L04.02
AVL Trees

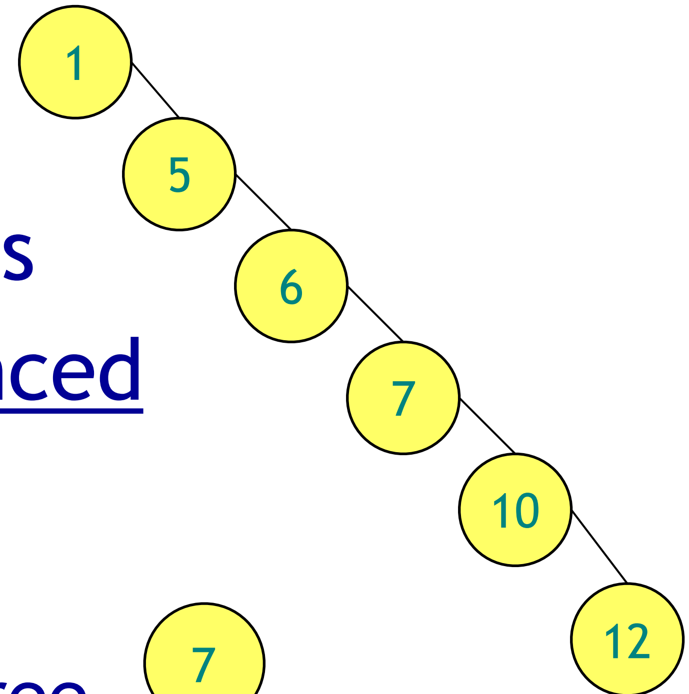
50.004 Introduction to Algorithm

Gemma Roig
(slides adapted from Dr. Simon LUI)
ISTD, SUTD

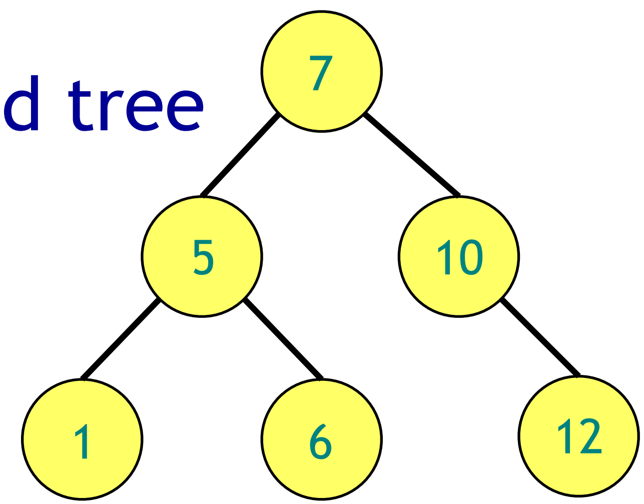
L04.02
AVL Trees

Chapter 13 CLRS book

Overview



- Review Binary Search Trees
- The Importance of a <u>balanced</u> tree
- AVL Tree
  - It is one kind of balanced tree
    - definition
    - rotations, insertion



BST review

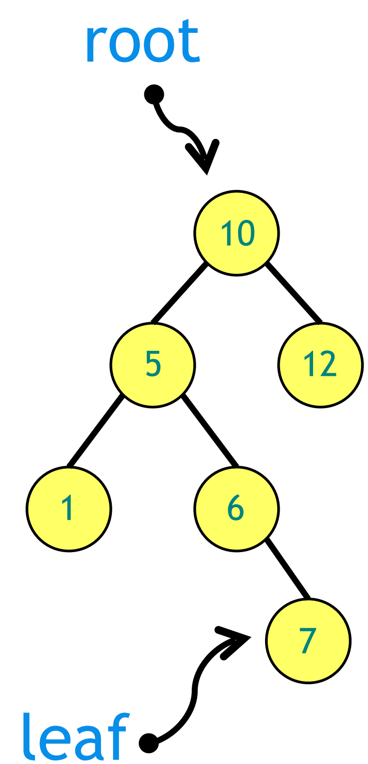
- Each node x has:
  - key[x]
  - Pointers: left[x], right[x], p[x]

Parent[x]

- Property: for any node x:
  - For all nodes y in the left subtree of x:

    key[y] ≤ key[x]
  - For all nodes y in the right subtree of x:

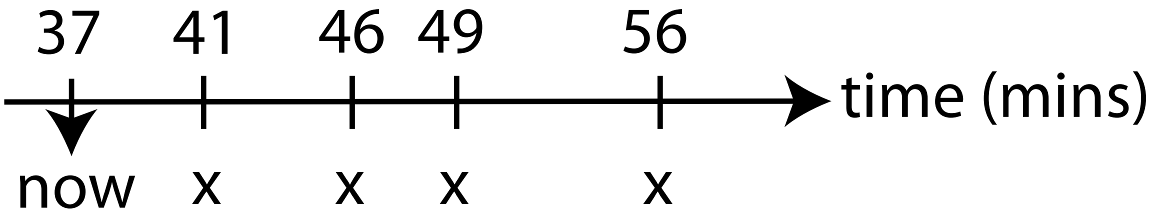
    key[y] ≥ key[x]

