Transform & Conquer

Transform-and-Conquer

1. **Transformation stage:** the problem's instance **is modified** to be **more tractable** to solution.

2. **Conquering stage:** the problem is solved.

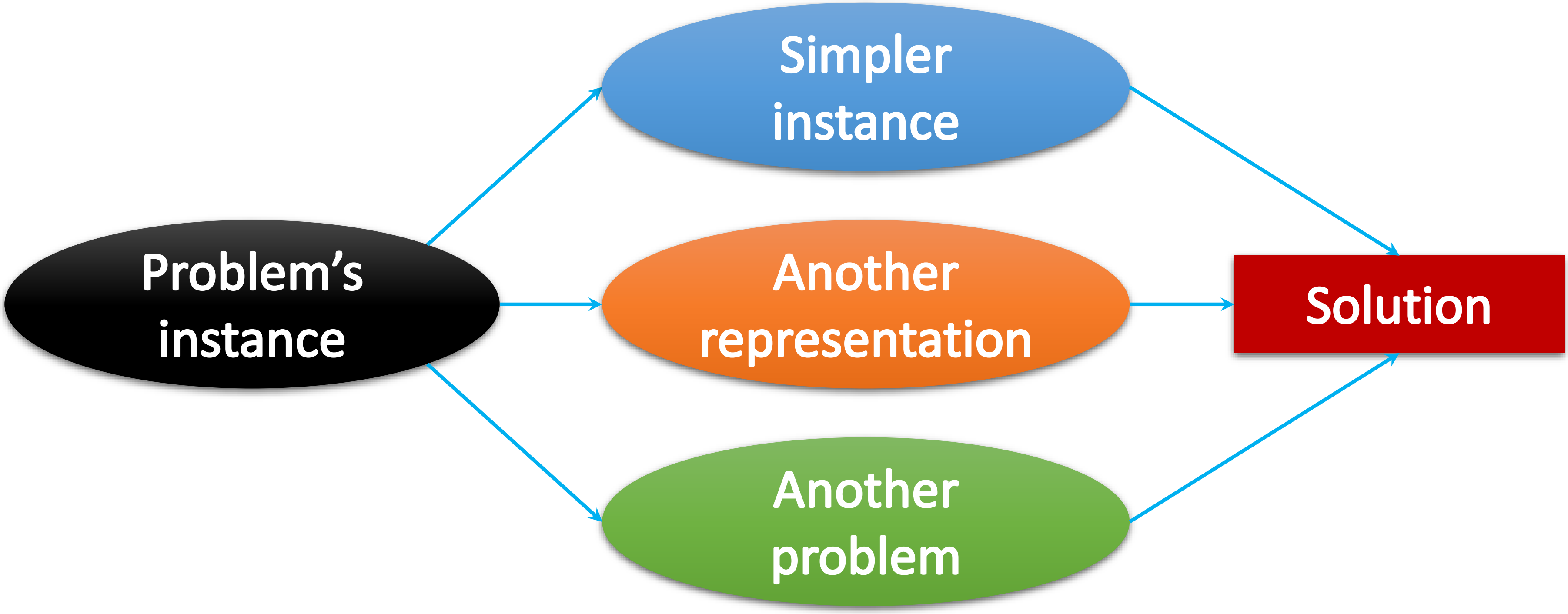
Transformation Types

1. **Instance simplification**: transformation to a simpler or more convenient instance of the same problem.

2. **Representation change**: transformation to a different representation of the same instance.

3. **Problem reduction**: transformation to an instance of different problem for which an algorithm is already available.
   [Computation & Complexity]

Transform-and-Conquer



Presorting

Example: Checking element uniqueness in an array

Presorting

Example: Checking element uniqueness in an array

1. Sort the array
2. Check only its consecutive elements

Presorting

Example: Checking element uniqueness in an array

1. Sort the array
2. Check only its consecutive elements

Sorting algorithms:

- Selection/Bubble sort: $\Theta(n^2)$
- Merge sort: $\Theta(n \log n)$
- Quicksort: $\Theta(n^2)$ (worst case), $\Theta(n \log n)$ (avg case)
- Counting/Radix sort: $\Theta(n)$

Presorting

Algorithm $\mathrm{PresortElementUniqueness}(A[0..n-1])$

// Input: $A[0..n-1]$

//Output: return True if A has no equal elements,
False otherwise

sort the array A

for $i \leftarrow 0$ **to** $n-2$ **do**

  if $A[i] = A[i+1]$

    return False

return True

Presorting

Time efficiency:

$$T(n) = T_{sort}(n) + T_{scan}(n)$$

$$= \Theta(n \log n) + \Theta(n)$$

$$= \Theta(n \log n)$$

More examples:

1. Computing a mode, median, …
2. Searching problem

Horner's Rule

Problem: Compute the value of a polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \qquad (1)$$

at a given point x_0.

Horner's rule is an example of the **representation-change** technique:

$$p(x) = (\cdots(a_n x + a_{n-1})x + \cdots)x + a_0 \qquad (2)$$

Horner's Rule

- (2) is obtained from (1) by successfully taking x as a **common factor** in the remaining polynomials of diminishing degrees:

$$p(x) = 2x^4 - x^3 + 3x^2 + x - 5$$

Horner's Rule

- (2) is obtained from (1) by successfully taking x as a **common factor** in the remaining polynomials of diminishing degrees:

$$\begin{aligned} p(x) &= 2x^4 - x^3 + 3x^2 + x - 5 \\ &= x(2x^3 - x^2 + 3x + 1) - 5 \end{aligned}$$

Horner's Rule

- (2) is obtained from (1) by successfully taking x as a **common factor** in the remaining polynomials of diminishing degrees:

$$\begin{aligned} p(x) &= 2x^4 - x^3 + 3x^2 + x - 5 \\ &= x(2x^3 - x^2 + 3x + 1) - 5 \\ &= x(x(2x^2 - x + 3) + 1) - 5 \end{aligned}$$

Horner's Rule

- (2) is obtained from (1) by successfully taking x as a **common factor** in the remaining polynomials of diminishing degrees:

$$p(x) = 2x^4 - x^3 + 3x^2 + x - 5$$
$$= x(2x^3 - x^2 + 3x + 1) - 5$$
$$= x(x(2x^2 - x + 3) + 1) - 5$$
$$= x(x(x(2x - 1) + 3) + 1) - 5$$

Horner's Rule

Example: Evaluate $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$ at $x = 3$

$$\begin{aligned} p(x) &= 2x^4 - x^3 + 3x^2 + x - 5 \\ &= x(x(x(2x - 1) + 3) + 1) - 5 \end{aligned}$$

Coef.	2	-1	3	1	-5
$x = 3$					

Horner's Rule

Example: Evaluate $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$
at $x = 3$

$$p(x) = 2x^4 - x^3 + 3x^2 + x - 5$$
$$= x(x(x(2x - 1) + 3) + 1) - 5$$

Coef.	2	-1	3	1	-5
$x = 3$	2				

Horner's Rule

Example: Evaluate $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$
at $x = 3$

$$\begin{aligned} p(x) &= 2x^4 - x^3 + 3x^2 + x - 5 \\ &= x(x(x({\color{red}2x - 1}) + 3) + 1) - 5 \end{aligned}$$

Coef.	2	-1	3	1	-5
$x = 3$	2	$3 \cdot 2 + (-1) = {\color{red}5}$			

Horner's Rule

Example: Evaluate $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$ at $x = 3$

$$\begin{aligned} p(x) &= 2x^4 - x^3 + 3x^2 + x - 5 \\ &= x(x(x(2x - 1) + 3) + 1) - 5 \end{aligned}$$

Coef.	2	-1	3	1	-5
$x = 3$	2	$3 \cdot 2 + (-1) = 5$	$3 \cdot 5 + 3 = 18$		

Horner's Rule

Example: Evaluate $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$ at $x = 3$

$$p(x) = 2x^4 - x^3 + 3x^2 + x - 5$$
$$= x(x(x(2x - 1) + 3) + 1) - 5$$

Coef.	2	-1	3	1	-5
$x = 3$	2	$3 \cdot 2 + (-1) = 5$	$3 \cdot 5 + 3 = 18$	$3 \cdot 18 + 1 = 55$	

Horner's Rule

Example: Evaluate $p(x) = 2x^4 - x^3 + 3x^2 + x - 5$ at $x = 3$

$$
\begin{aligned}
p(x) &= 2x^4 - x^3 + 3x^2 + x - 5 \\
&= x(x(x(2x - 1) + 3) + 1) - 5 \\
&= 160
\end{aligned}
$$

