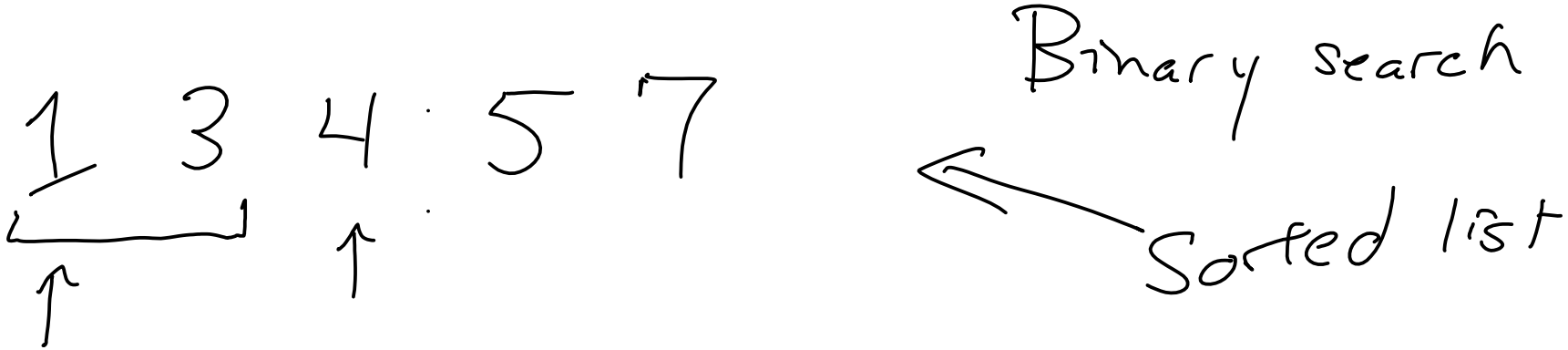
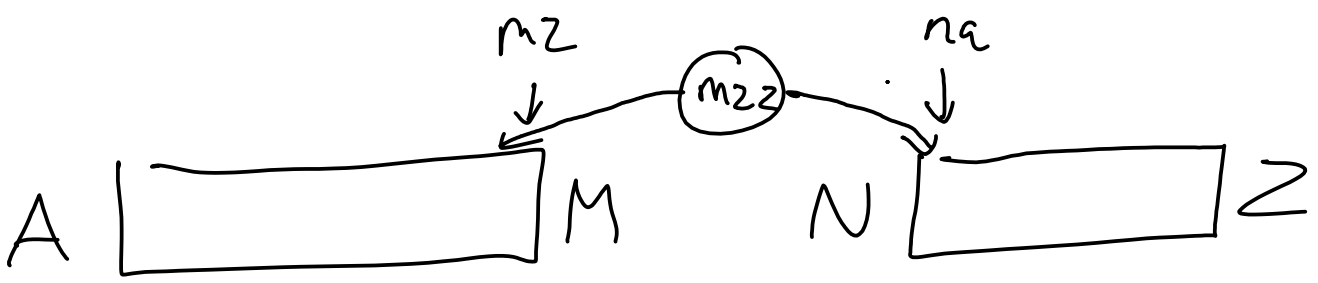
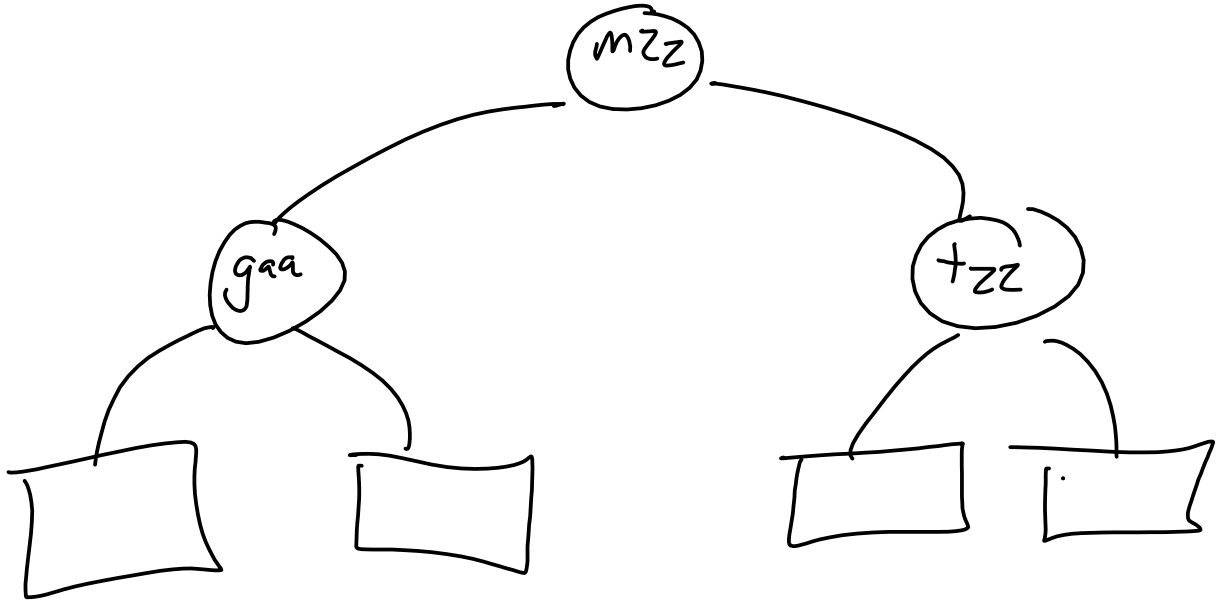
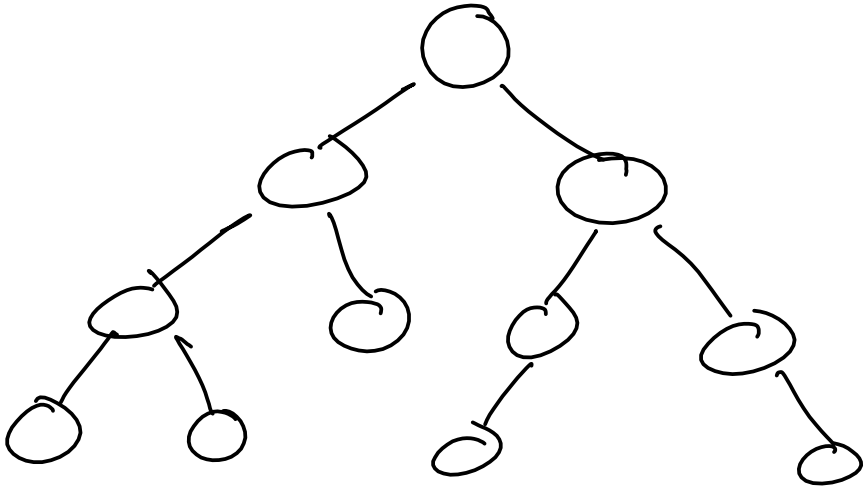


