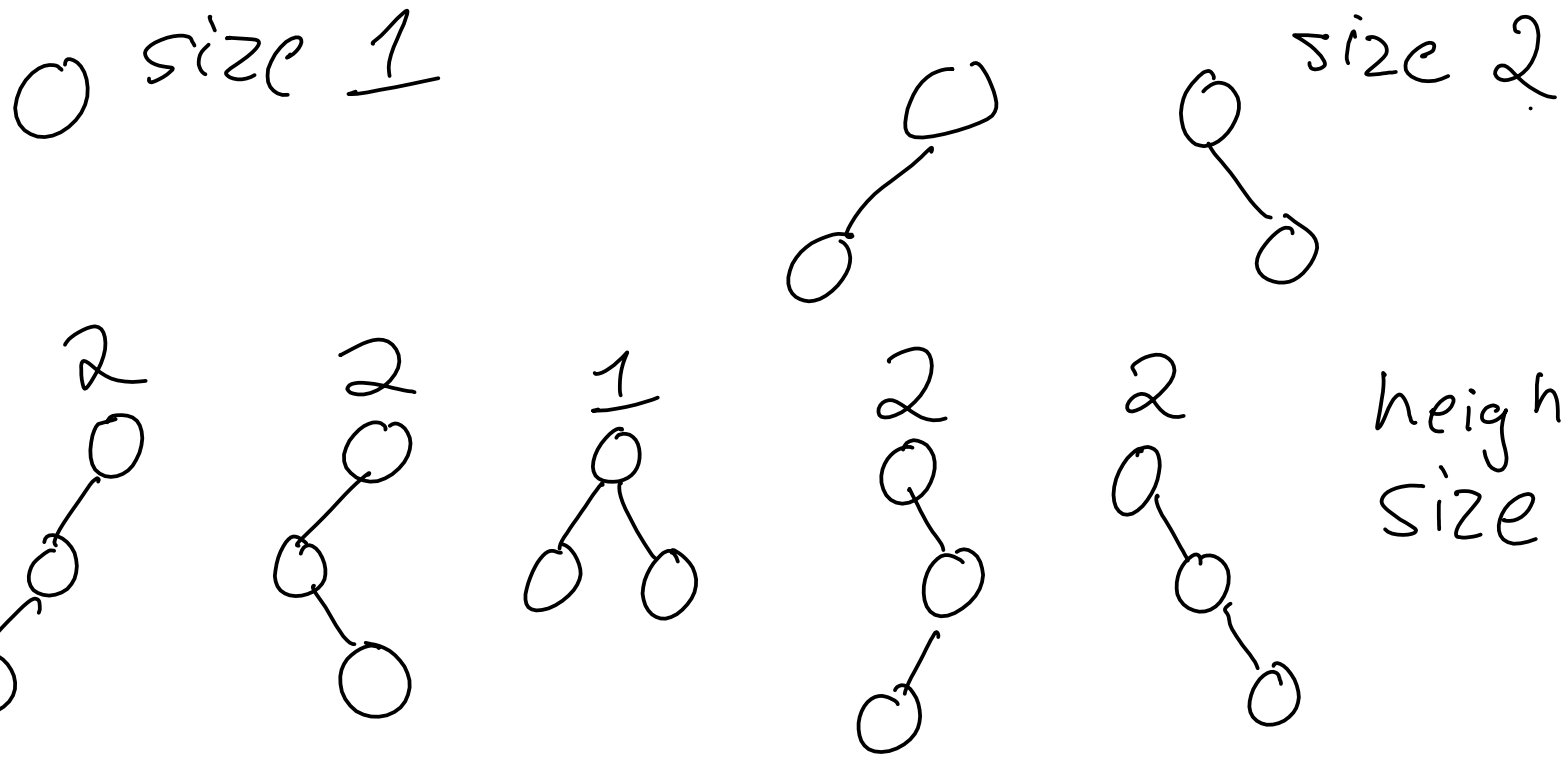
Each node has at most

1 "left" child

and 1 "right" child

Node w/ no children: "leaf"

Node w/ no parent: "root"



