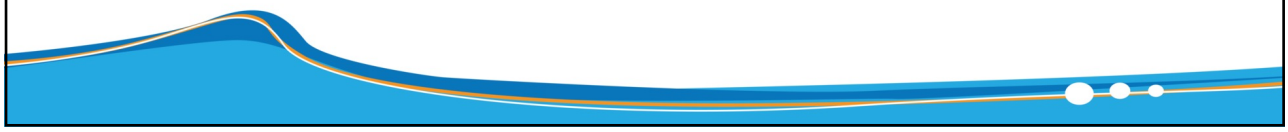
fit@hcmus

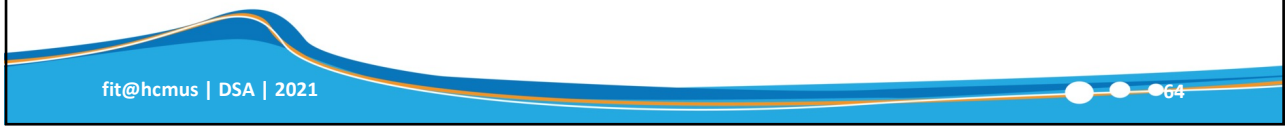
Tree Structures



1

fit@hcmus

AVL Tree



64

fit@hcmus

AVL Tree

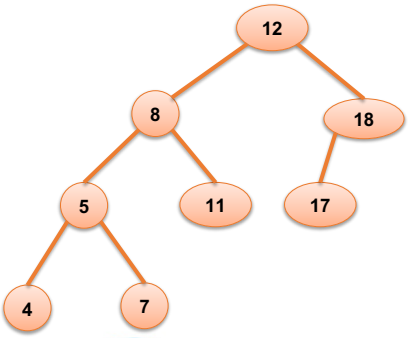
- Named for inventors, (Georgii) **A**delson-**V**elsky and (Evgenii) **L**andis
- Invented in 1962 (paper "*An algorithm for organization of information*").
- AVL Tree is a **self-balancing** binary search tree where
  - for ALL nodes, the difference between height of the left subtrees and the right subtrees **cannot be more than one**.

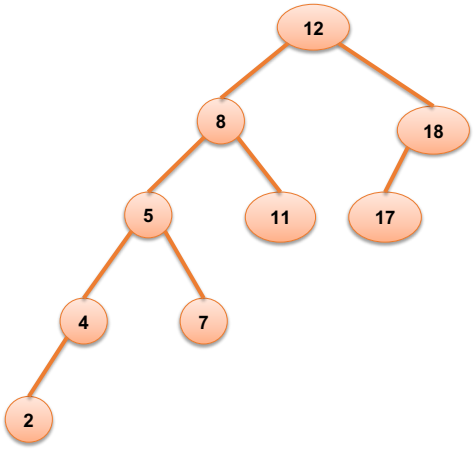
65

fit@hcmus

AVL Tree

Which is the AVL Tree?





66

fit@hcmus

AVL Tree

- A balanced binary search tree
  - Maintains height close to the minimum
  - After insertion or deletion, check the tree is still AVL tree – determine whether any node in tree has left and right subtrees whose heights differ by more than 1
- Can search AVL tree almost as efficiently as minimum-height binary search tree.

67

fit@hcmus

Cases of Un-balanced Nodes

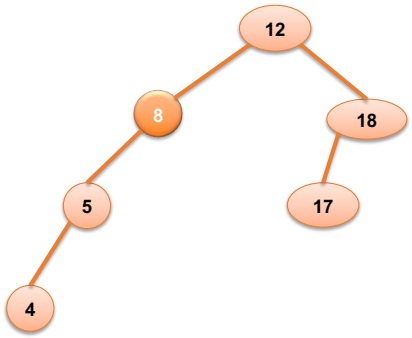
- Left-Left case
- Left-Right case
- Right-Right case
- Right-Left case