Coef.	2	-1	3	1	-5
$x = 3$	2	$3 \cdot 2 + (-1) = 5$	$3 \cdot 5 + 3 = 18$	$3 \cdot 18 + 1 = 55$	$3 \cdot 55 + (-5) = 160$

Horner's Rule

Algorithm $\mathrm{Horner}(P[0..n], x)$

// Input: An array $P[0..n]$ of coefficients of a polynomial of degree n and a number x

//Output: The value of the polynomial at x

$p \leftarrow P[n]$

for $i \leftarrow n - 1$ **downto** 0 **do**

    $p \leftarrow x * p + P[i]$

return p

Horner's Rule

- The number of multiplications and the number of additions:

$$M(n) = A(n) = \sum_{i=0}^{n-1} 1 = n \in \Theta(n)$$

Brute Force Polynomial Evaluation

Algorithm $\mathrm{BruteForcePolynomial}(\mathrm{P}[0..\mathrm{n}], \mathrm{x})$

// Input, Output: same as Horner's rule's

$\mathrm{p} \leftarrow 0.0$

for $\mathrm{i} \leftarrow \mathrm{n}$ **downto** 0 **do**

power $\leftarrow 1$

for $\mathrm{j} \leftarrow 0$ **to** i **do**

power $\leftarrow \mathrm{power} * \mathrm{x}$

$\mathrm{p} \leftarrow \mathrm{p} + \mathrm{P}[\mathrm{i}] * \mathrm{power}$

return p

Exercise

1) Evaluate:

- $p(x) = 3x^4 - x^3 + 2x + 5$ at $x = -2$

2) Find the total number of multiplication & addition:

Algorithm $\mathrm{BruteForcePolynomial(P[0..n], x)}$

// Input, Output: same as Horner's rule's

$\mathrm{p} \leftarrow 0.0$

for $\mathrm{i} \leftarrow \mathrm{n}$ **downto** 0 **do**

power $\leftarrow 1$

for $\mathrm{j} \leftarrow 0$ **to** i **do**

power $\leftarrow \mathrm{power} * \mathrm{x}$

$\mathrm{p} \leftarrow \mathrm{p} + \mathrm{P[i]} * \mathrm{power}$

return p

Balanced Search Trees

Binary Search Tree (BST) is one of the principal data structures for implementing **dictionaries** (**sets or multisets with operations searching, addition and deletion**).

- Binary tree?
- Binary search tree?
- Time efficiency for searching, deletion and insertion?

Balanced Search Trees

Binary Search Tree (BST) is one of the principal data structures for implementing **dictionaries** (**sets or multisets with operations searching, addition and deletion**).

- Binary tree?
- Binary search tree?
- Time efficiency for searching, deletion and insertion?

$\Theta(\log n)$ in best/average case, $\Theta(n)$ in worse case

Balanced Search Trees

Problem:

Find a structure that preserves the **good properties** of the **binary search tree** – the **logarithmic efficiency** of dictionary operations – and having the set elements **sorted** but avoids its worst-case degeneracy.

Approaches:

- **Instance simplification**: an unbalanced BST is transformed into a balanced one (**self-balancing**): **AVL trees, red-black trees**
- **Representation change**: allow more than one element in a node of BST: **2-3 trees, 2-3-4 trees, B-trees**

AVL Trees

- **Adelson-Velsky, Landis, 1962**

- The **balance factor** of a node in a BST is **the difference of the heights of its left and right subtrees**.

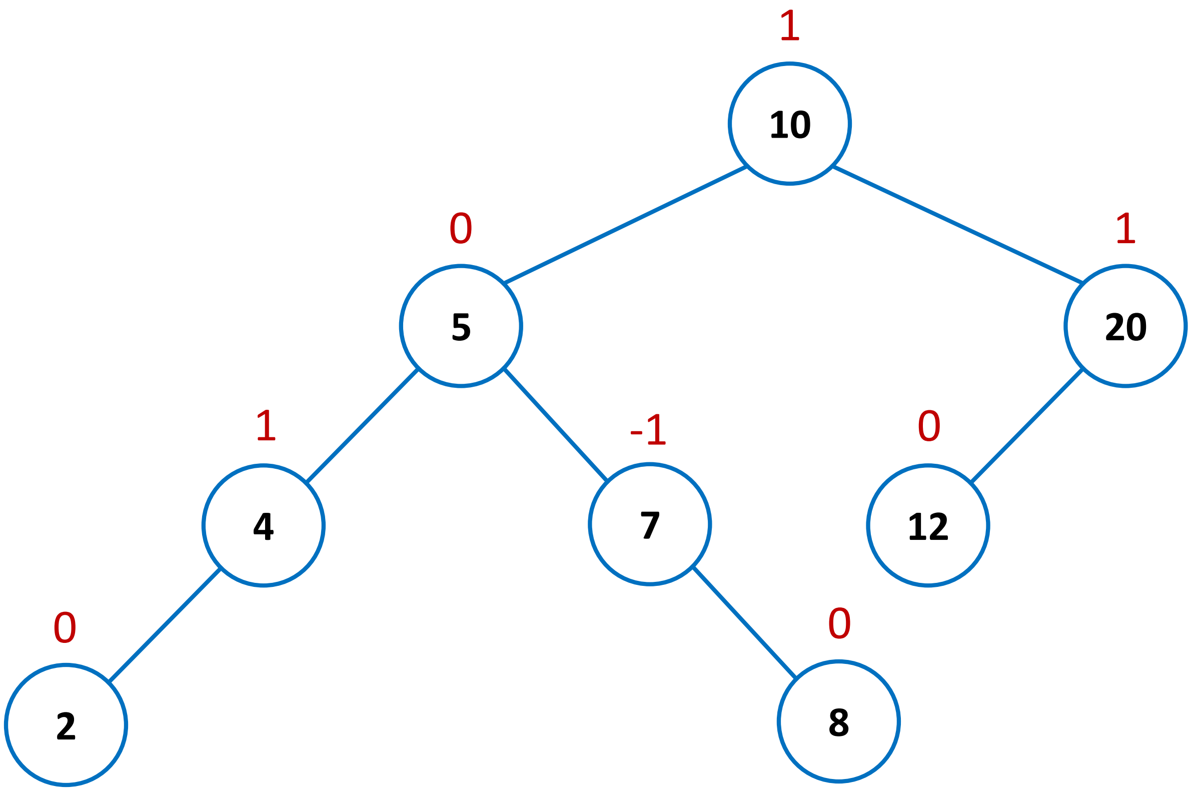
- An **AVL tree** is a BST in which the balance factor of every node is either 0 or $+1$ or -1.

- The height of the empty tree is -1.

AVL Trees

Example:

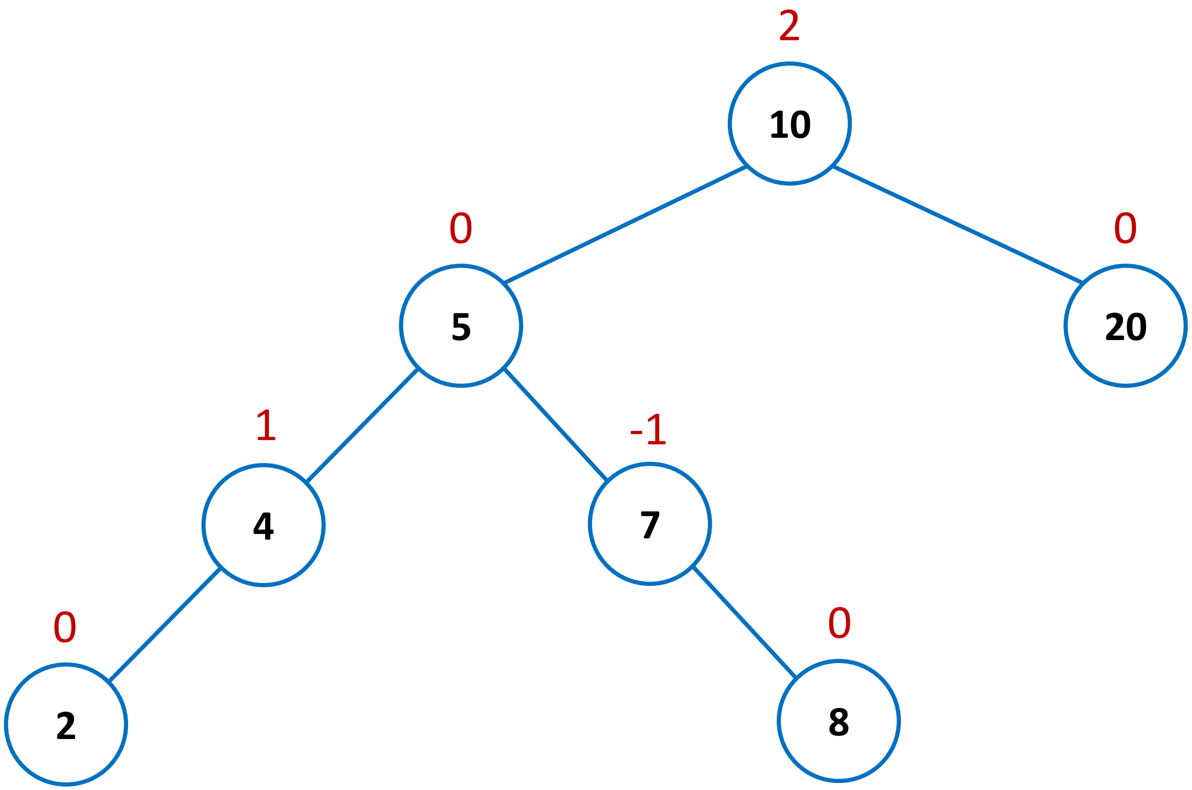
AVL



AVL Trees

Example:

not AVL



AVL Trees

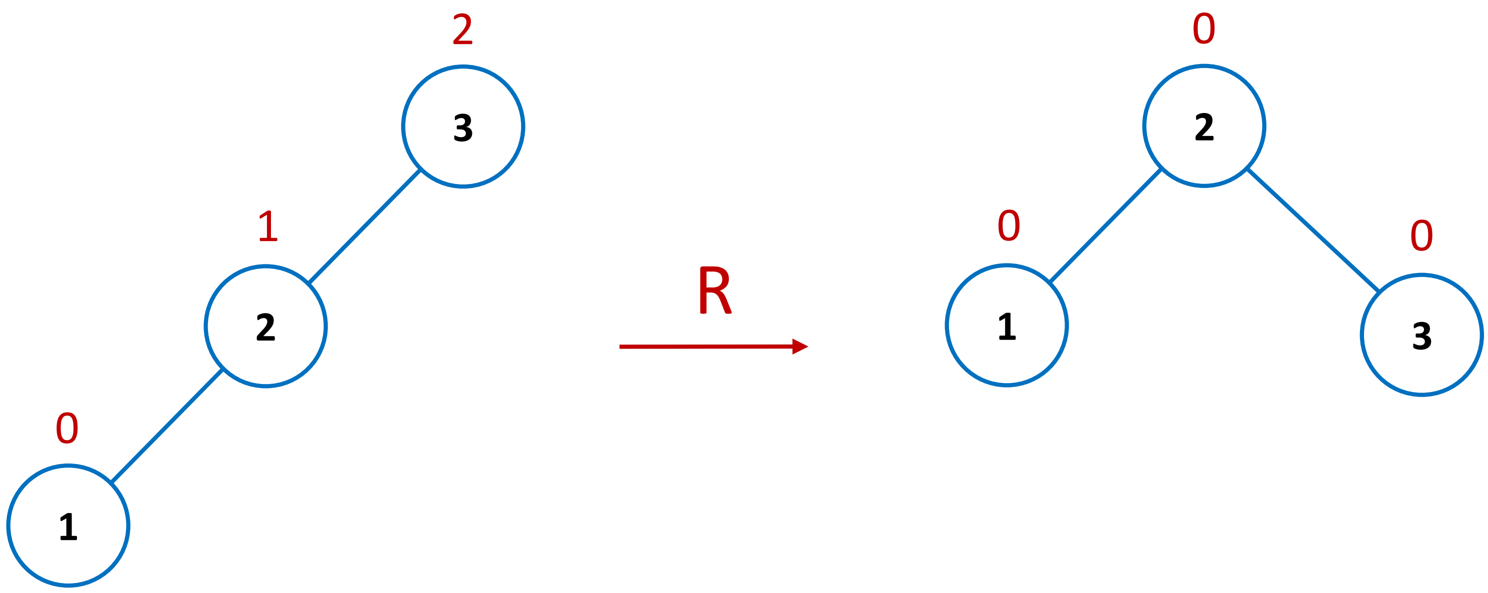
- If an **insertion** of a new node or an **deletion** of a node makes an AVL tree **unbalanced**, we transform the tree by a **rotation**.

- A **rotation** in an AVL tree is a **local transformation** of its subtree rooted at a node whose **balance factor** became either $+2$ or -2.

- If there are **several** such nodes, we rotate the subtree rooted at the unbalanced node that is the **closest to the newly inserted leaf**.

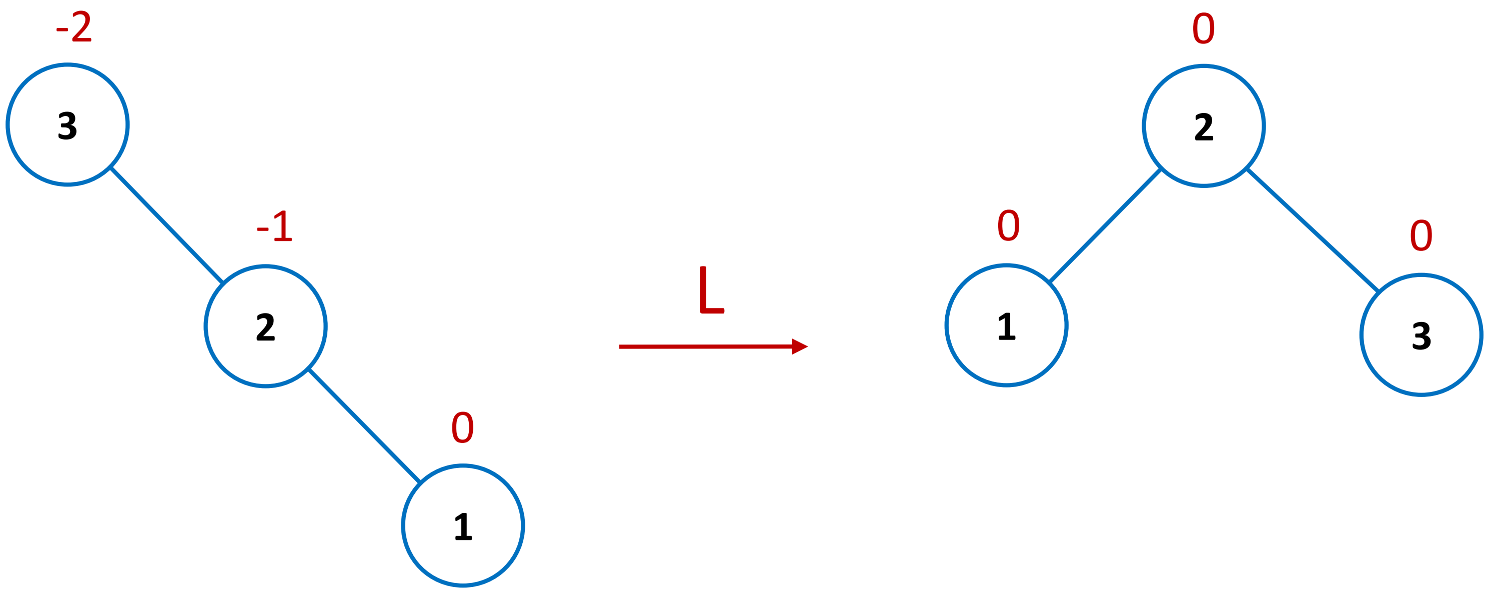
Types of Rotations

Single right rotation (R-rotation)



