FIT9136 Week 7

Binary Tree and BST

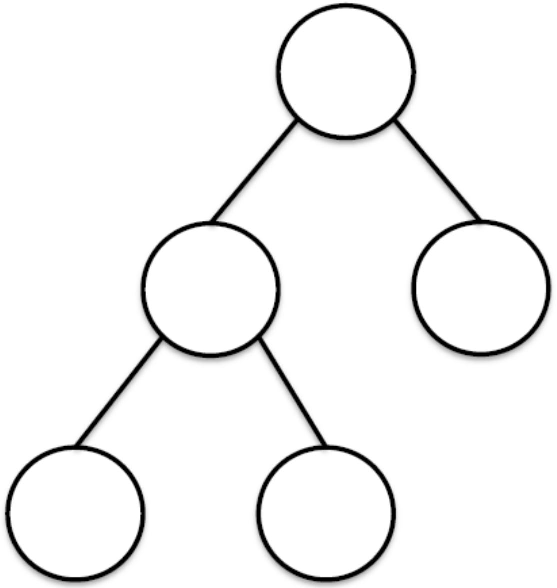
授课老师：Joe

FIT9136平时班 – Week 7

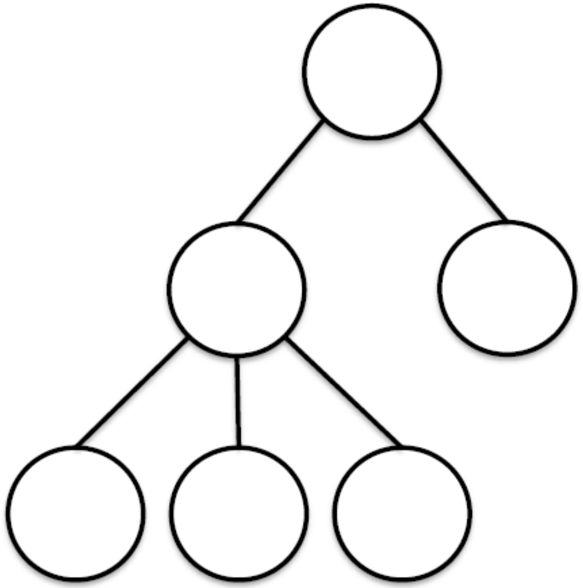
```
class Node:

    def __init__(self, item, left = None, right = None):
            self.item = item
            self.left = left
            self.right = right
```

- A binary tree is a tree whose nodes have at most 2 children.



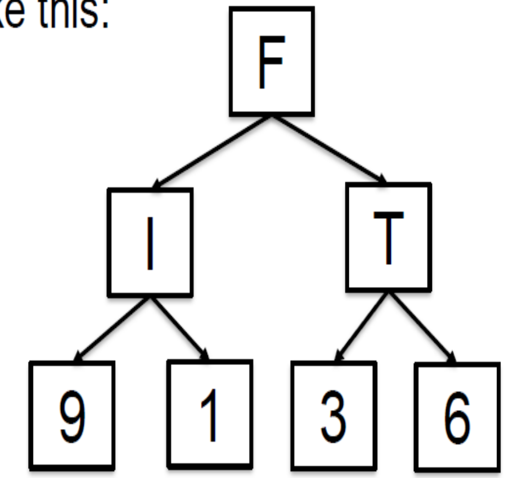
Binary Tree



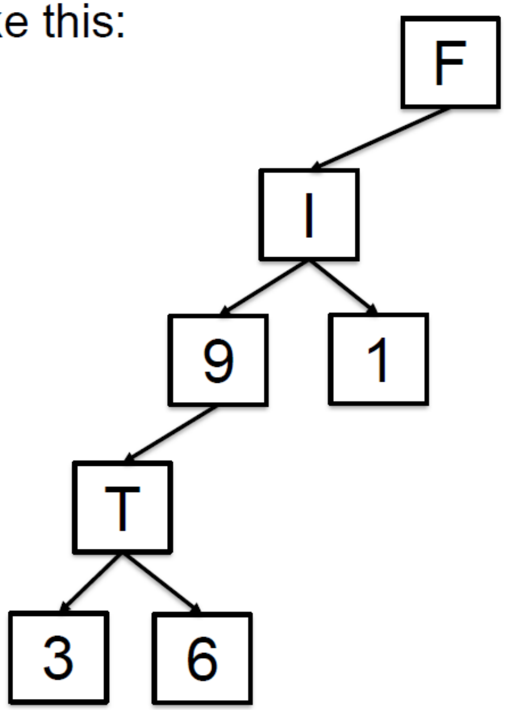
Non-Binary Tree

FIT9136平时班 – Week 7

Using a Binary Tree structure: our unit's name could look like this:

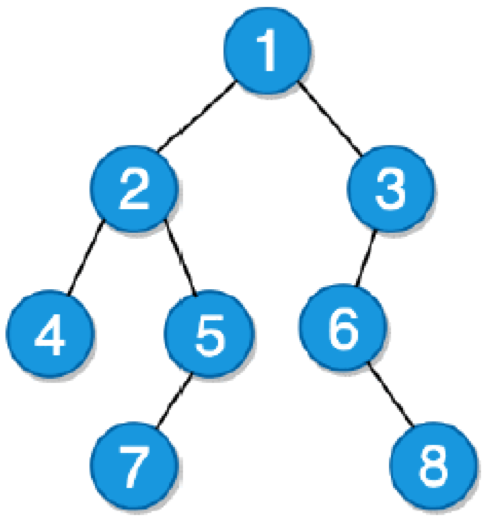


- Or like this:

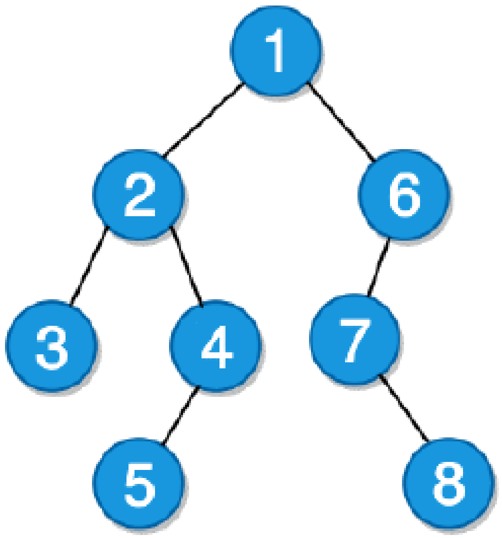


Binary Search Tree

BFS

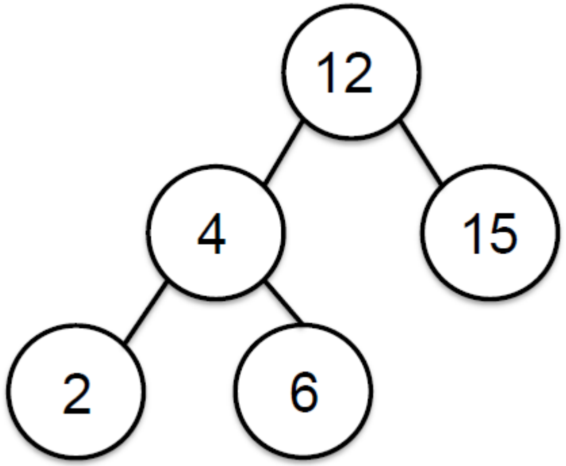


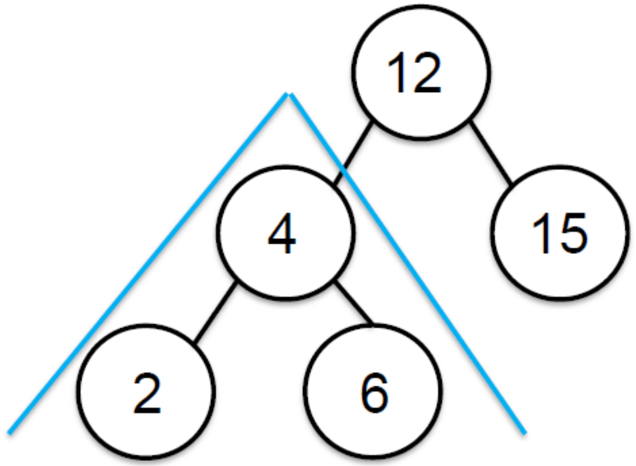
DFS



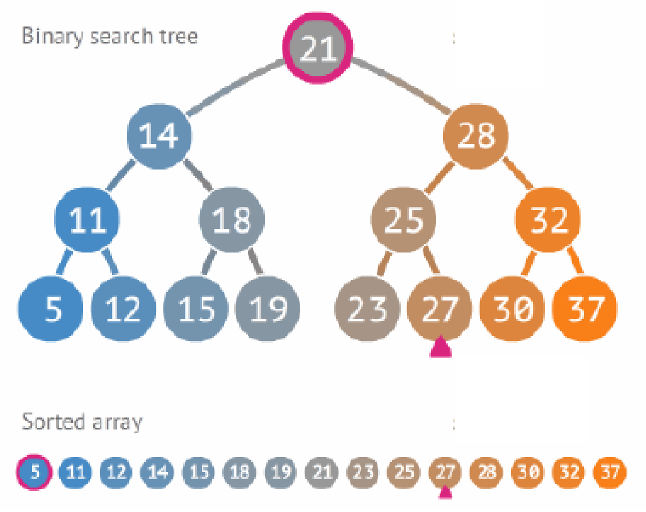
- **Binary Search Tree (BST)** is a node-based binary tree data structure which has the following properties:
  - The left subtree of a node contains only nodes with values smaller than the root.
  - The right subtree of a node contains only nodes with values larger than the root.
  - The left and right subtree each must also be a binary search tree

FIT9136平时班 – Week 7



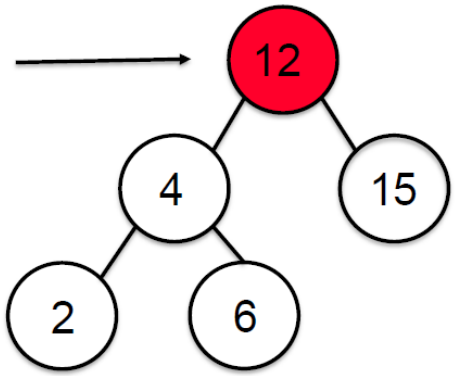