In lab

LinkedBSTNode

Binary Search Tree

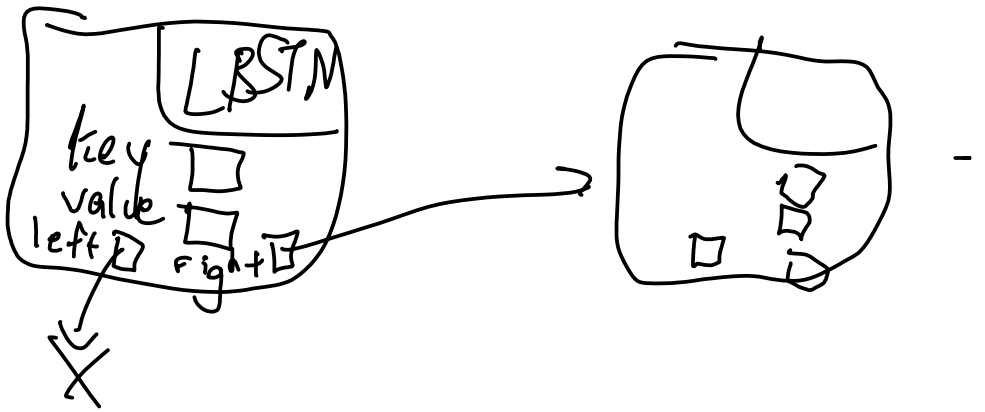
BST is a kind of binary tree

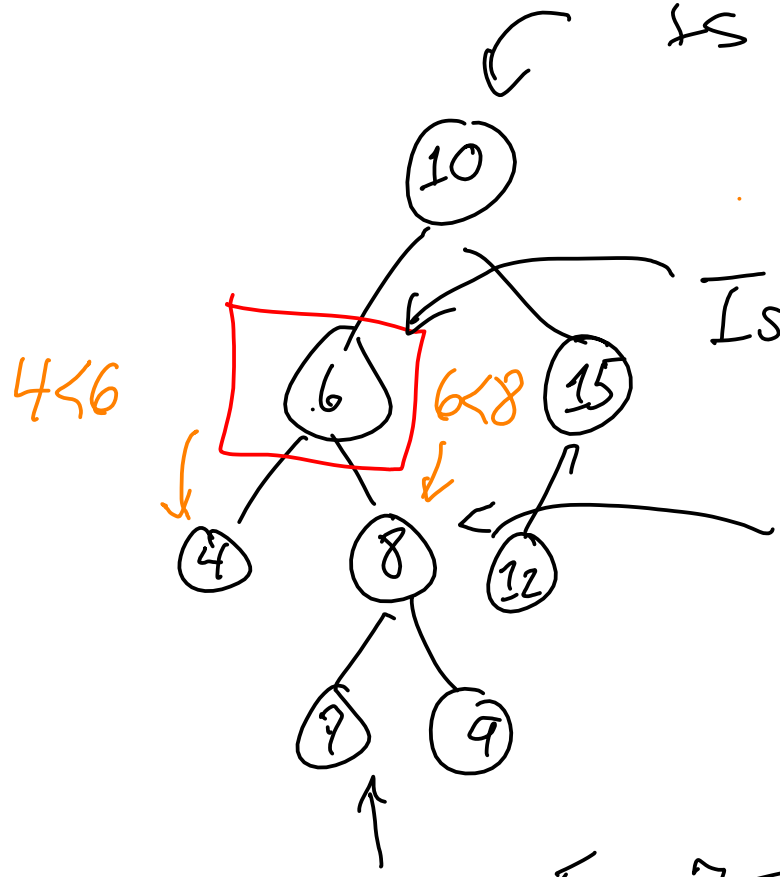
```
template <typenamee K, typename V>
class LinkedBSTNode
....
```

User object

```
private:
    K key;    V value;
    LinkedBSTNode<K,V> *  left;
                          right;
```

username





Is 10=7 no

10<7 no => go left

Is 6=7 no

6<7 yes => go right

Is 8=7 no

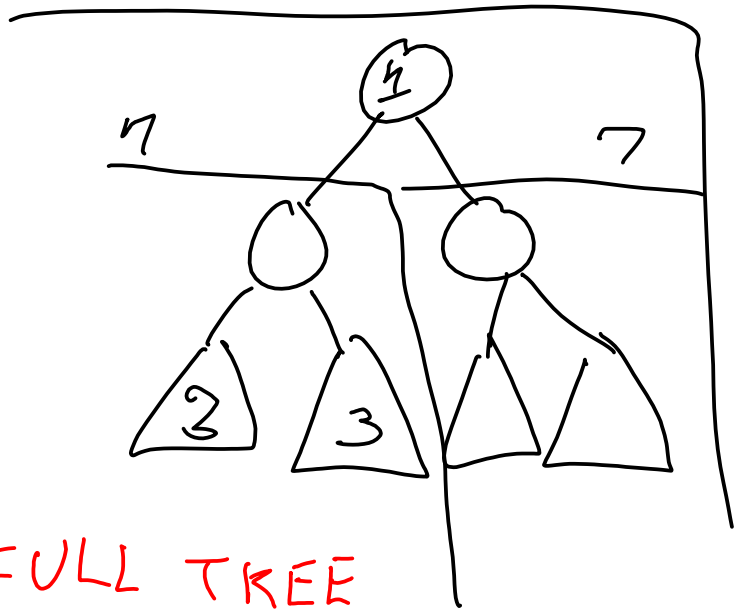
Is 8<7 no => go left

Is 7=7 yes!