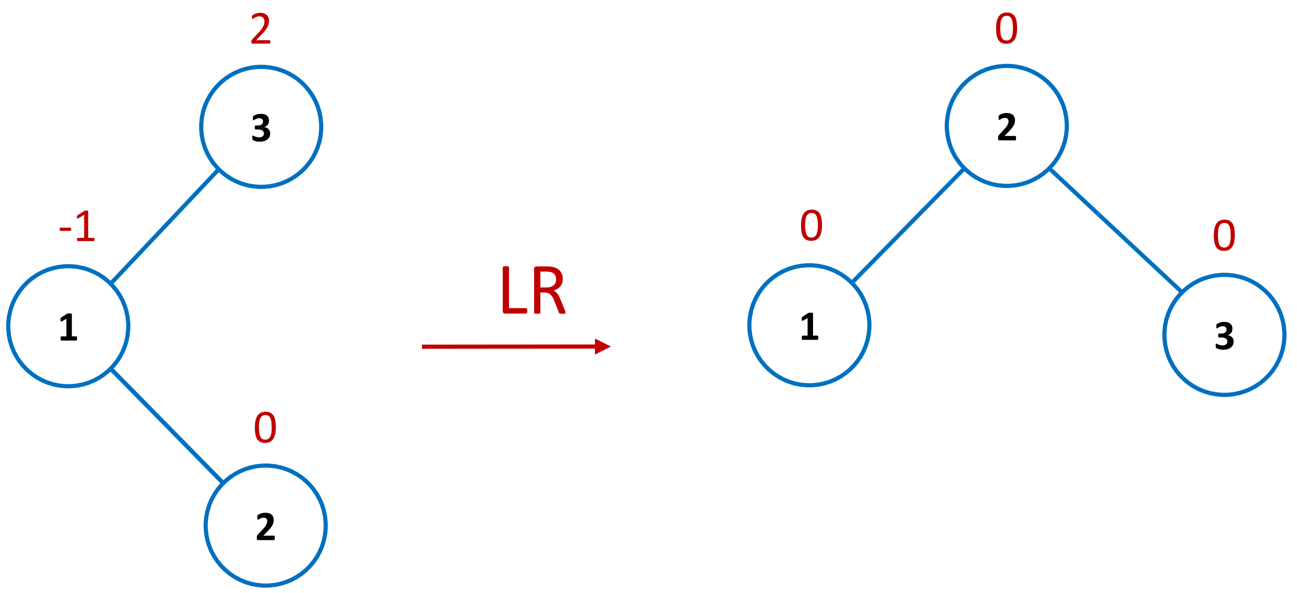
Types of Rotations

Single left rotation (L-rotation)



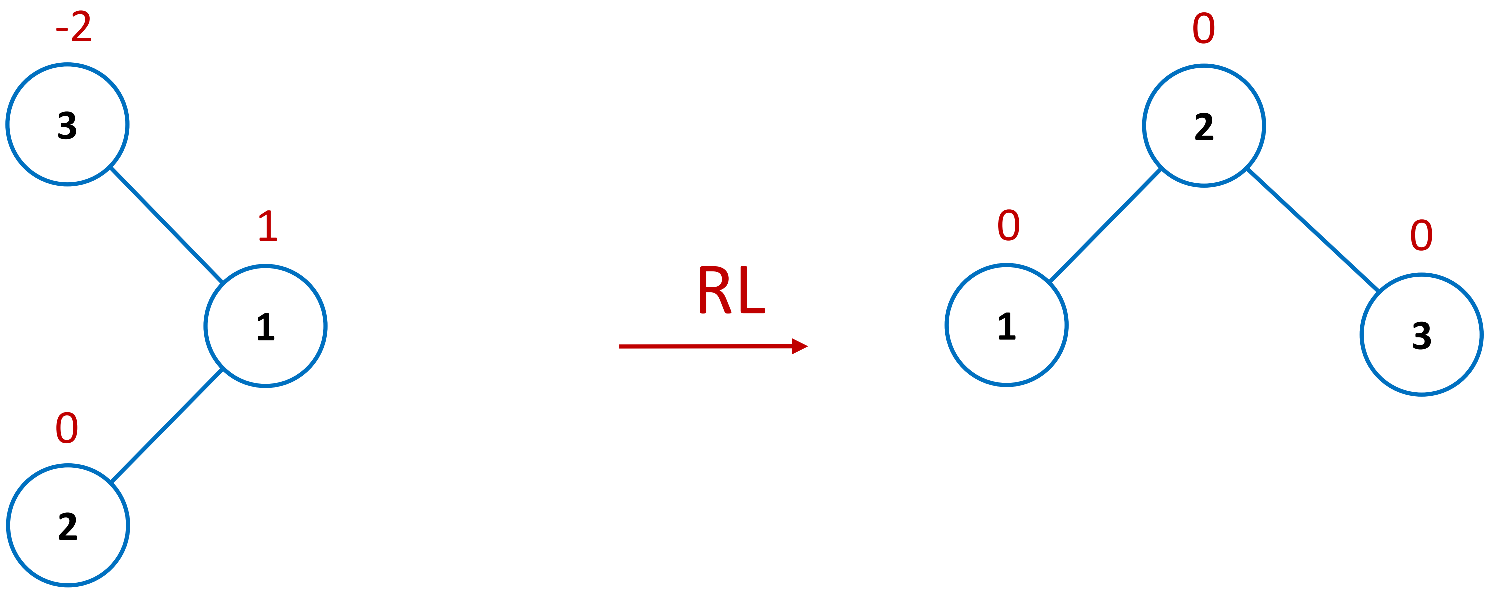
Types of Rotations

Double left-right rotation (LR-rotation)



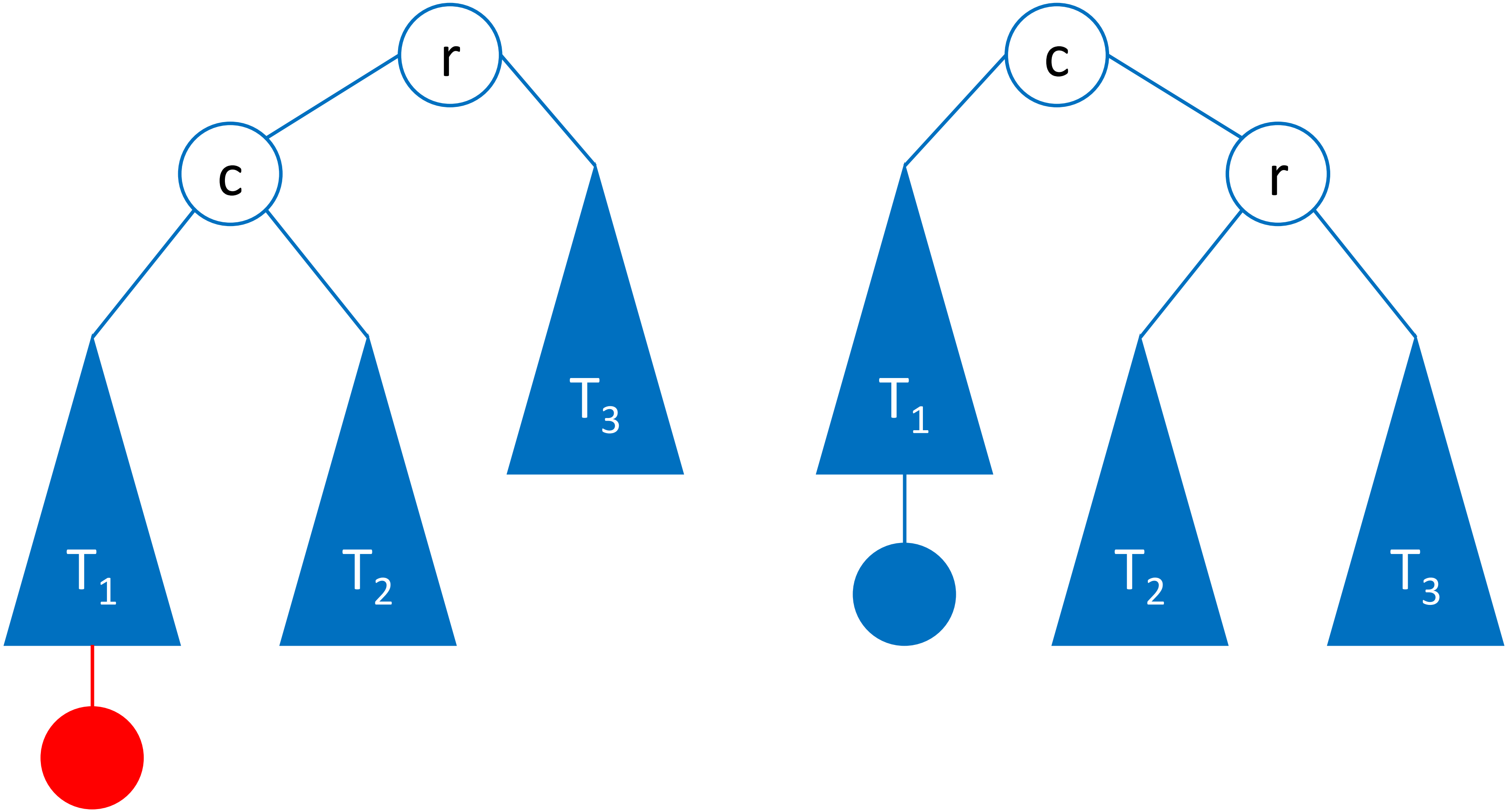
Types of Rotations

Double right-left rotation (RL-rotation)