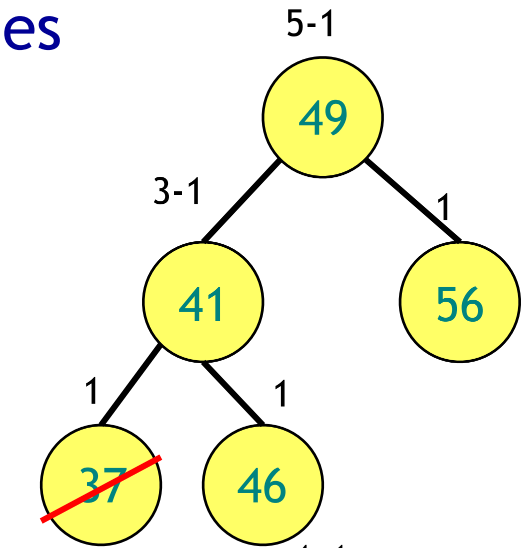


Tree height = 3

BST for runway reservation system

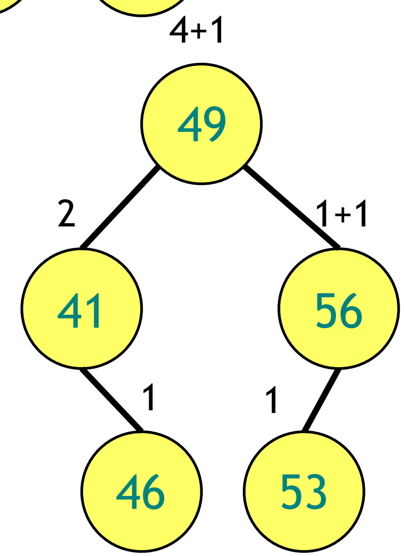
- R = (37, 41, 46, 49, 56) current landing times





- remove t from the set when a plane lands

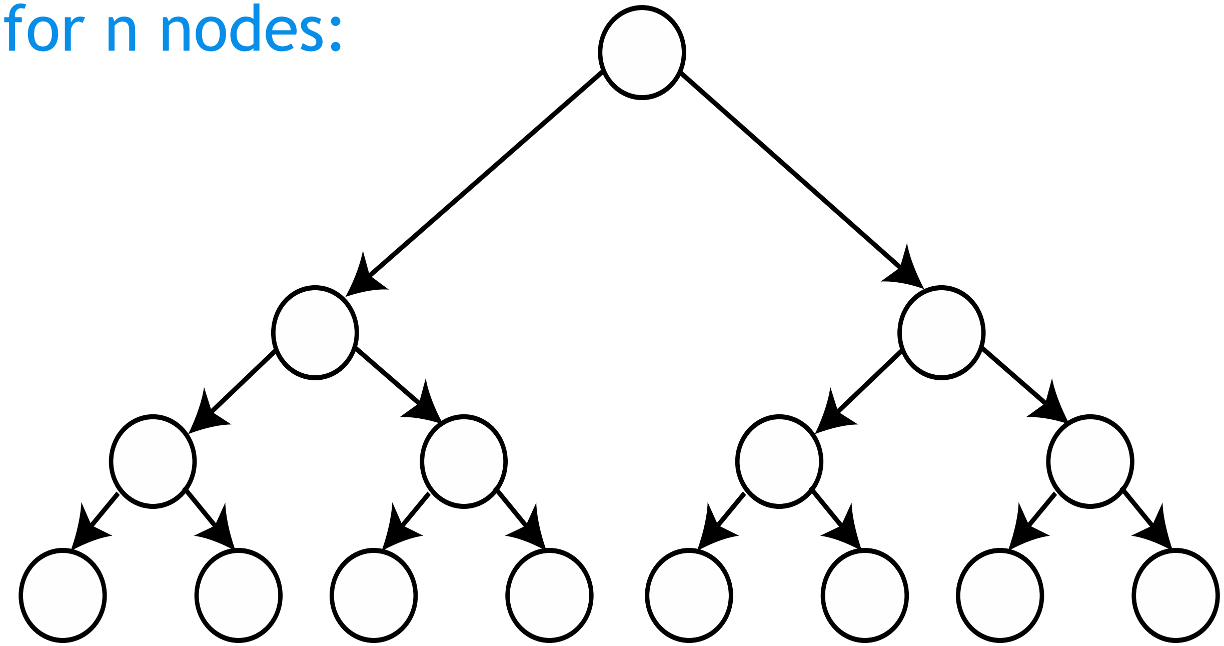
  R = (41, 46, 49, 56)



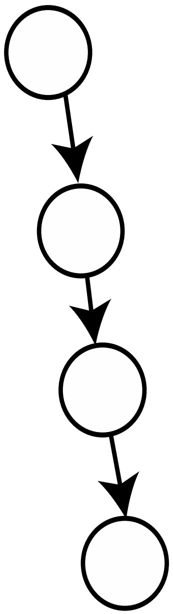
- add new t to the set if no other landings are scheduled within < 3 minutes from t
  - 40 => reject (41 in R)
  - 42 => reject (41 in R)
  - 53 => ok
- delete, insert take O(h), h=height of the tree

The importance of being balanced

for n nodes:



vs.



Perfectly Balanced

Path

$h = \Theta(\log n)$

$h = \Theta(n)$

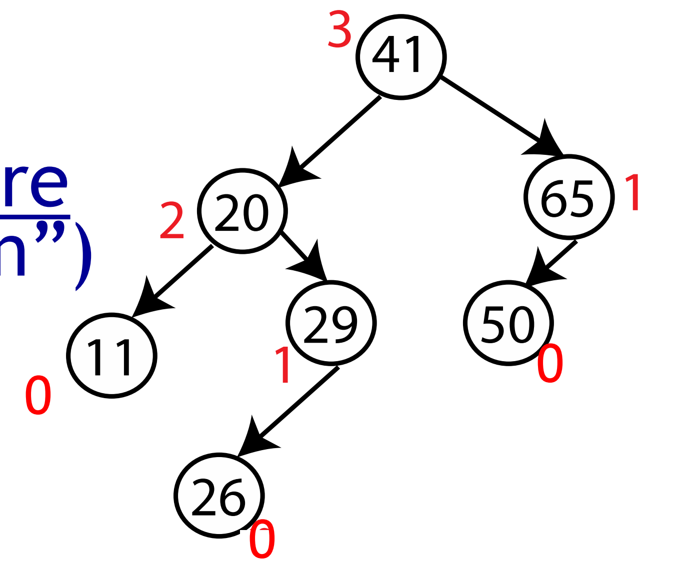
Balanced BST strategy

- Augment every node with some property
- Define a invariant on the property
- Show that the invariant guarantees $\Theta(\log n)$ height
- Design algorithms to maintain the property and the invariant

