Trees

Ivan Velkov

2020

What is a tree

Non linear data structure

Recursive

Set of Nodes

Definitions

Root

Edge

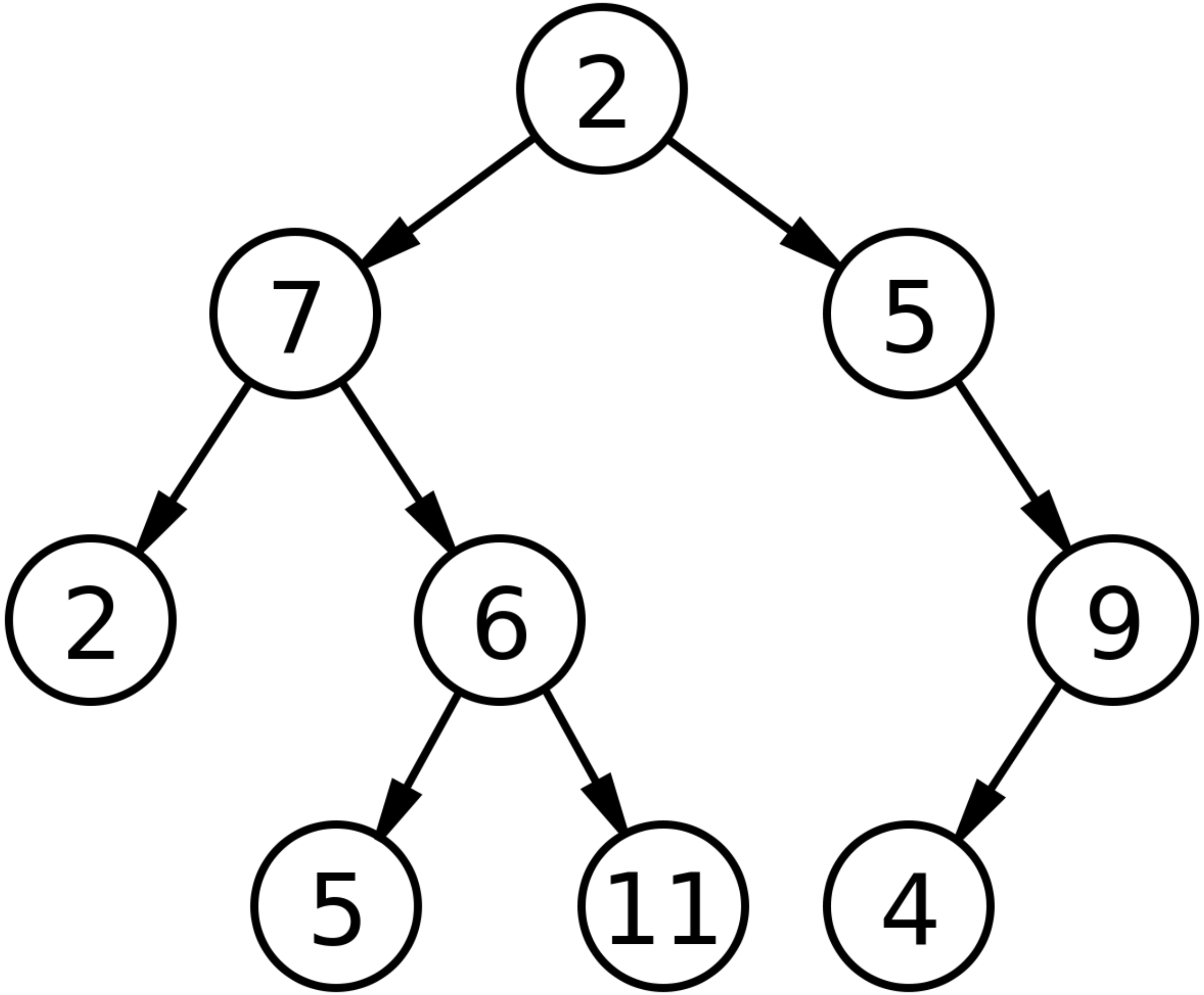
Child

Parent

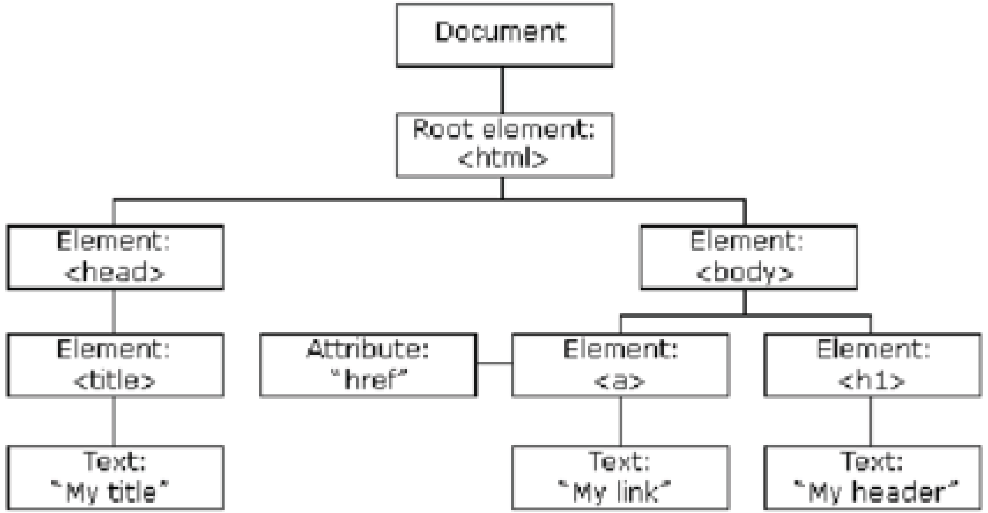
Leaf