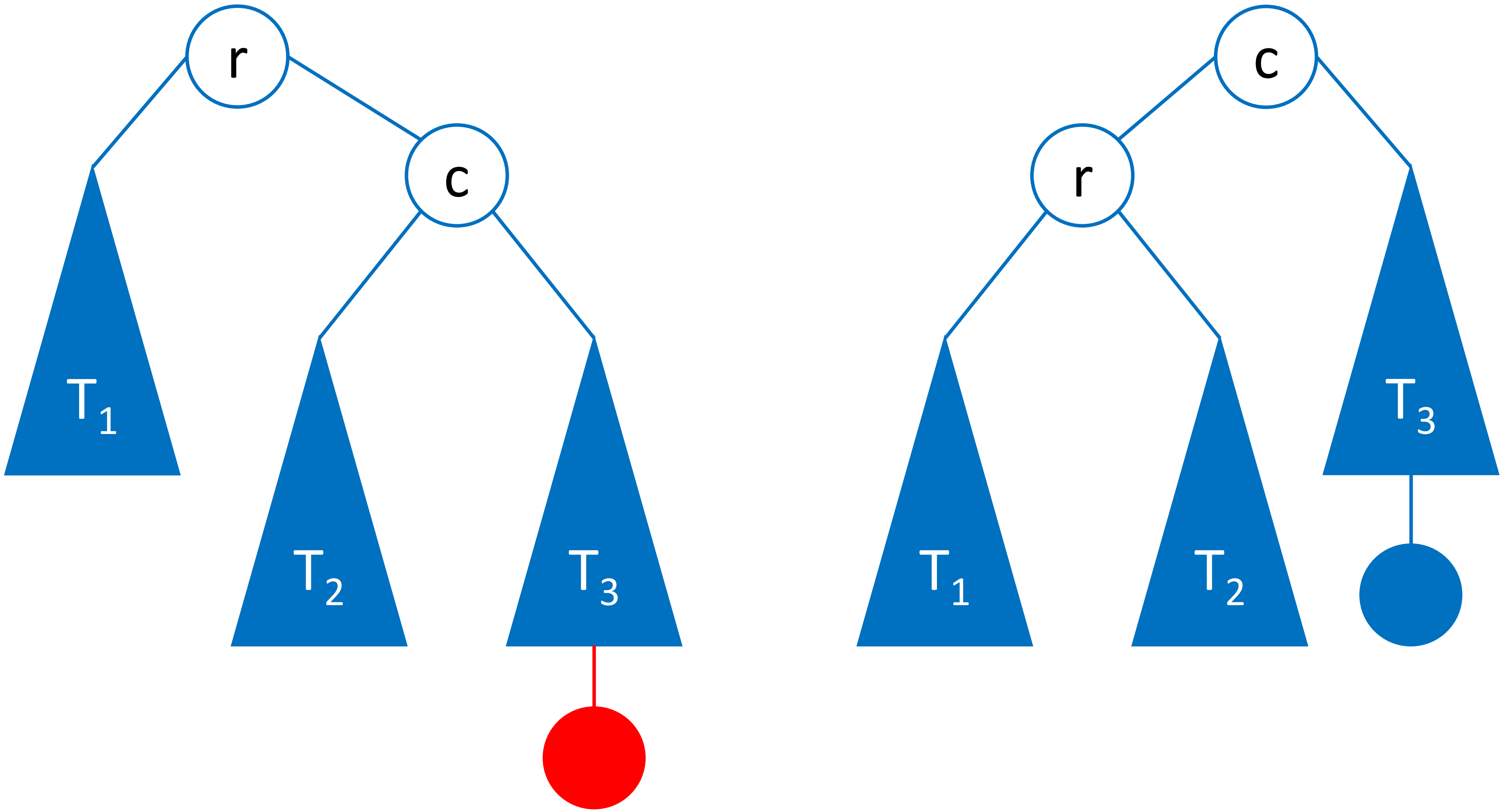
Types of Rotations

Single right rotation (R-rotation)



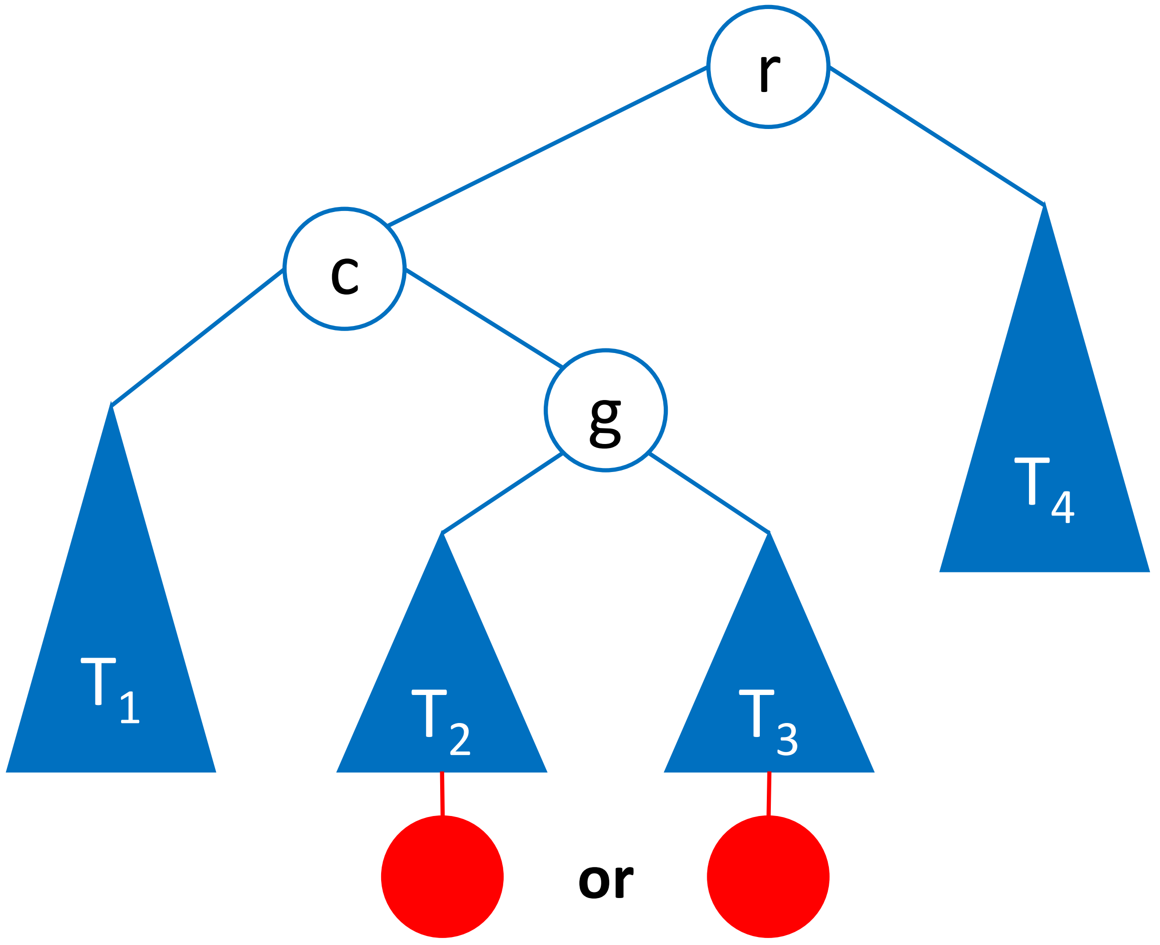
Types of Rotations

Single left rotation (L-rotation)