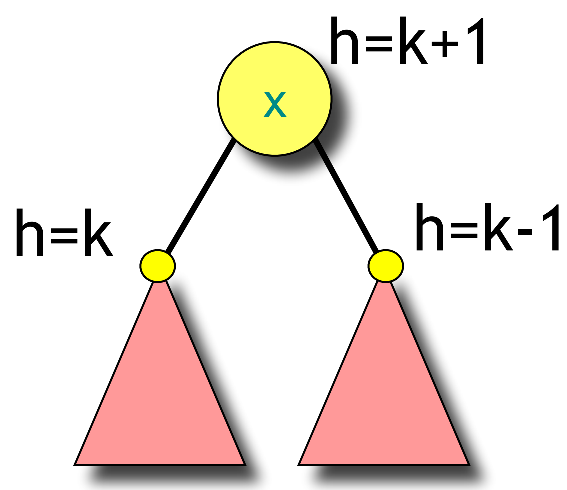
AVL Trees: Definition

[Adelson-Velskii and Landis'62]

- **Property**: for every node, <u>store</u> its own height ("augmentation")
  - Leaves have height 0
  - NIL has "height" -1



- **Invariant**: for every node x, the heights of its left child and right child differ by at most 1



Now the proof….
AVL Tree's height is Θ(log n)

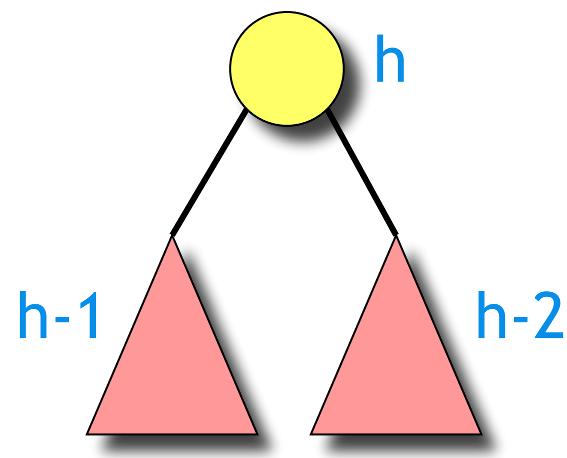
AVL trees have height $\Theta(\log n)$

- Let $\underline{n}_h$ be the minimum number of nodes of an AVL tree of height h
- We have

$$
\begin{aligned}
\underline{n}_h &\geq 1 + \underline{n}_{h-1} + \underline{n}_{h-2} \\
&> 2\underline{n}_{h-2} > 2 \cdot 2\underline{n}_{h-4} > \cdots \\
&> 2^{h/2}
\end{aligned}
$$

$$\Leftrightarrow h < 2\log \underline{n}_h \leq 2\log n$$

$$\Leftrightarrow h = O(\log n)$$



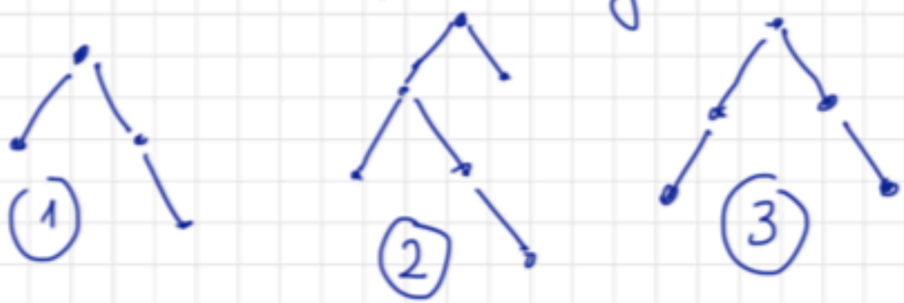
Why $h = \Omega(\log n)$?

- Let m_h be the max number of nodes of an AVL tree of height h
- $m_h <= 2^h$
- $\log(m_h) <= h$