Full tree: biggest tree for its height

Full → ○ not Full → ○

Full → ○ (○ ○)



15

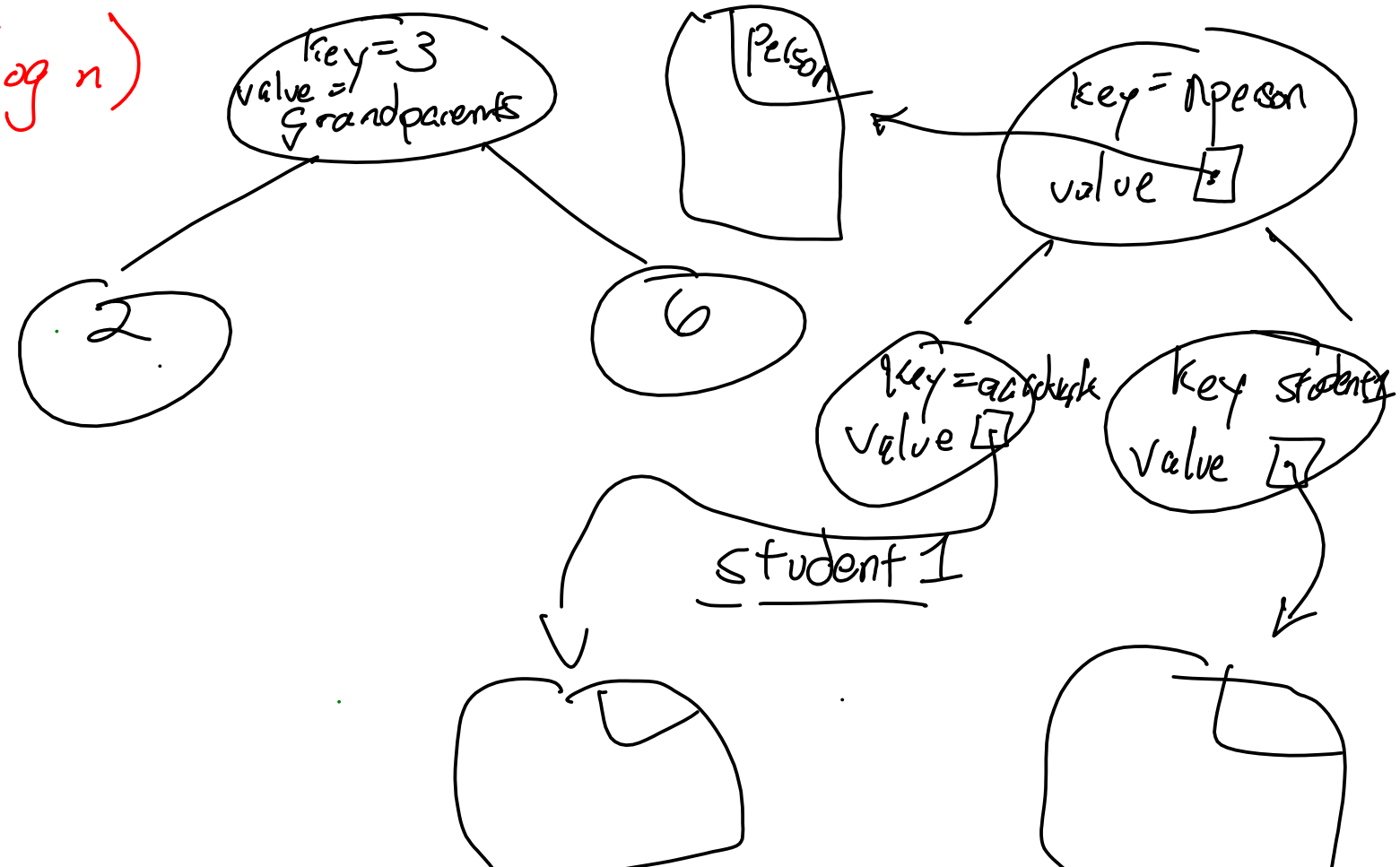