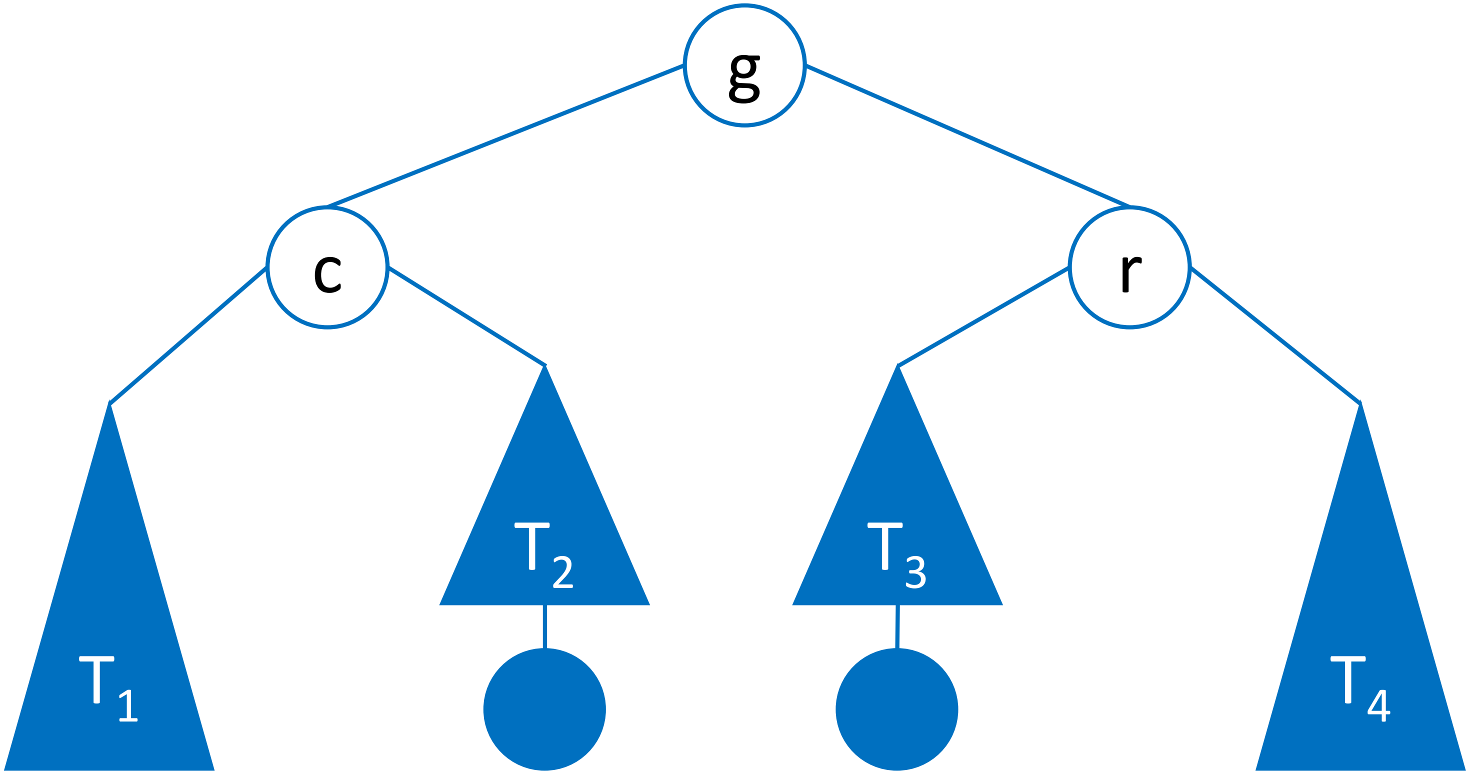
Types of Rotations

Double left-right rotation (LR-rotation)



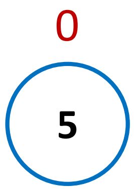
Types of Rotations

Double left-right rotation (LR-rotation)



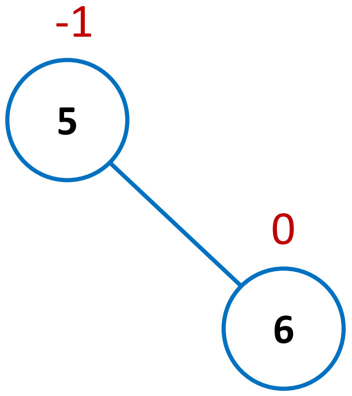
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



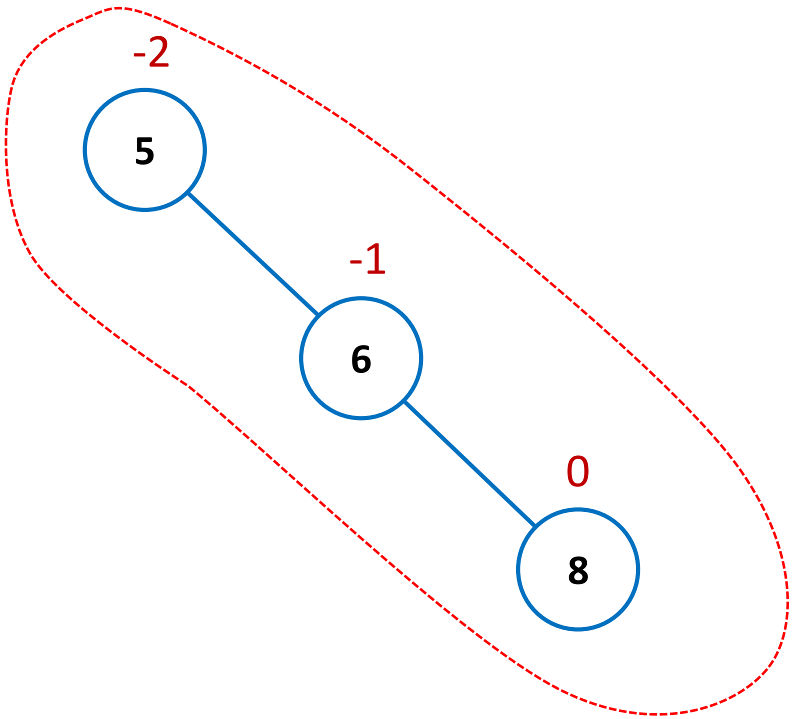
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



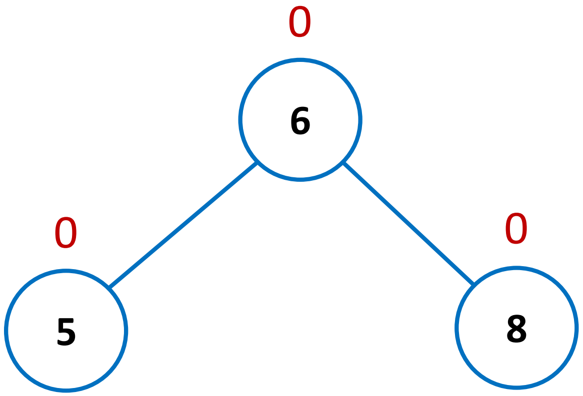
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



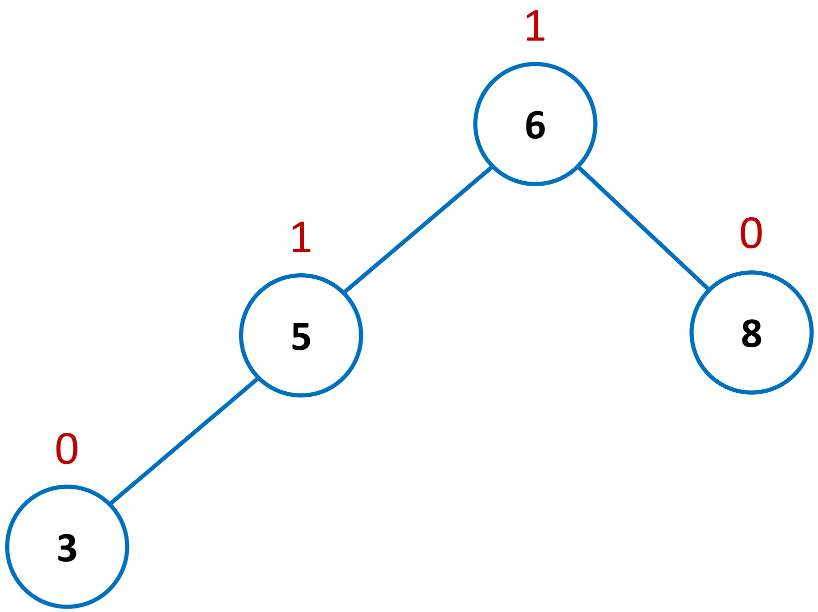
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