Height

Depth

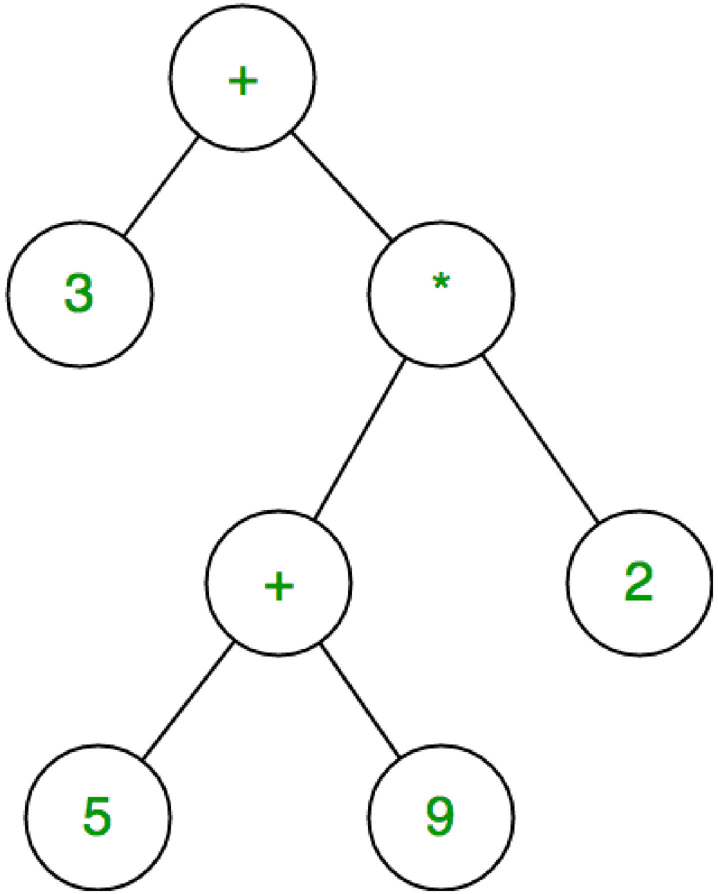


HTML

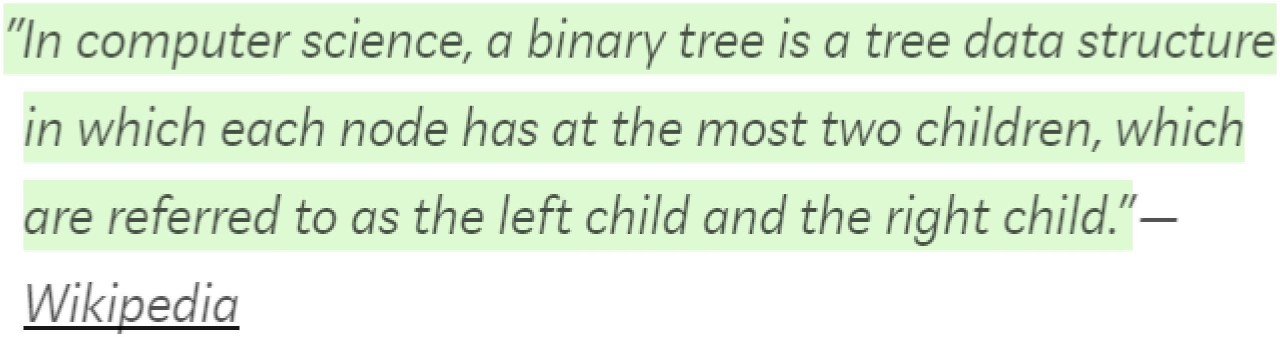


```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="“">My link</a>
    <h1>My header</h1>
  </body>
</html>
```