1 5 4 3 7


mz
nq
mzz
A
M
N
Z


mzz
gaa
tzz



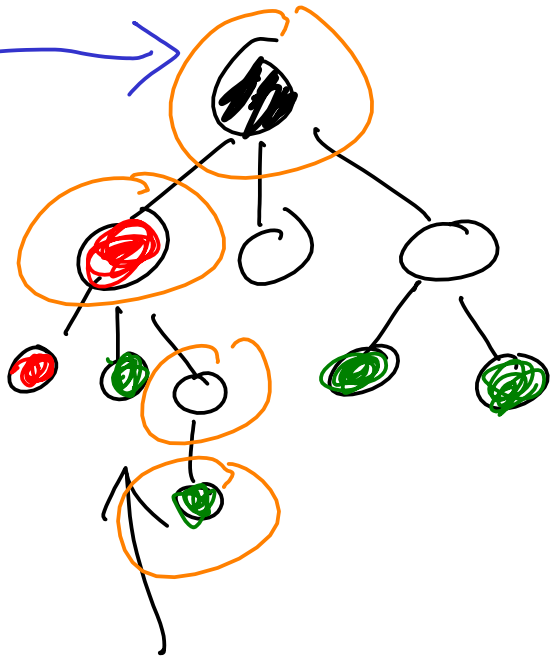
Tree — data structure

made of nodes

Nodes have children

and a parent

(except root)



H=1

: Leaves

Leaf is node w/ no children

○ H=∅

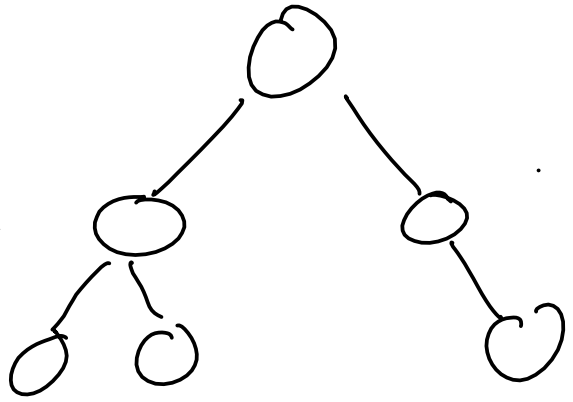
Size is # of nodes ← same as LL size ... H=-1