Exercise

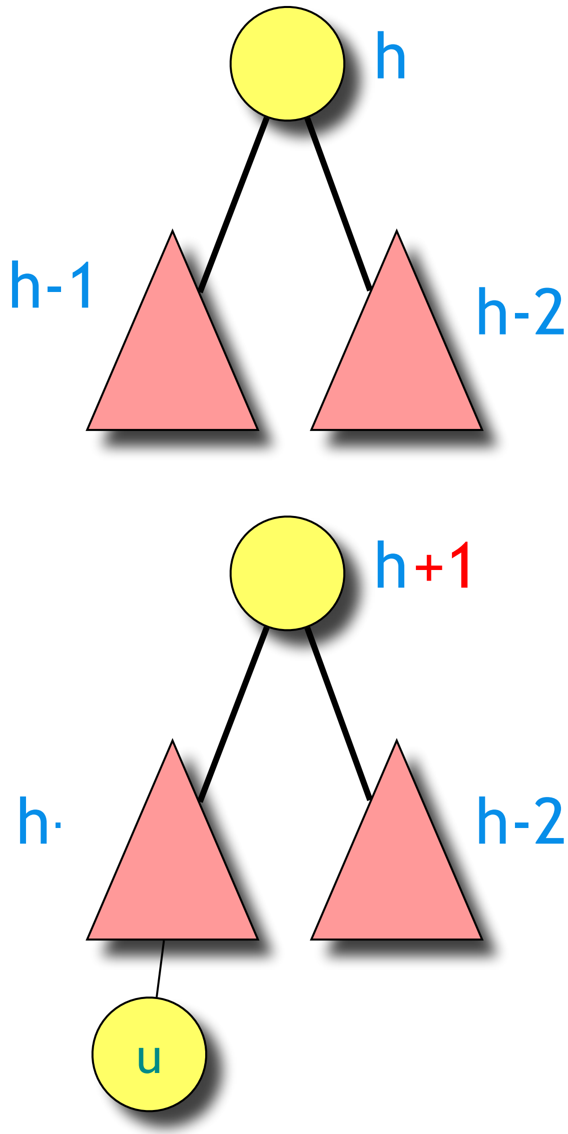
Question

A) Which of the following AVL trees are balanced?



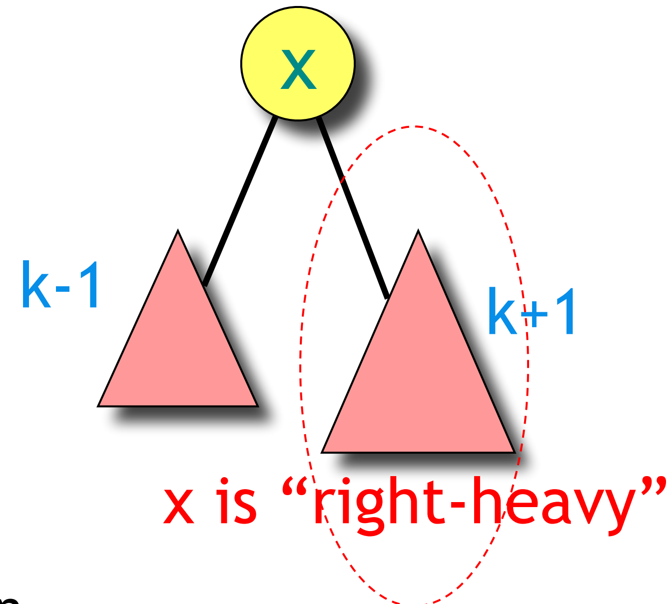
Insertions/Deletions

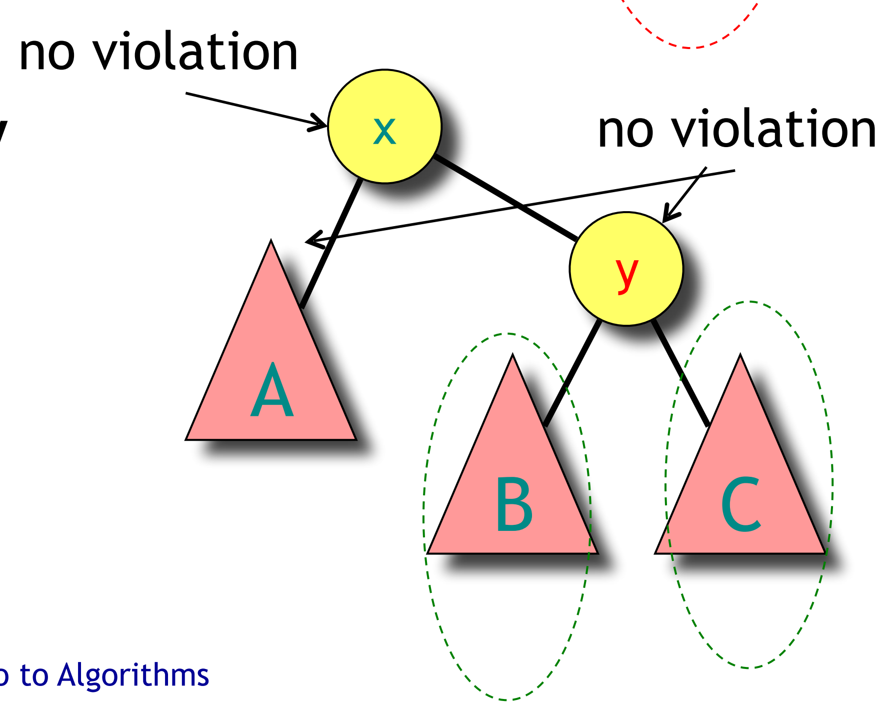
- Insert new node u as in the simple BST
  - Can create imbalance
- Similar issue/solution when deleting a node