Binary Search Trees is much more efficient for searching one element in a list

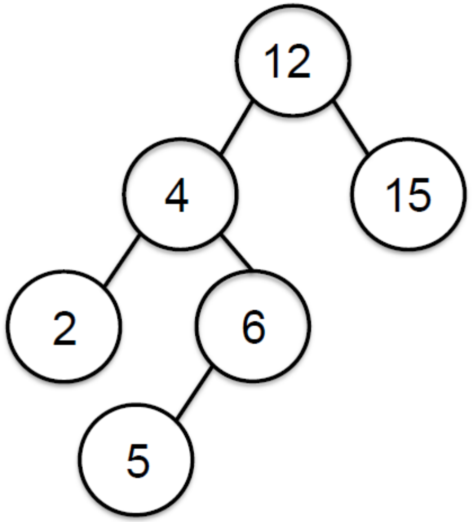


FIT9136平时班 – Week 7

First find the location. This is what we have already done. Lets try to add 5.



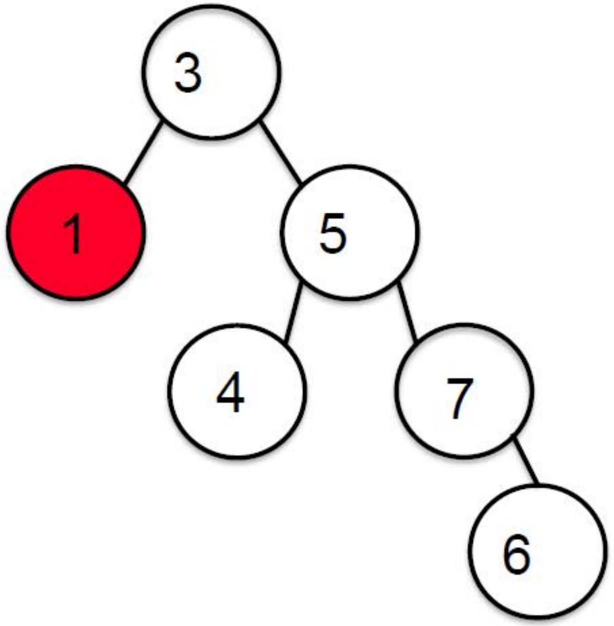
- Is the root what we want? No.
- Is 5 > 12? No.
  – 5 must be inserted to the left of 12.
- Go left.



- The left node does not exist.
- We have found the location for 5.
- Insert 5 here.

FIT9136平时班 – Week 7

We have the list: [3,5,4,7,6,1]



What happens if we try to insert an element already there?

- When we try to search for a location, if we come across the item in a node, stop searching.
- Do not insert two elements of the same value into a search tree.