Based on some "key", find my data fast

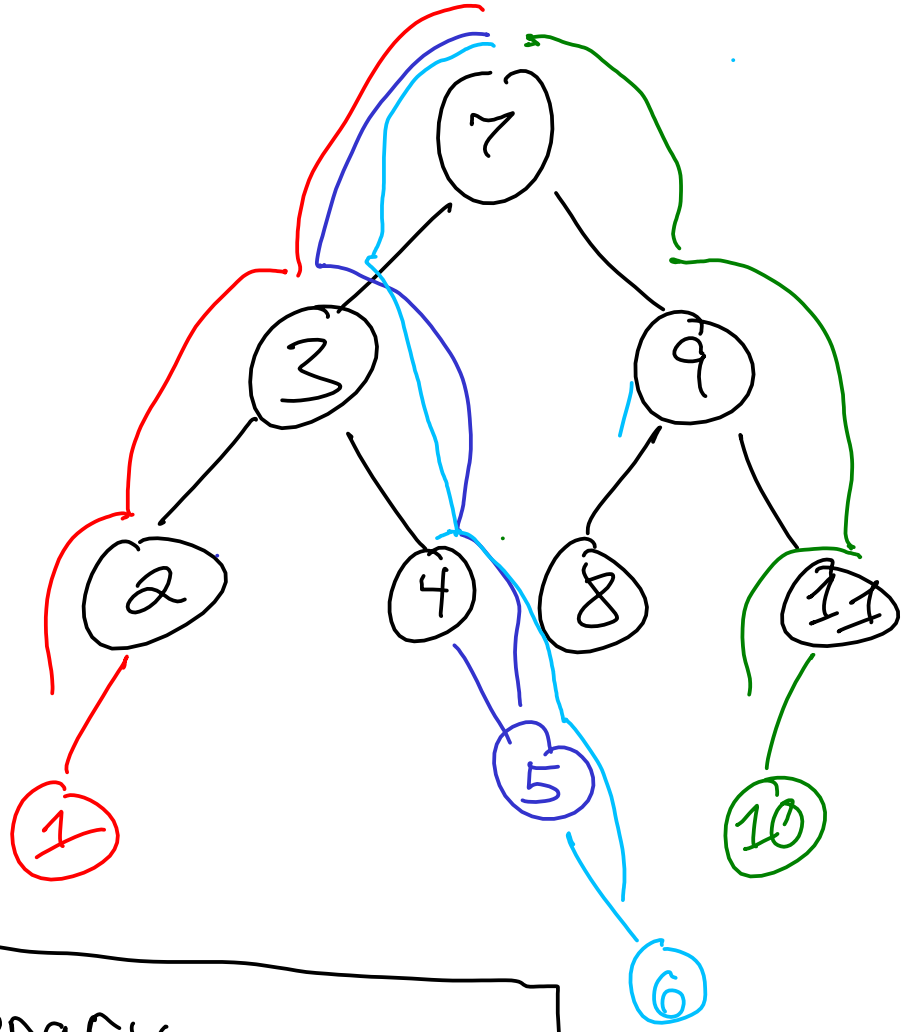
Find (get) in BST is O(log n)

FULL TREE

get O(log n)

insert ?