68

fit@hcmus

Cases of Un-balanced Nodes

- Left-Left case

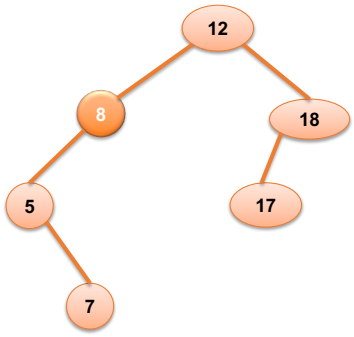


69

fit@hcmus

Cases of Un-balanced Nodes

- Left-Right case

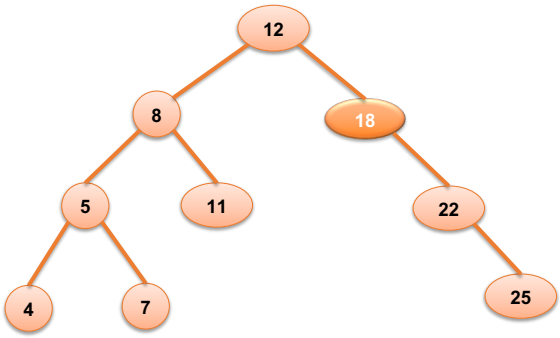


70

fit@hcmus

Cases of Un-balanced Nodes

- Right-Right case

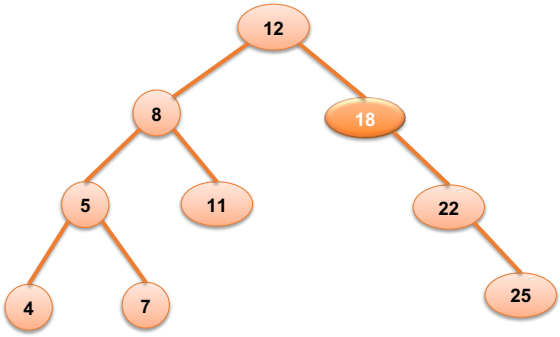


71

fit@hcmus

Cases of Un-balanced Nodes

- Right-Right case

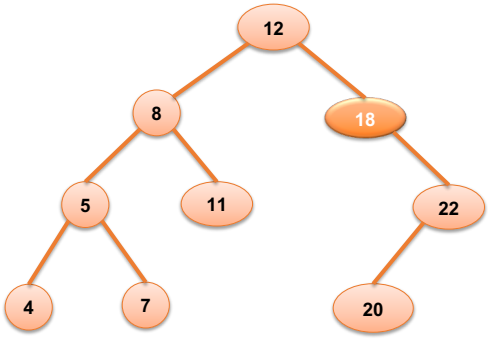


72

fit@hcmus

Cases of Un-balanced Nodes

- Right-Left case



73

fit@hcmus

Un-balanced Resolving

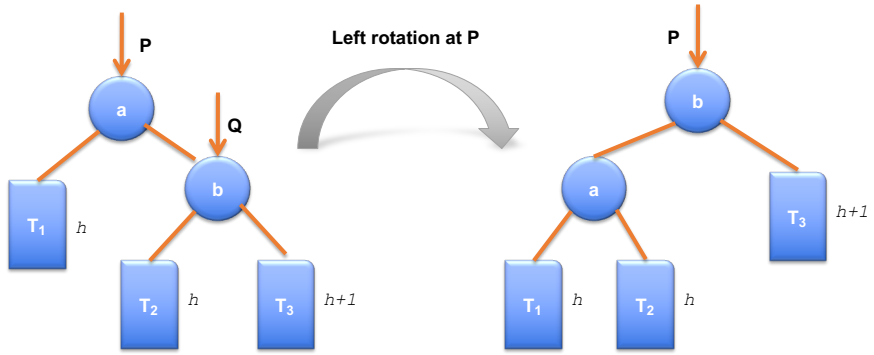
- Right-Right case:
  - Left rotation at un-balanced node.
- Right-Left case:
  - Right rotation at un-balanced node's right child.
  - Left rotation at un-balanced node.