Arithmetic expressions

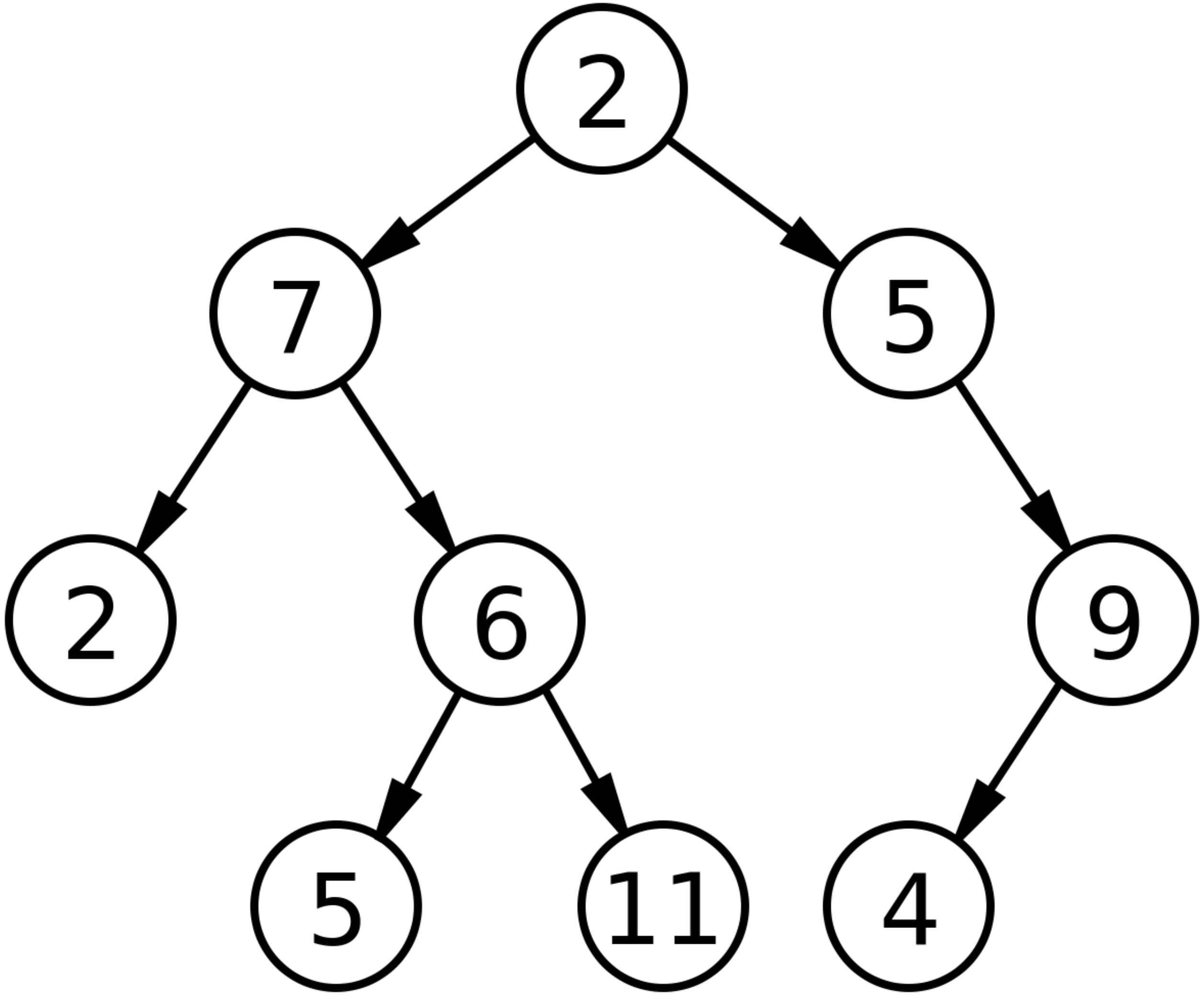


Binary Tries



"In computer science, a binary tree is a tree data structure in which each node has at the most two children, which are referred to as the left child and the right child."— Wikipedia

Binary Tries



Binary Tries

```
class Node
{
        int key;
        Node left, right;
        public Node(int item)
        {
                key = item;
                left = right = null;
        }
}
```