insert 5

insert 1

insert 10

insert 6

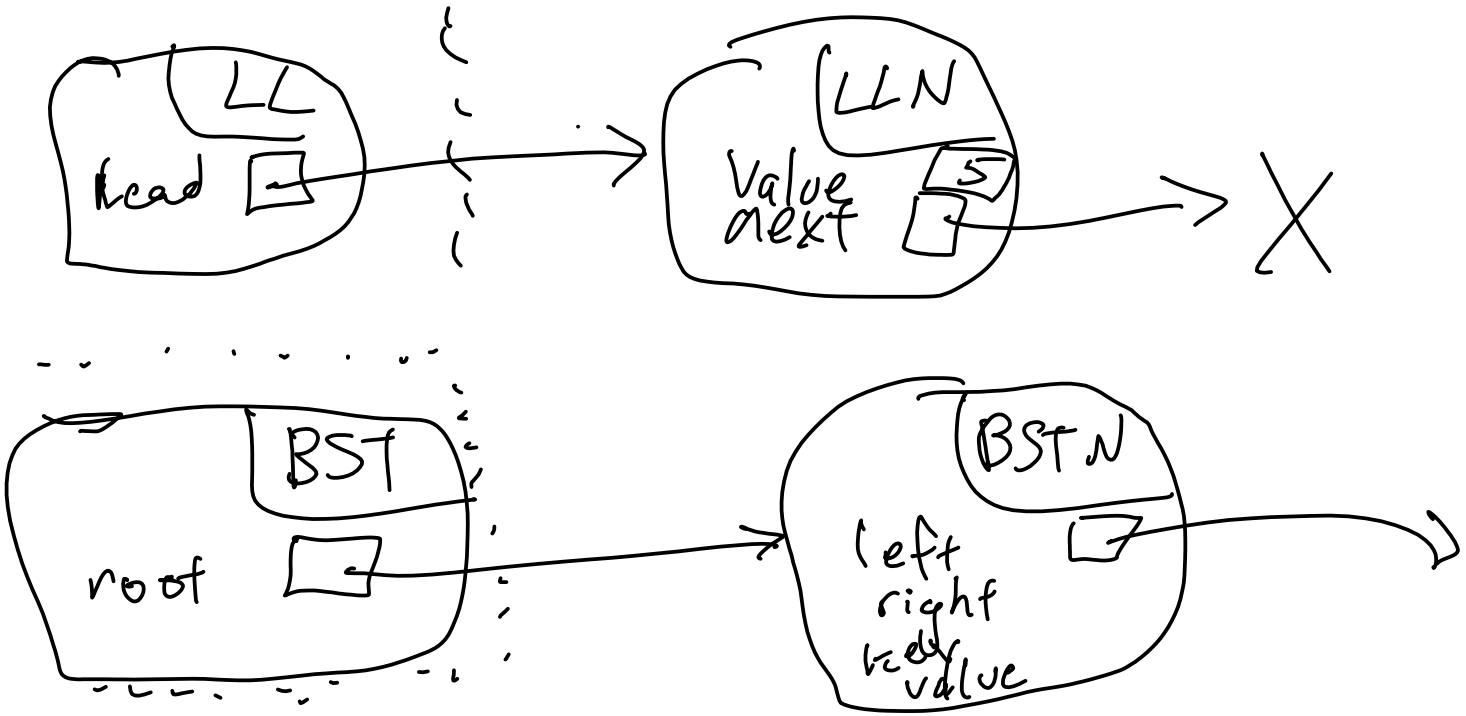
Pictionary
V get(K key) void put(K key, V value) V remove(K key)
get_size()

"dictionary"

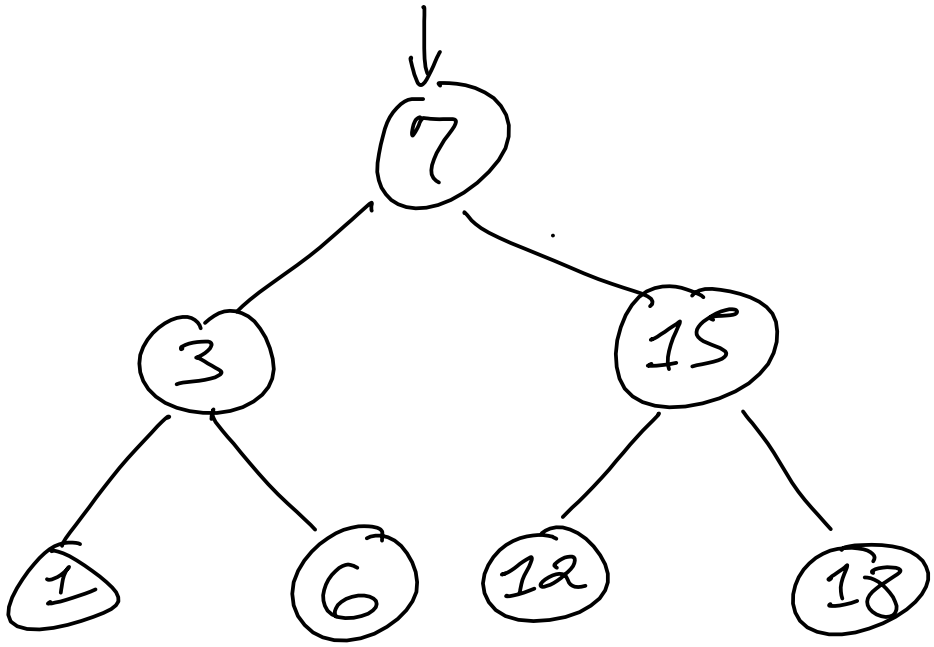
"finite map"

"map"

~~"hash"~~


LL
head
LLN
value
next
5
BST
root
BSTN
left
right
key
value

Traversal



In-order : Traverse left
Add node
Traverse right

1, 3, 6, 7, 12, 15, 18

Post-order
Traverse left
Traverse right
Add node

1, 6, 3, 12, 18, 15, 7

Pre-order
Add node
Trav Left
Trav Right

Traversal

Level-order