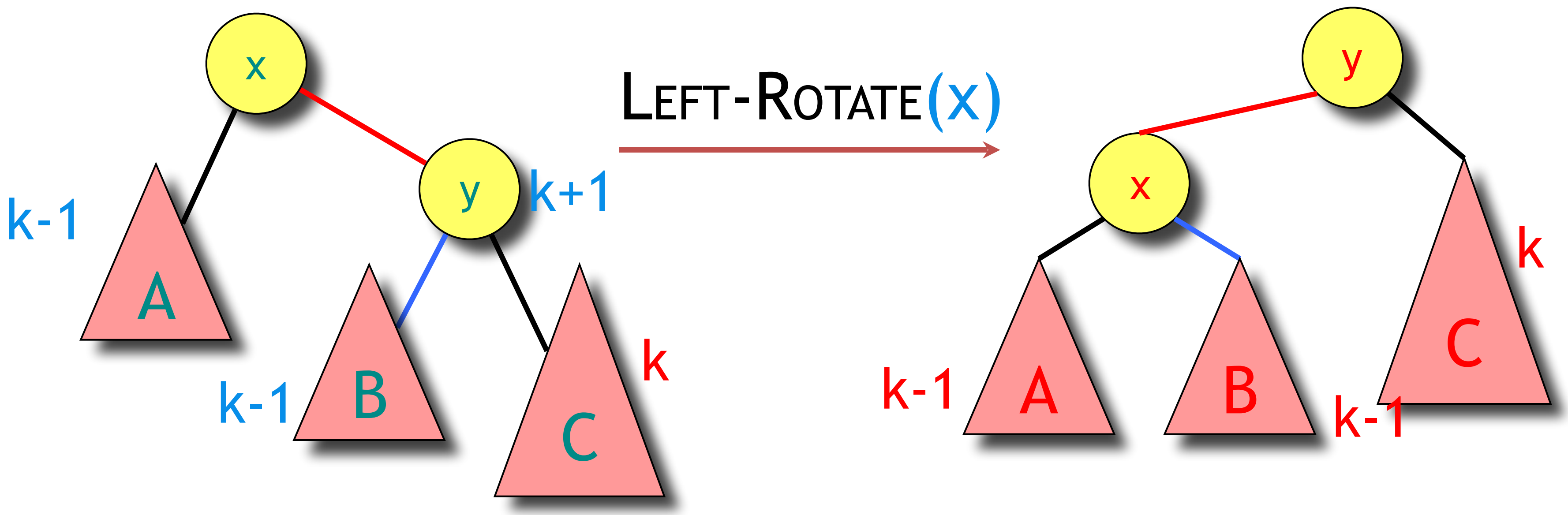
Balancing

- Let x be the lowest "violating" node (we will try to correct that)
- Assume that x is "right-heavy" (hence we analyze more the right subtree of x)
  - y = right child of x
- Scenarios:
  - Case 1: y is right-heavy
  - Case 2: y is balanced
  - Case 3: y is left-heavy