74

fit@hcmus

Un-balanced Resolving

- Right-Right case:

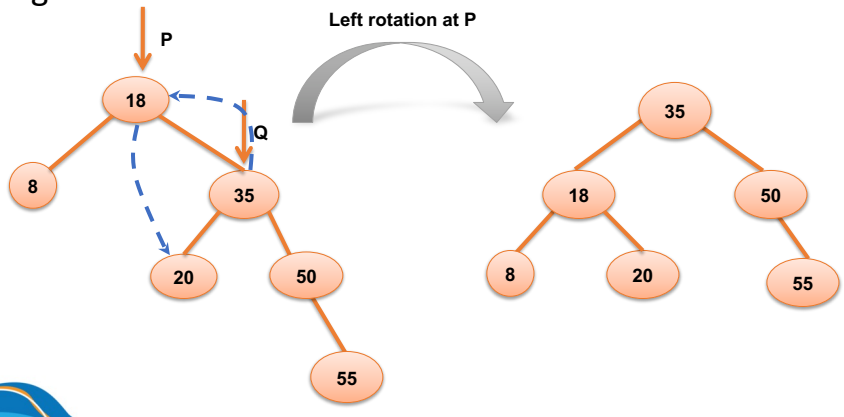


75

fit@hcmus

Un-balanced Resolving

- Right-Right case:

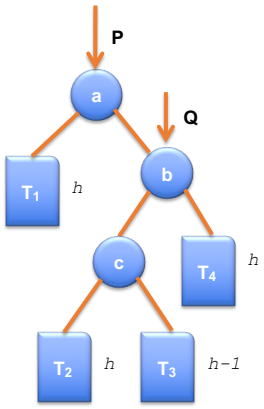


76

fit@hcmus

Un-balanced Resolving

- Right-Left case:
  - Right rotation at Q
  - Left rotation at P

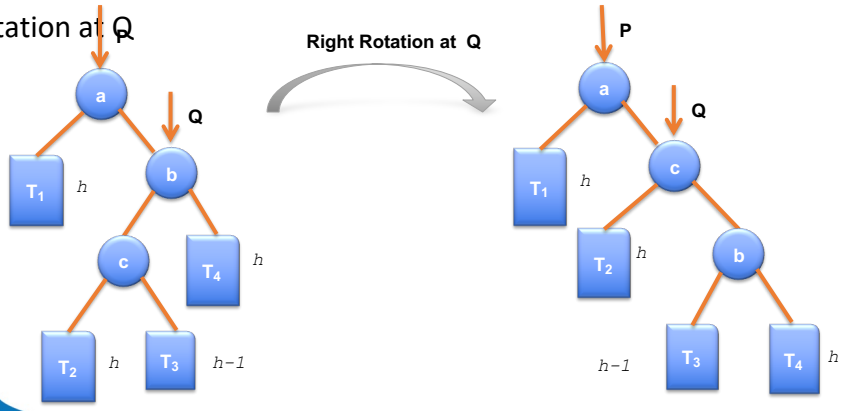


77

fit@hcmus

Un-balanced Resolving

- Right-Left case:
  - Right rotation at Q

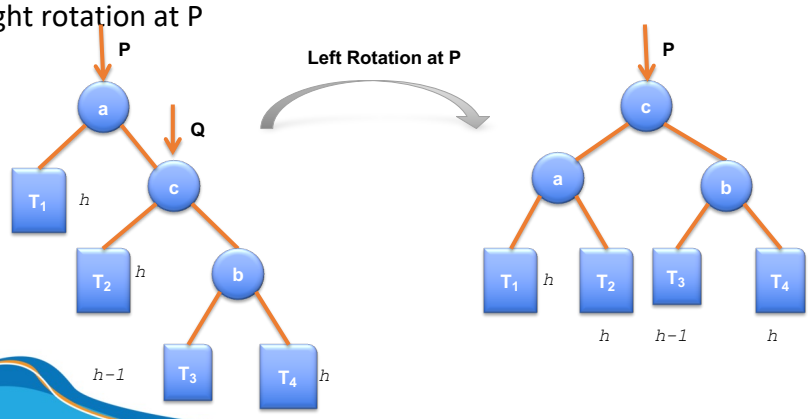


78

fit@hcmus

Un-balanced Resolving

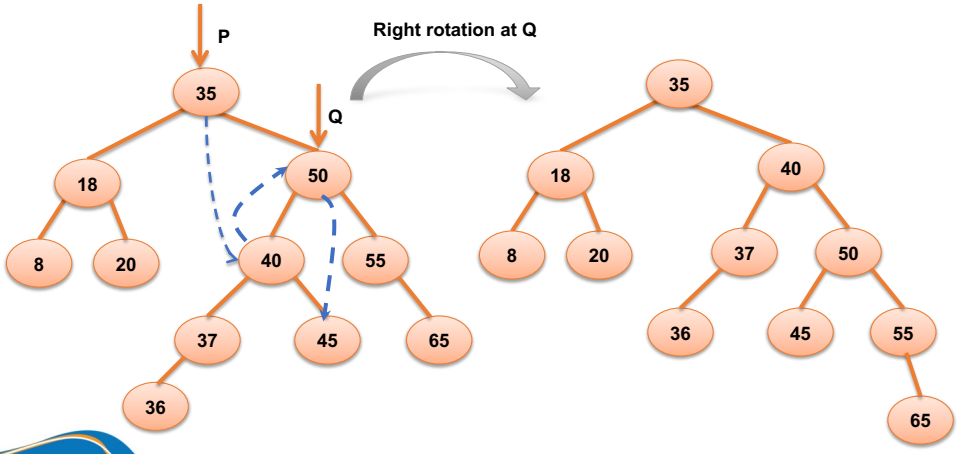
- Right-Left case:
  - Right rotation at P



79

fit@hcmus

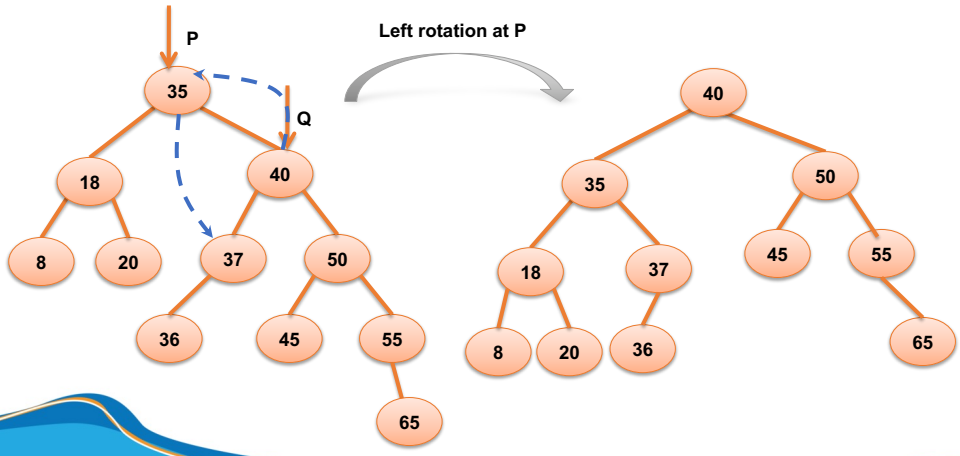
Un-balanced Resolving



80

fit@hcmus

Un-balanced Resolving



81

fit@hcmus

Un-balanced Resolving

- Left-Left case:
  - Right rotation at un-balanced node.
- Left-Right case:
  - Left rotation at un-balanced node's left child.
  - Right rotation at un-balanced node.

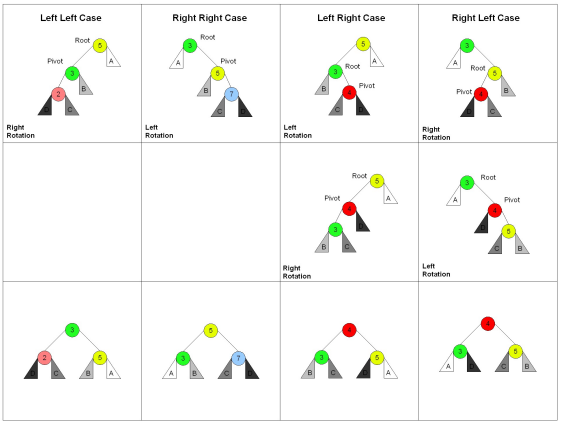
82

fit@hcmus

Un-balanced Resolving

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.



Source: Wikipedia