ADT - List<T>

... get(int index)

List<int>

[5, 7, 8, 2]

ADT — Dictionary<K, V> (K: unique key, V: value)

a dictionary is a collection of key-value mappings

all keys are unique

V get(K key)

void insert(K key, V value)

void update(K key, V value) } = put(K, V)

V remove(K key)

int getSize()

bool contains(K key)

List<K> getKeys()

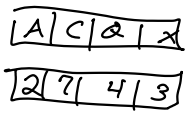
Dictionary<string, int>

{ "a" ↦ 5,
"b" ↦ 7,
"cat" ↦ 2,
"dog" ↦ 7
}

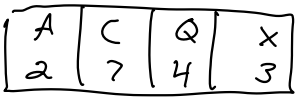
for today: assume K is ordered

ParallelListDictionary

- list of keys
- list of values
- at a given index, key↦value



ArrayDictionary

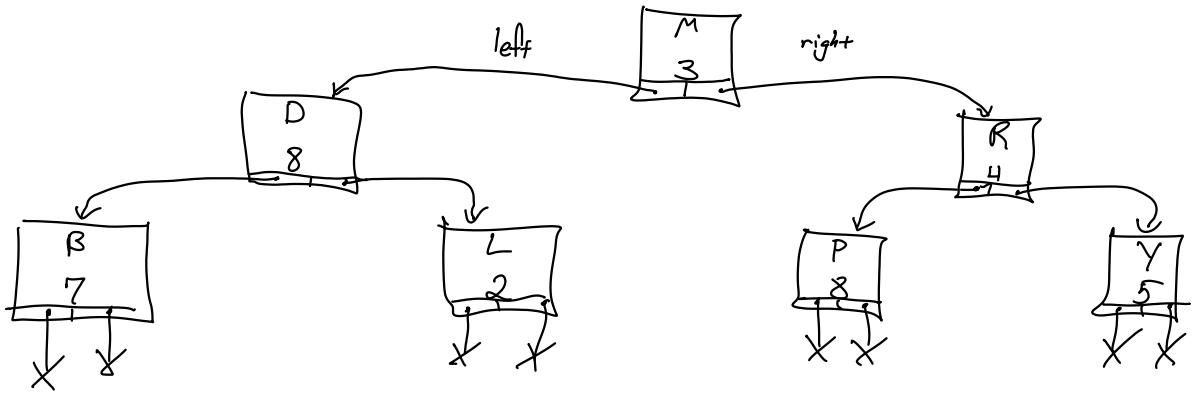


SortedArrayDictionary

+ invariant: array is always sorted

get(K): O(log n)

insert(K, V): O(n)



Tree*: a collection of nodes related to each other in the form above

Node*: a member in a tree

Root: the node with no nodes above it (no parent)

Parent: the node immediately above a given node

Child: a node below a given node

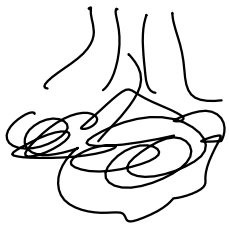
Leaf: a node with no children

Descendant: any node below a given node

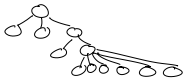
Ancestor: any node above a given node

Size: the number of nodes

Height: the number of steps between root & furthest leaf



-1 ∅ 1 2 3



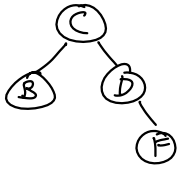
Binary tree: a tree where each node has at most one left child and one right child

Binary Search Tree (BST): a binary tree where

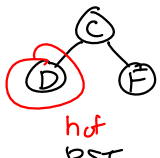
the key in any node is greater than the key in all of its left descendants and the key in any node is less than the key in all of its right descendants

BST invariant

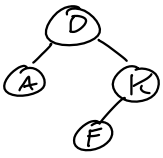
BST



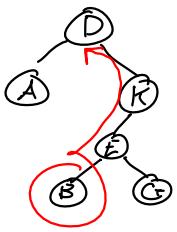
BST



not a BST

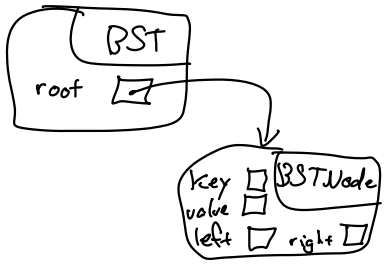


BST



not a BST

```
Method get(K key):
    Return getInSubtree(key, this->root)
End Method
```




```
Method getInSubtree(K key, BSTNode<K,V>* node):
    If(node == nullptr):
        !!
        ^
    Else If (key == node->key):
        Return node->value;
    Else If (key < node->key):
        Return getInSubtree(key, node->left)
    Else:
        Return getInSubtree(key, node->right)
    End If
End Method
```