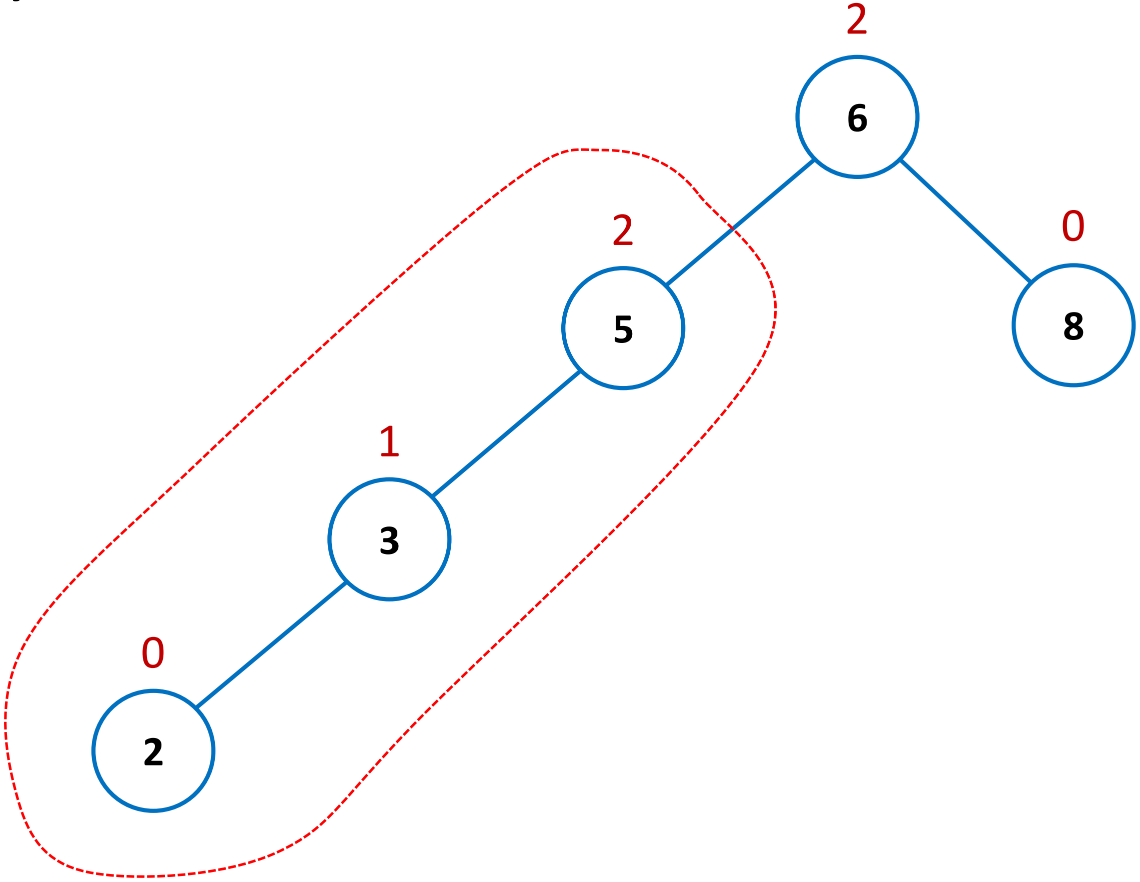
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



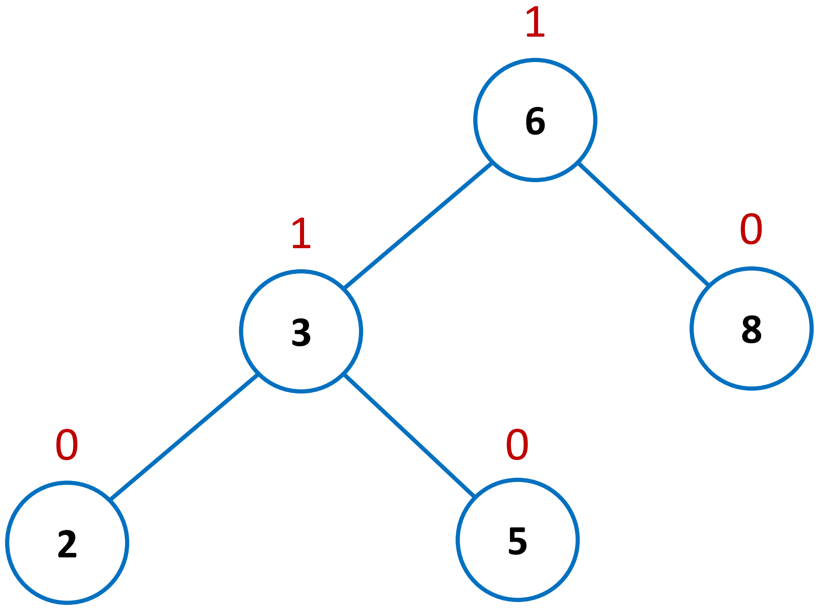
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



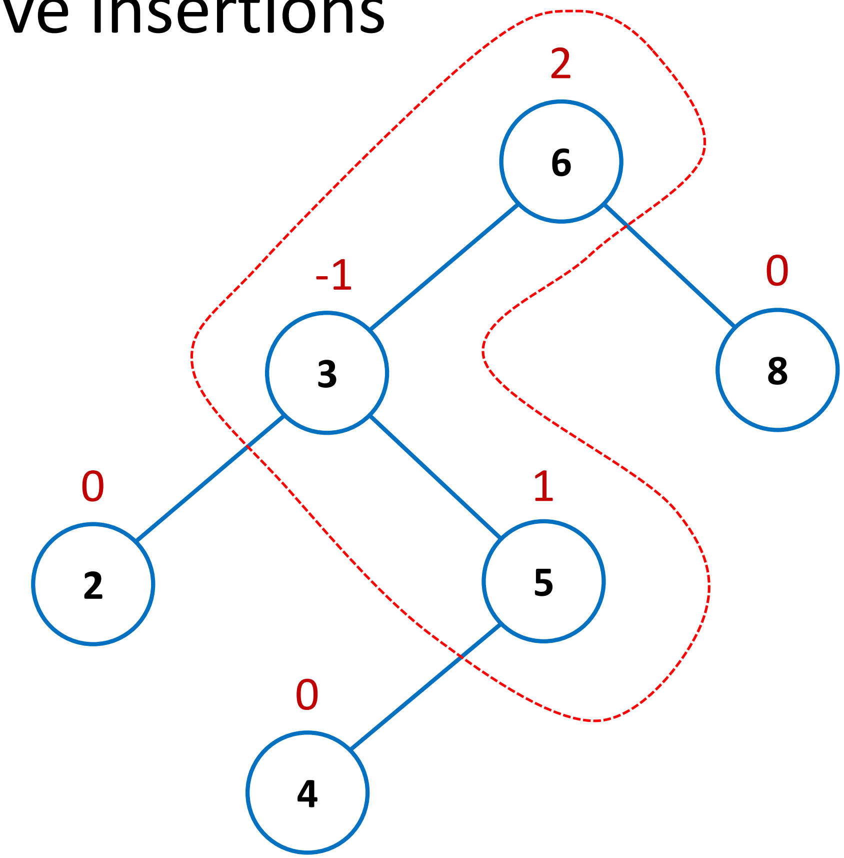
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



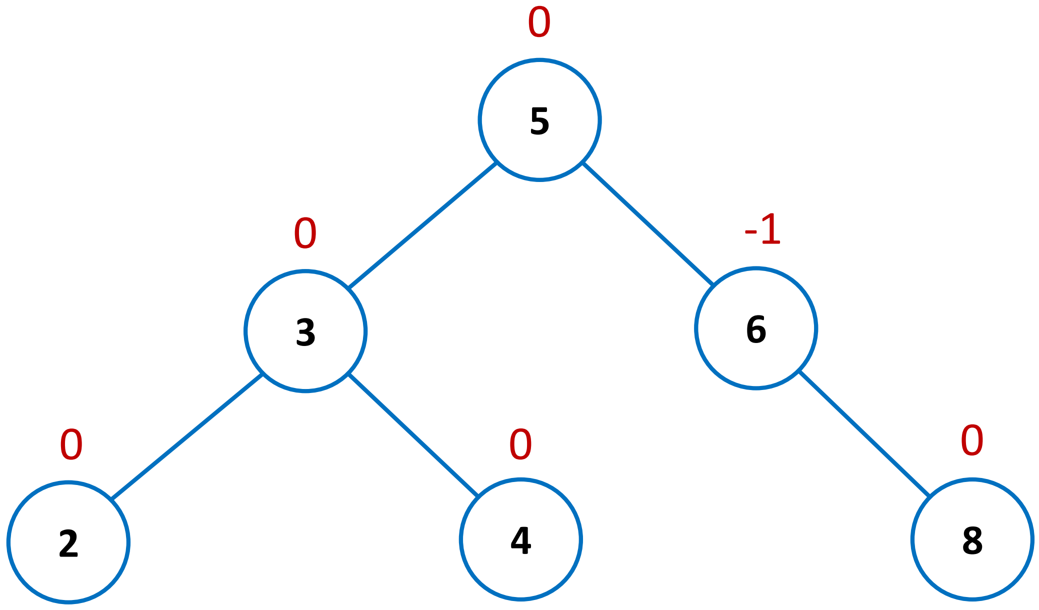
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