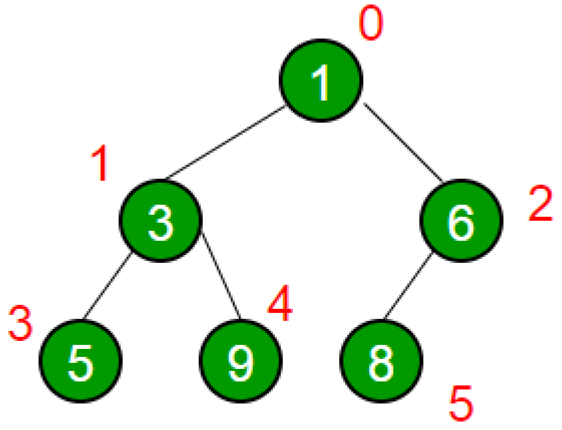
Array representation

Used for complete trees

Arr[(i – 1) / 2] = parent node

Arr[(2 * i) + 1] = left child

Arr[(2 * i) + 2] = right child

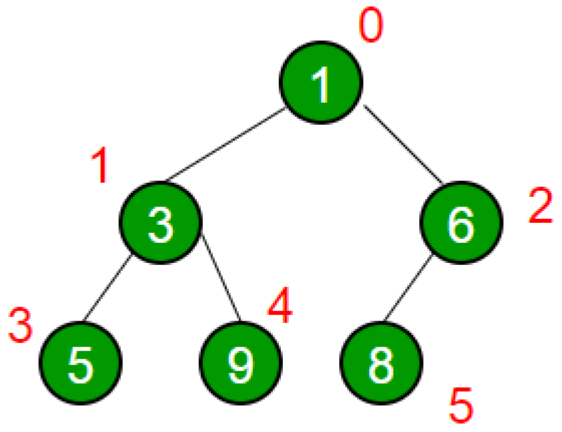


1	3	6	5	9	8
0	1	2	3	4	5

Binary Heap

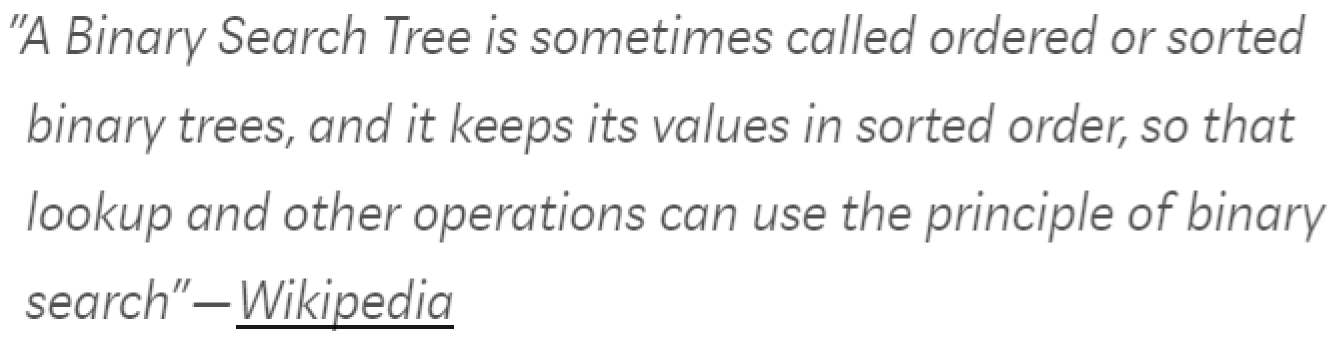
Used for priority queue

Heapsort – n log n



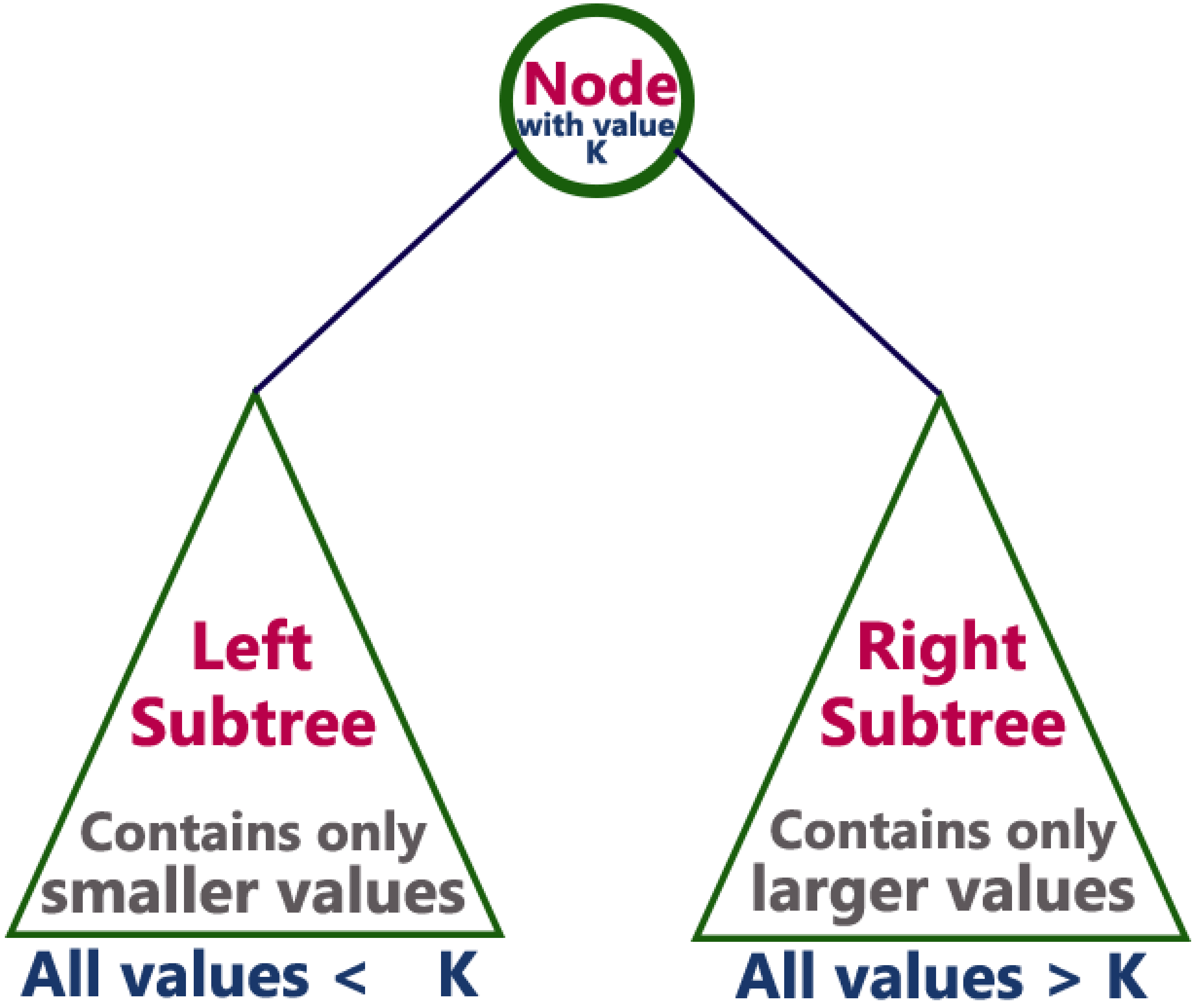
1	3	6	5	9	8
0	1	2	3	4	5

Binary Search Tree



"A Binary Search Tree is sometimes called ordered or sorted binary trees, and it keeps its values in sorted order, so that lookup and other operations can use the principle of binary search"—Wikipedia

Binary Search Tree



Binary Search Tree

Insertion

Search

Deletion

- No children
- 1 child
- 2 children

Binary Search Tree

Insertion – O(h)

Search – O(h)

Deletion – O(h)

h - Average O(log n) Worst O(n)