Height is max # of nodes on a path from root

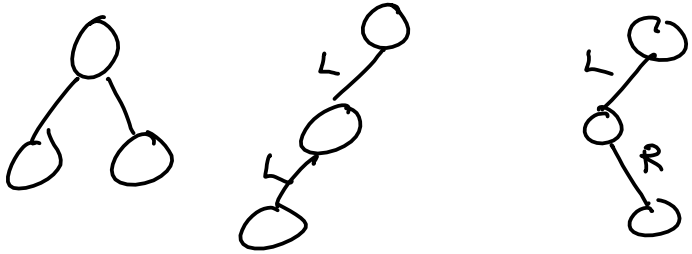
not counting root

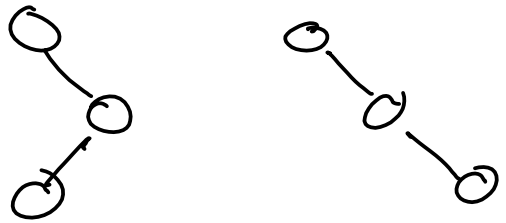
Binary Tree

is a tree with each node having max 2 children and each child is either left or right (unique)



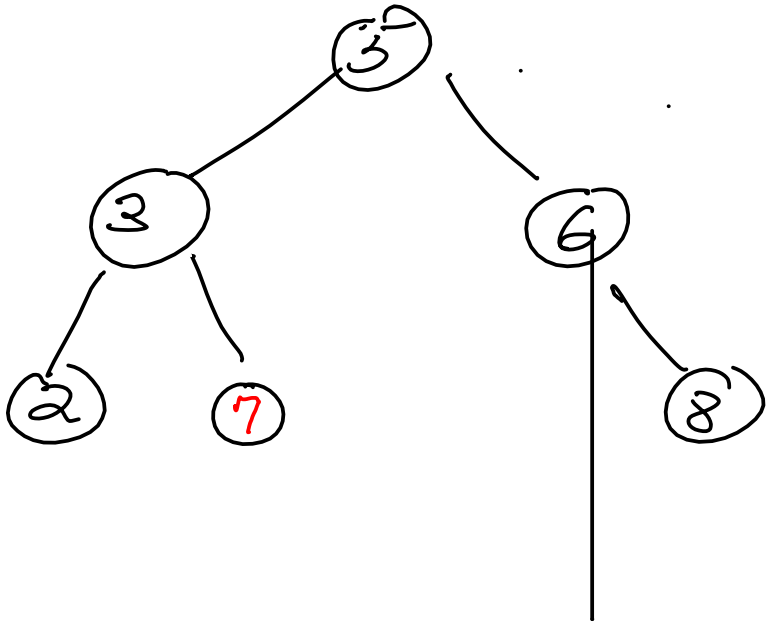
All binary trees of size 3





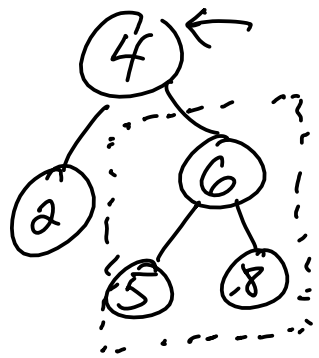
Binary Search Tree

BST is a binary tree where, for each node, all data in left subtree is less than data here; all data in right subtree is greater



ADT — Dictionary

two types
K, V
(key) (value)



V get(K key)

void insert(K, V) } put

V update(K, V) } put

V remove(K)

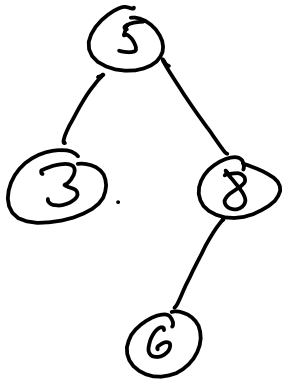
isEmpty()

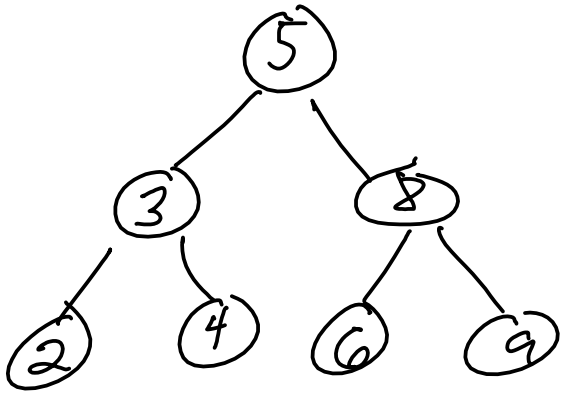
getSize()

List<K> getKeys()

```
Function insert(Node n, K key, V value):
  If n->key < key:
    If n->right == NULL:
      n->right <- new Node(key, value)
    Else:
      insert(n->right, key, value)
  Else If n->key > key:
    If n->left == NULL
      n->left <- new Node(key, value)
    Else
      insert(n->left, key, value)
  Else:
```

Traversal is a walk through a data structure





2 3 4 5 6 8 9

In-order traversal

Start at root

Traverse left

Visit Root

Traverse right