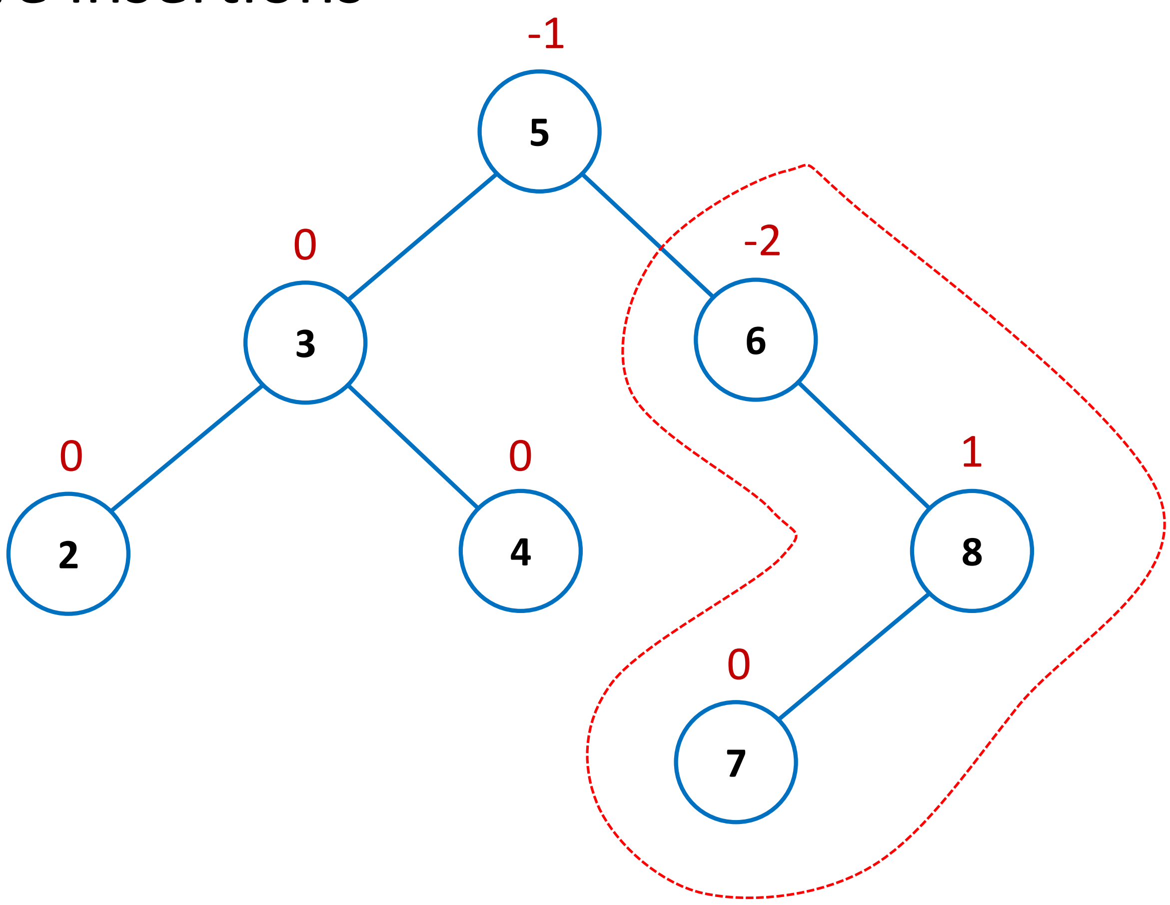
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



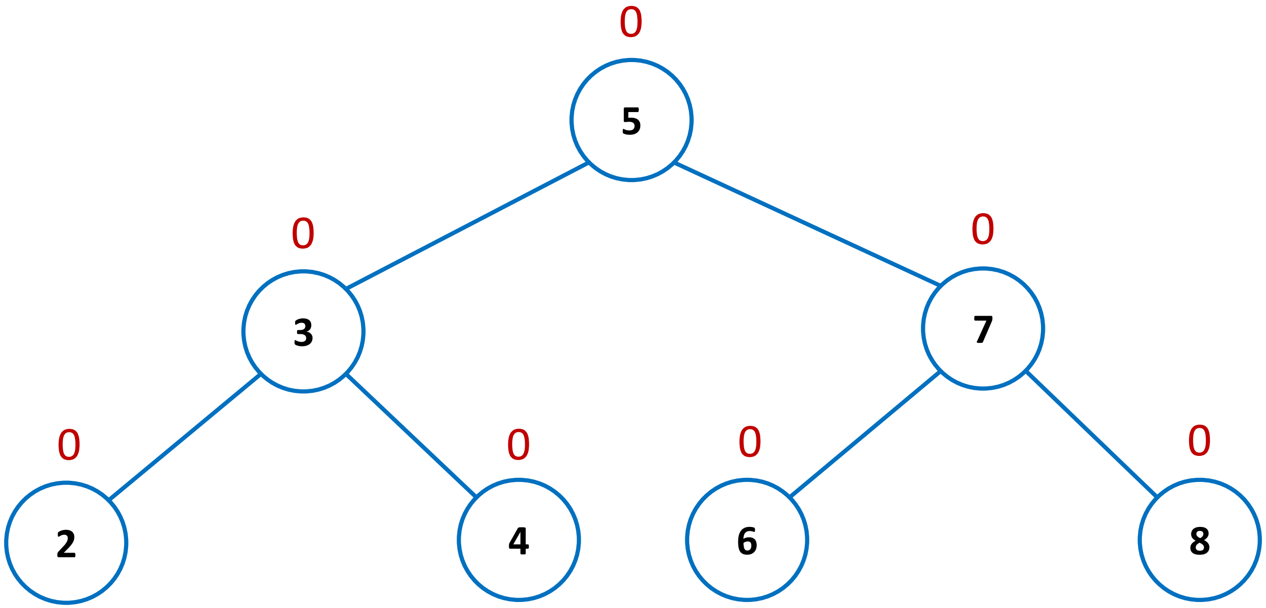
AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



AVL Trees

Example: Construct an AVL tree for the list $5, 6, 8, 3, 2, 4, 7$ by successive insertions



AVL Trees

The efficiency of AVL trees:

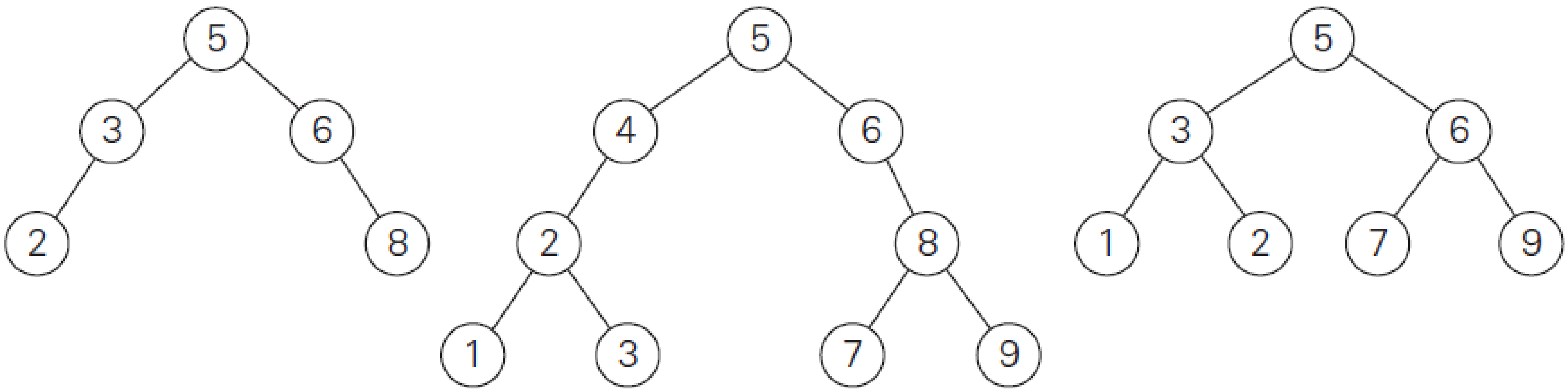
- The critical characteristic is the height of the tree
- $\lfloor \log_2 n \rfloor \leq h \leq 1.4405 \log_2(n+2) - 1.3277$
  - searching: $\Theta(\log n)$
  - insertion: $\Theta(\log n)$
  - deletion: $\Theta(\log n)$

http://interactivepython.org/runestone/static/pythonds/Trees/AVLTreePerformance.html

Drawbacks: frequent rotations and the need to maintain balances for its nodes

AVL Trees – Exercises

- Which of the following binary trees are AVL trees?



(a) (b) (c)

AVL Trees – Exercises

- For each of the following lists, **construct an AVL tree by inserting their elements successively**, starting with the empty tree.

  a) 1, 2, 3, 4, 5, 6

  b) 6, 5, 4, 3, 2, 1

  c) 3, 6, 5, 1, 2, 4