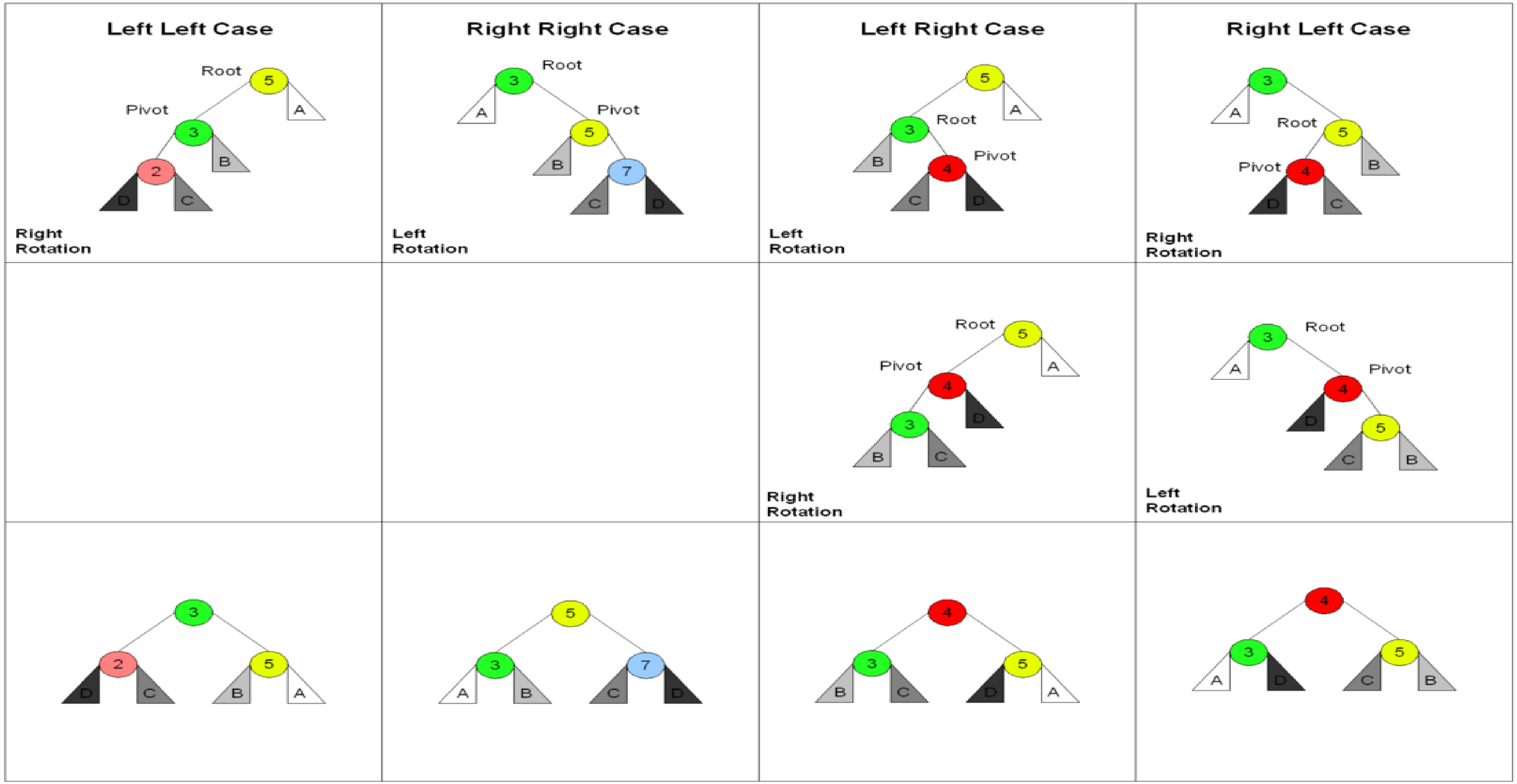
Balanced tree

Balanced tree
Perfectly balanced tree
Rotations

Rotations

There are 4 cases in all, choosing which one is made by seeing the direction of the first 2 nodes from the unbalanced node to the newly inserted node and matching them to the top most row.

Root is the initial parent before a rotation and **Pivot** is the child to take the root's place.