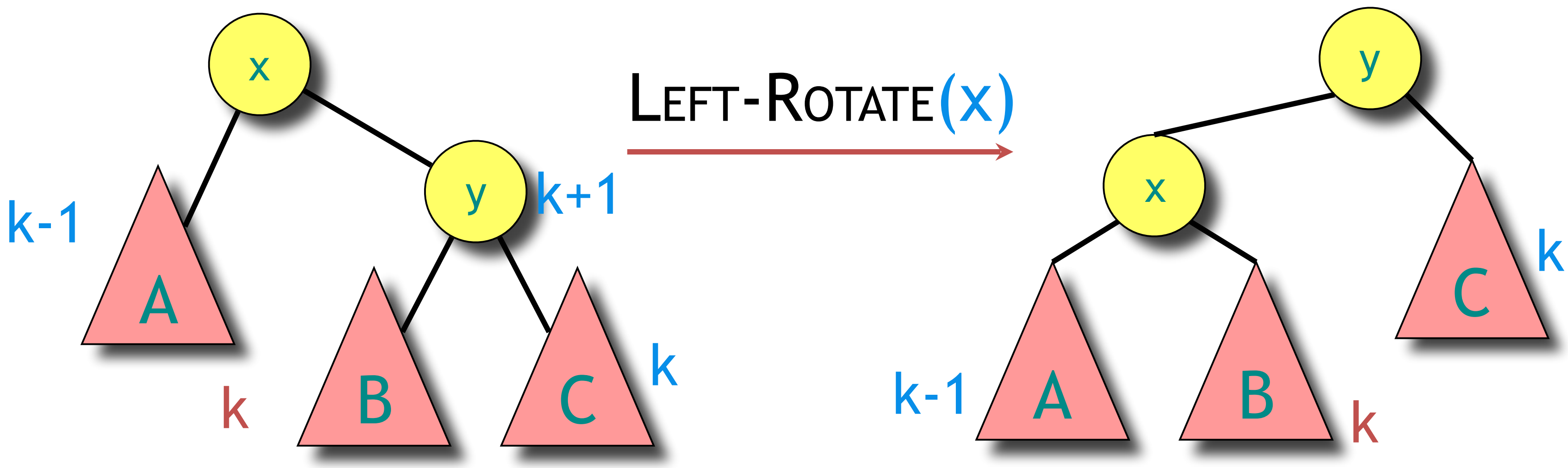




Case 1: y is right-heavy

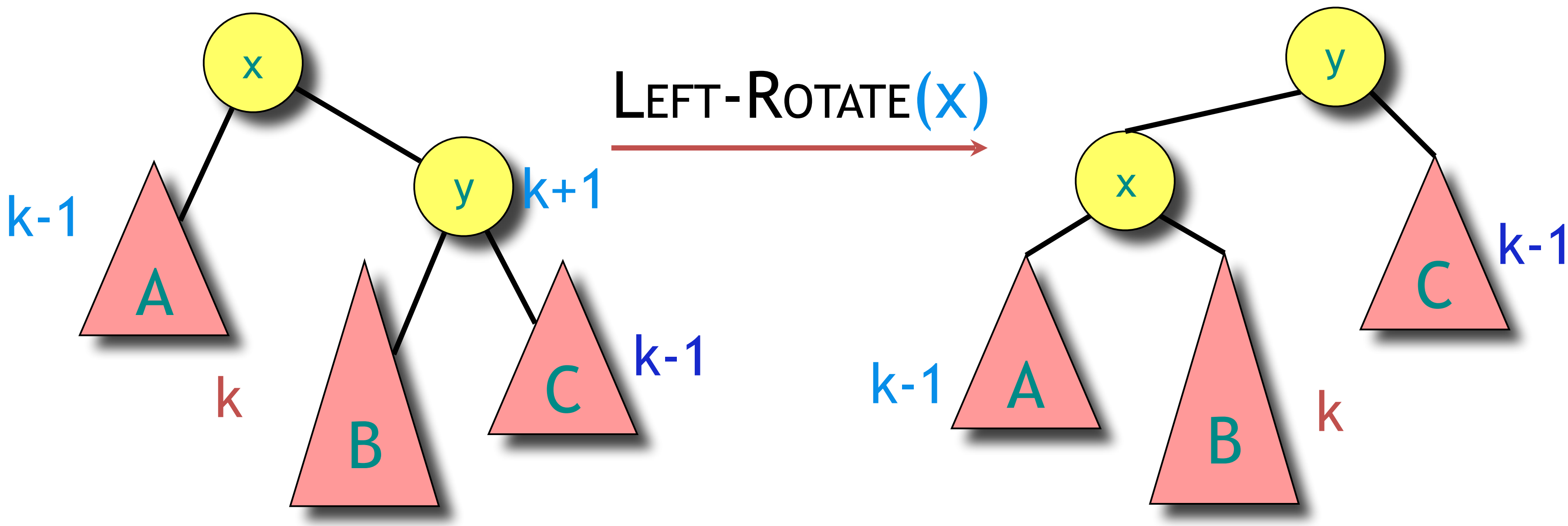


Case 2: y is balanced



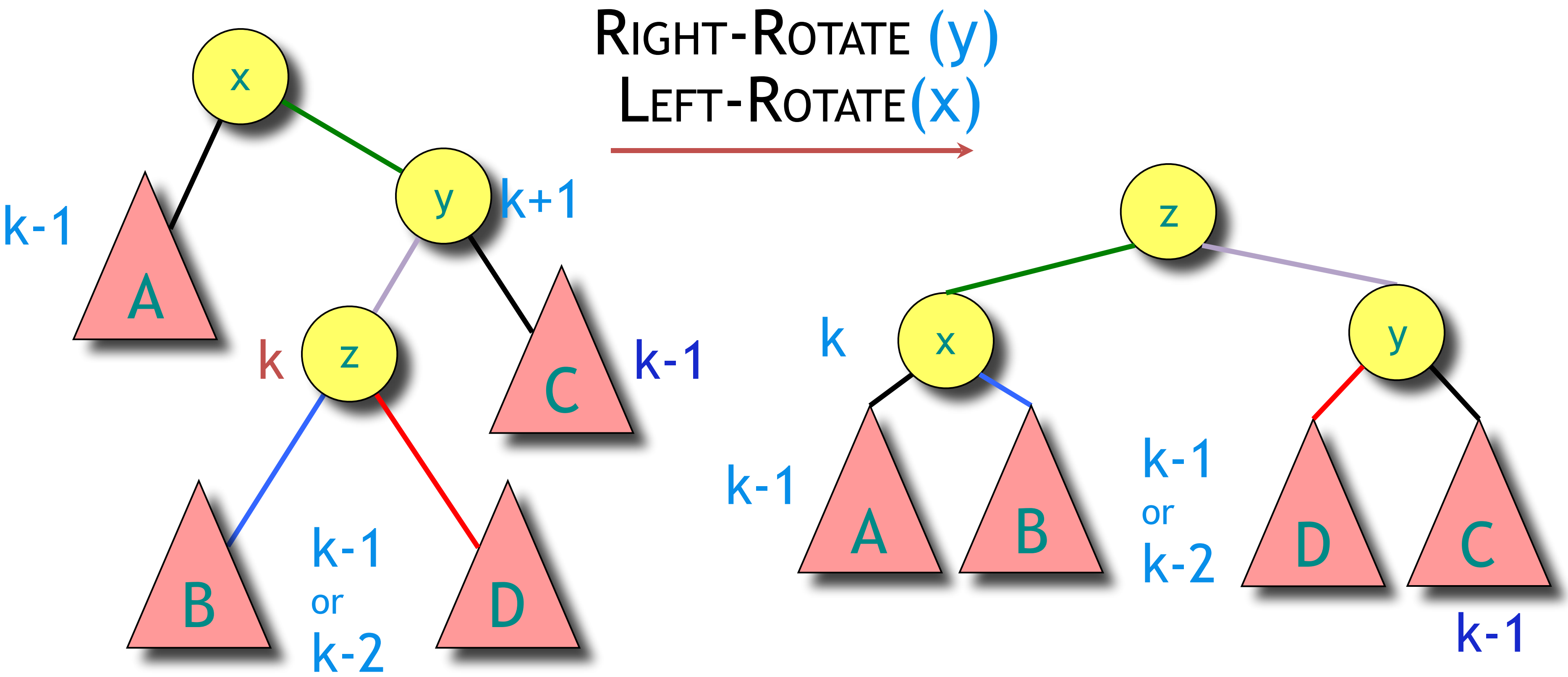
Same as Case 1

Case 3: y is left-heavy



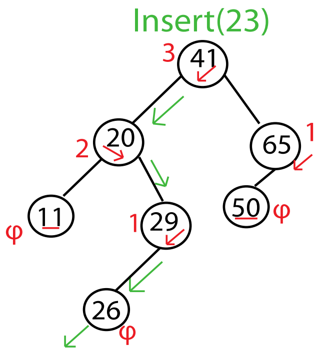
Need to do more ...