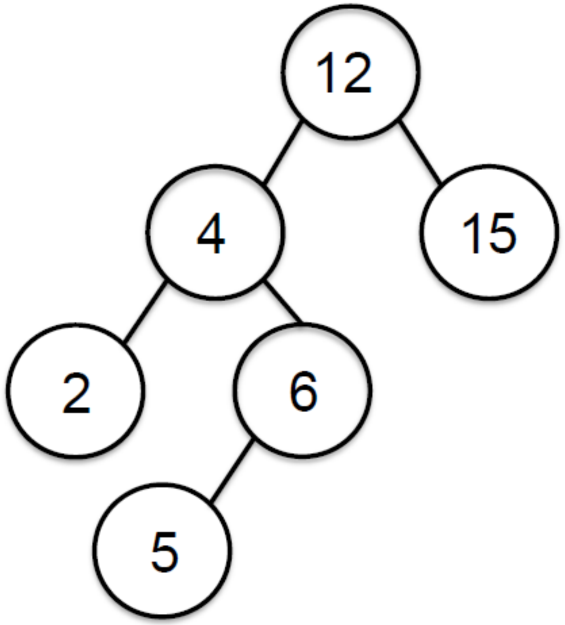
No Duplicates !

To enable fast search in a BST, balance is the key!

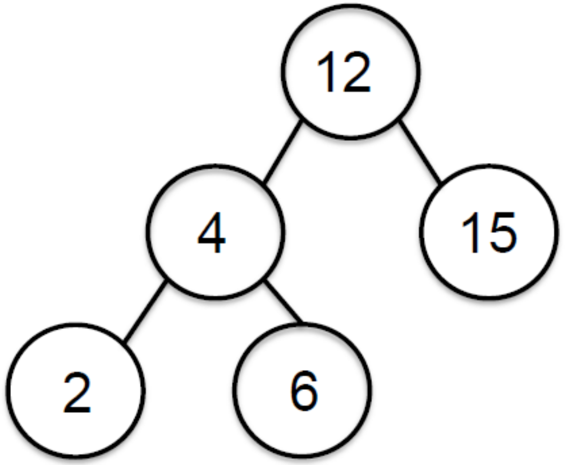
Balance =
Depth of left subtree – depth of right subtree =
+/- 1

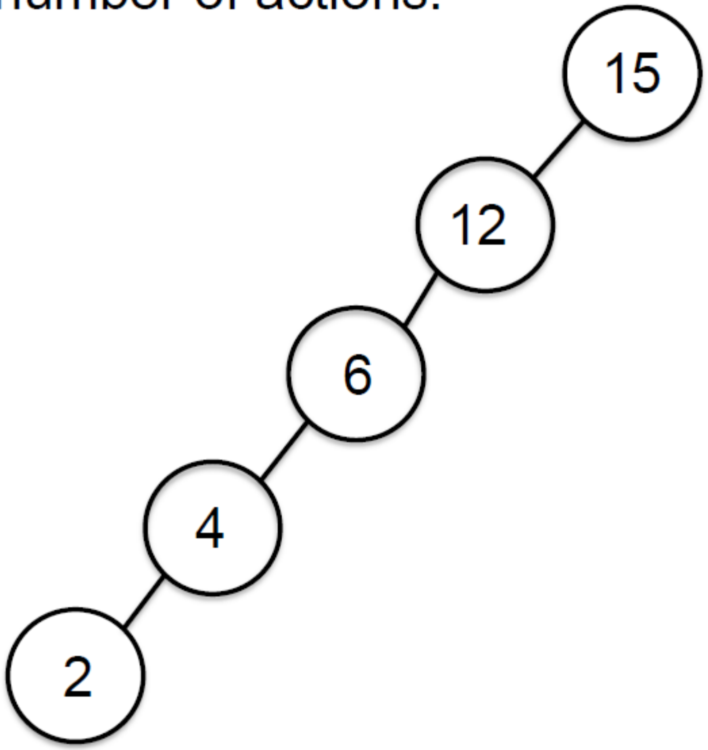
FIT9136平时班 – Week 7

The tree we just made is considered NOT balanced.



- Search for 2, count the number of actions.





- For a balanced tree, the complexity is $\boldsymbol{log_2 n}$ because with every step down the tree, the search space possibilities are halved.
- For an imbalanced tree, the complexity is $\boldsymbol{n}$ because probing for an element that is not there, in the worst case, will require each element to be examined.