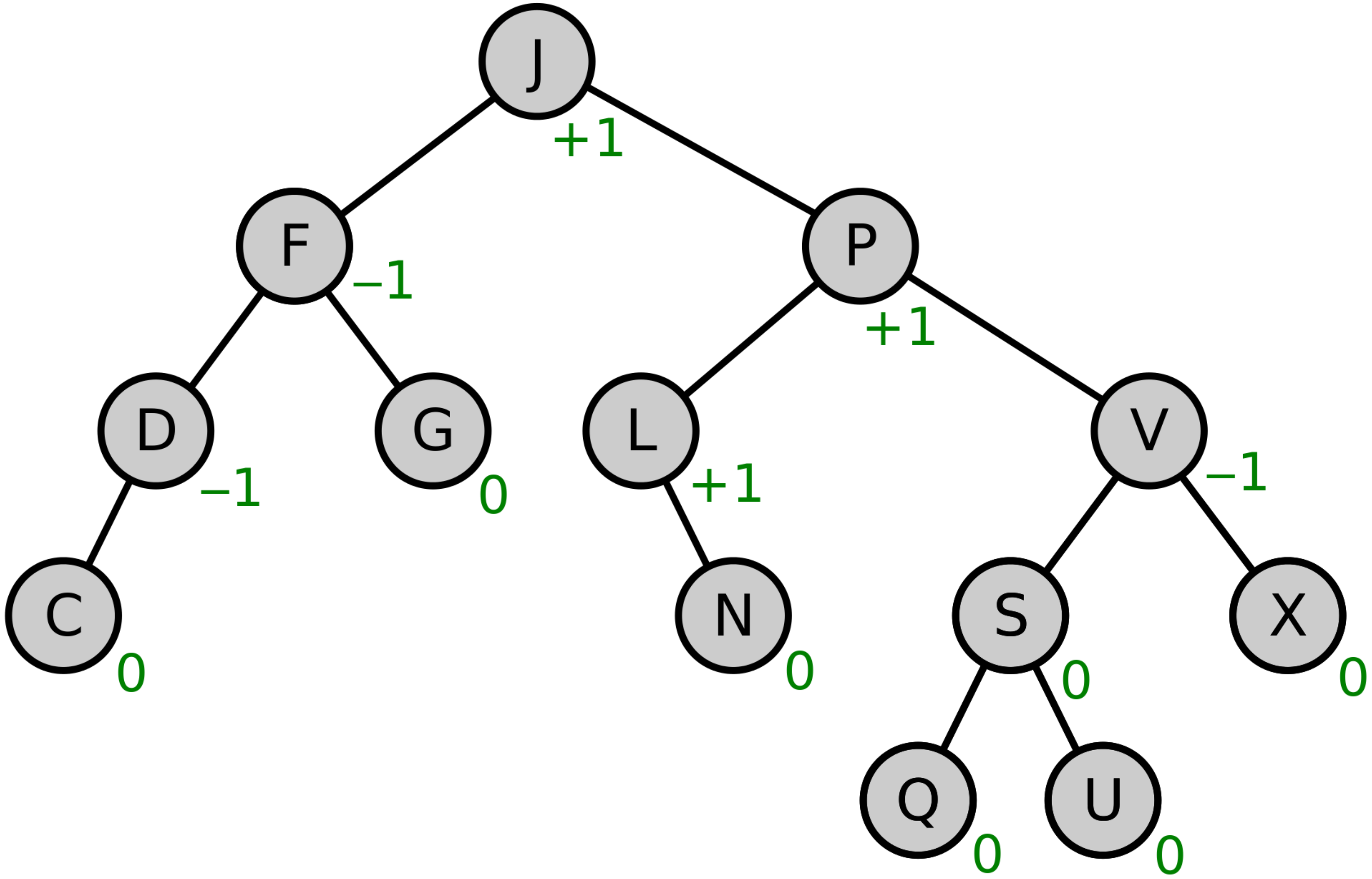
AVL tree

Self balancing tree – automatically keeps its height low

BalanceFactor(N) := Height(RightSubtree(N)) – Height(LeftSubtree(N))

BalanceFactor(N) ∈ {-1, 0, 1}

AVL tree



Homework

Modify the BST from the lecture into an AVL tree.

What to keep in the node

Value – mandatory

Children – mandatory

Height?

Parent?

Count of Nodes in the subtree?

Tons more depending on the situation

Tree Traversal

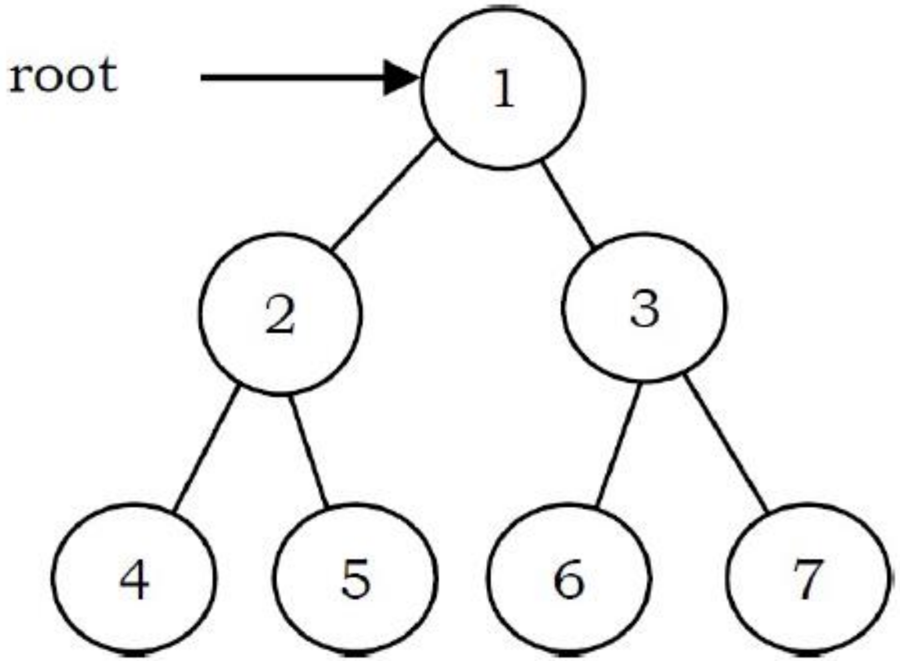
BFS – 1 2 3 4 5 6 7

DFS – 1 2 4 5 3 6 7

Pre order – Same as DFS

In order – 4 2 5 1 6 3 7

Post order – 4 5 2 6 7 3 1



Task

Given two BSTs (Binary Search Tree) that may be unbalanced, convert them into a balanced BST that has minimum possible height.

Note: You don’t have to keep the two input trees.

Trees usage

- Maps – Set of unique values
- Dicts – Set of unique keys and values

Strategies for repeating keys

End of Presentation
ANY QUESTIONS?
memecreator.org

Thank You