Case 3: y is left-heavy

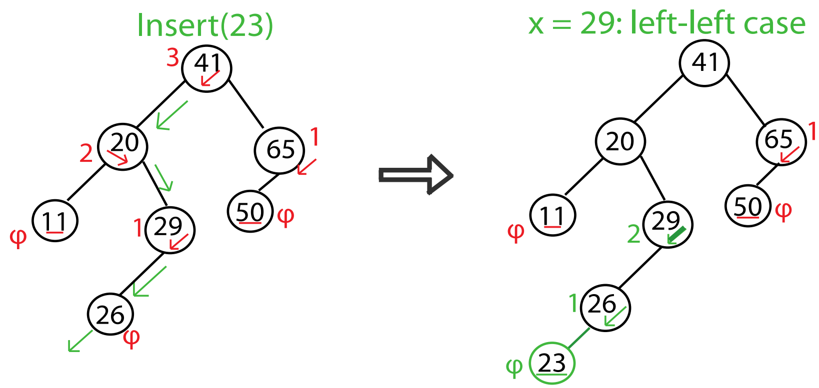


And we are done!

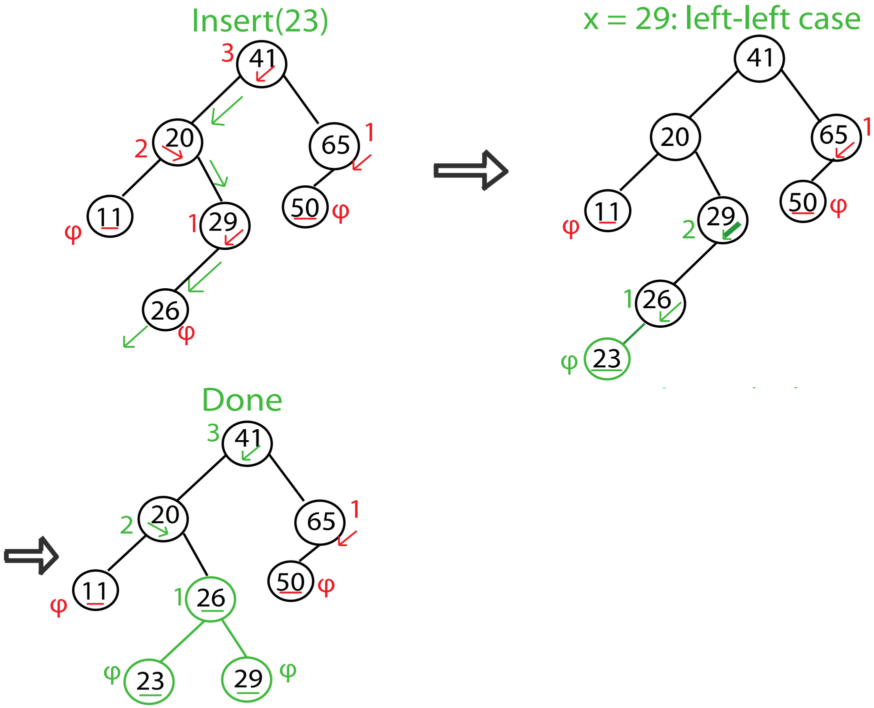
Examples of insert/balancing



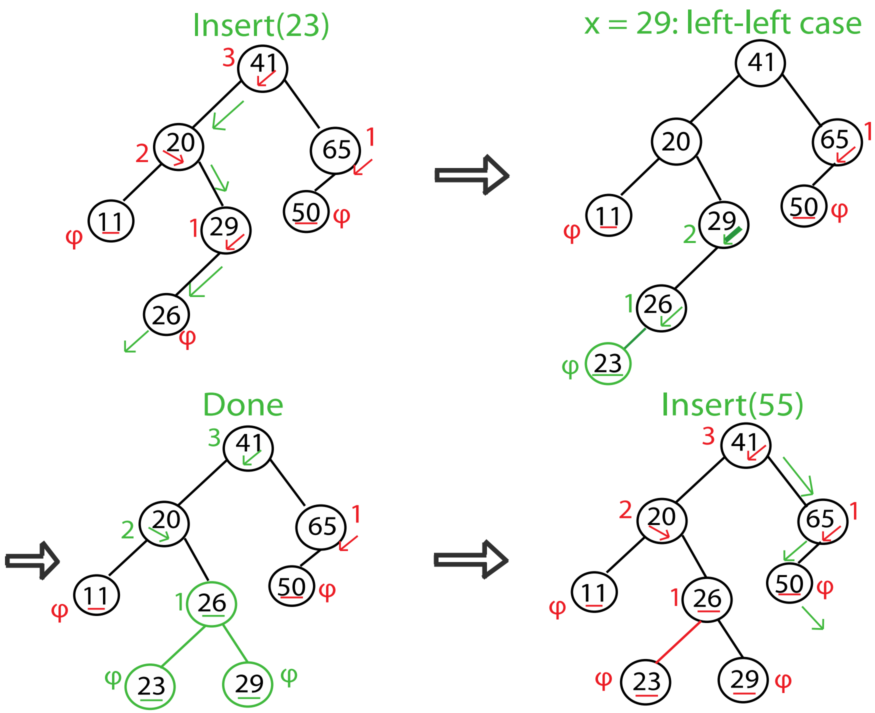
Examples of insert/balancing



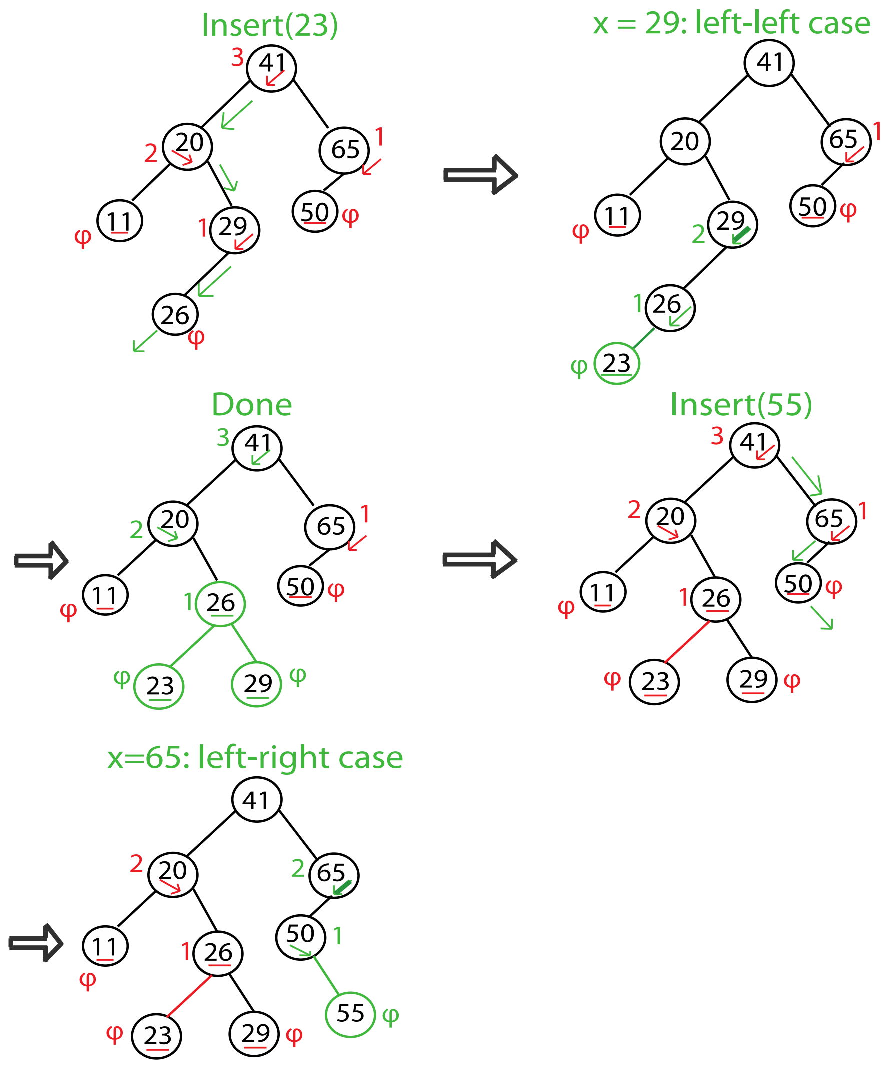
Examples of insert/balancing



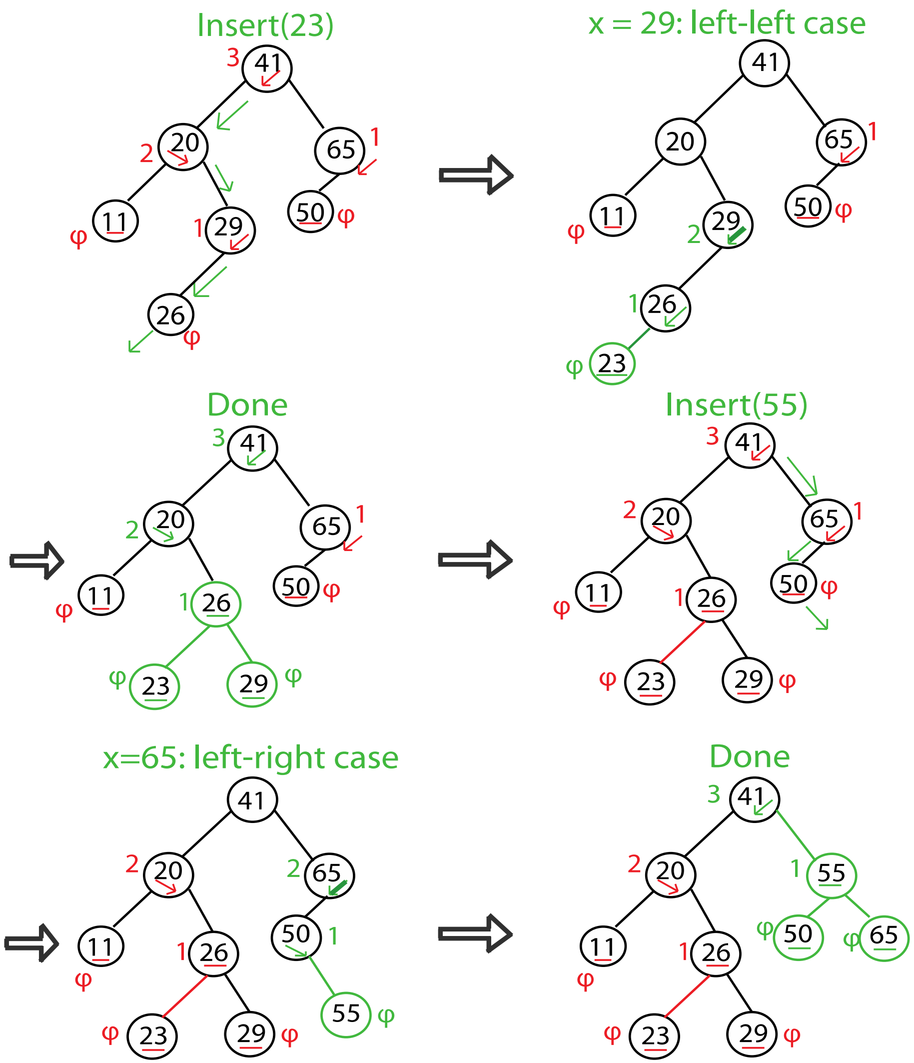
Examples of insert/balancing



Examples of insert/balancing



Examples of insert/balancing



Conclusions

- In balanced BSTs all operations take O(log$n$) time
- Can maintain balanced BSTs using O(log$n$) time per insertion

Exercise

- Let's look at AVL tree exercise.ppt