

AVL TREES

By Asami Enomoto

CS 146

AVL Tree is...

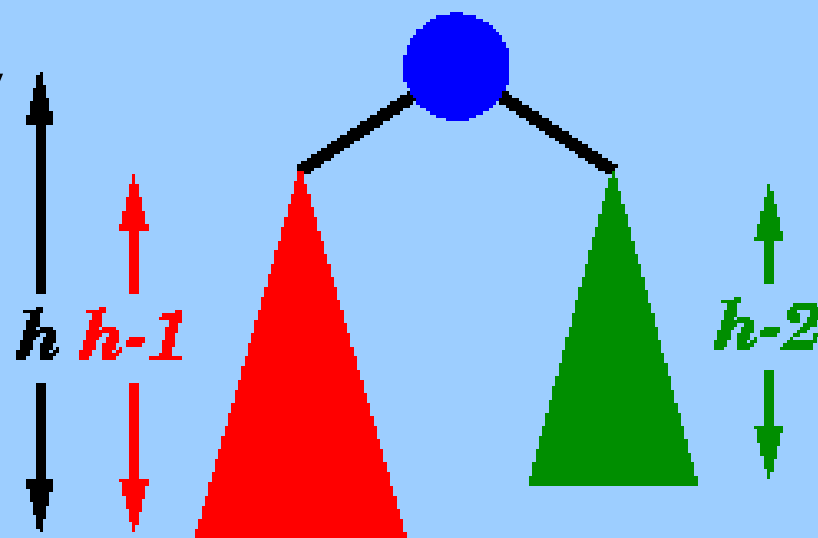
- named after **A**delson-Velskii and **L**andis
- the first dynamically balanced trees to propose
- Binary search tree with **balance condition** in which the sub-trees of each node can differ by at most 1 in their height

Definition of a balanced tree

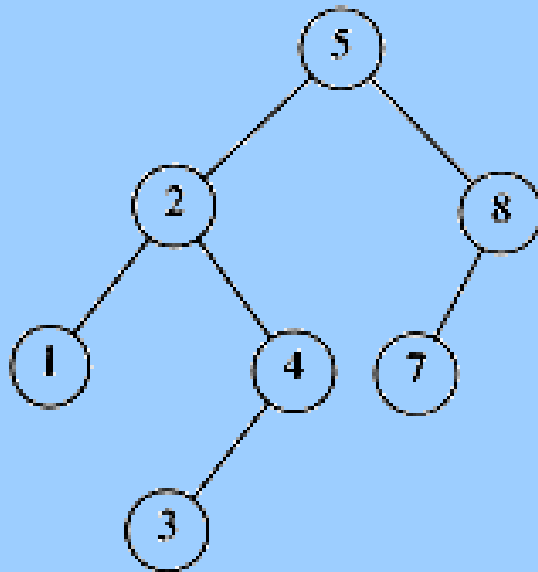
- Ensure the depth = $O(\log N)$
- Take $O(\log N)$ time for searching, insertion and deletion
- Every node must have left & right sub-trees of the same height

An AVL tree has the following properties

1. Sub-trees of each node can differ by at most 1 in their height
2. Every sub-tree is an AVL tree

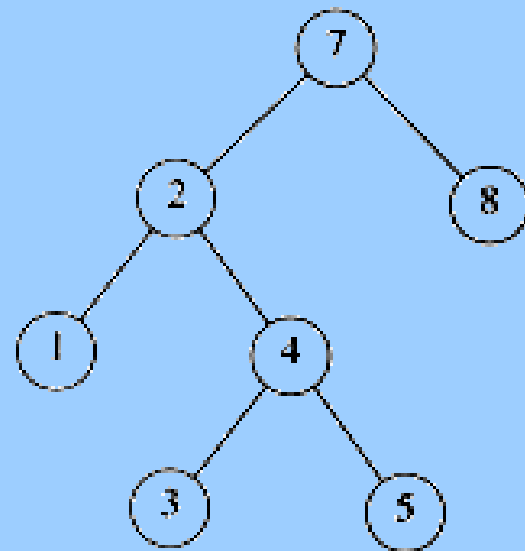


AVL tree?



YES

Each left sub-tree has height 1 greater than each right sub-tree



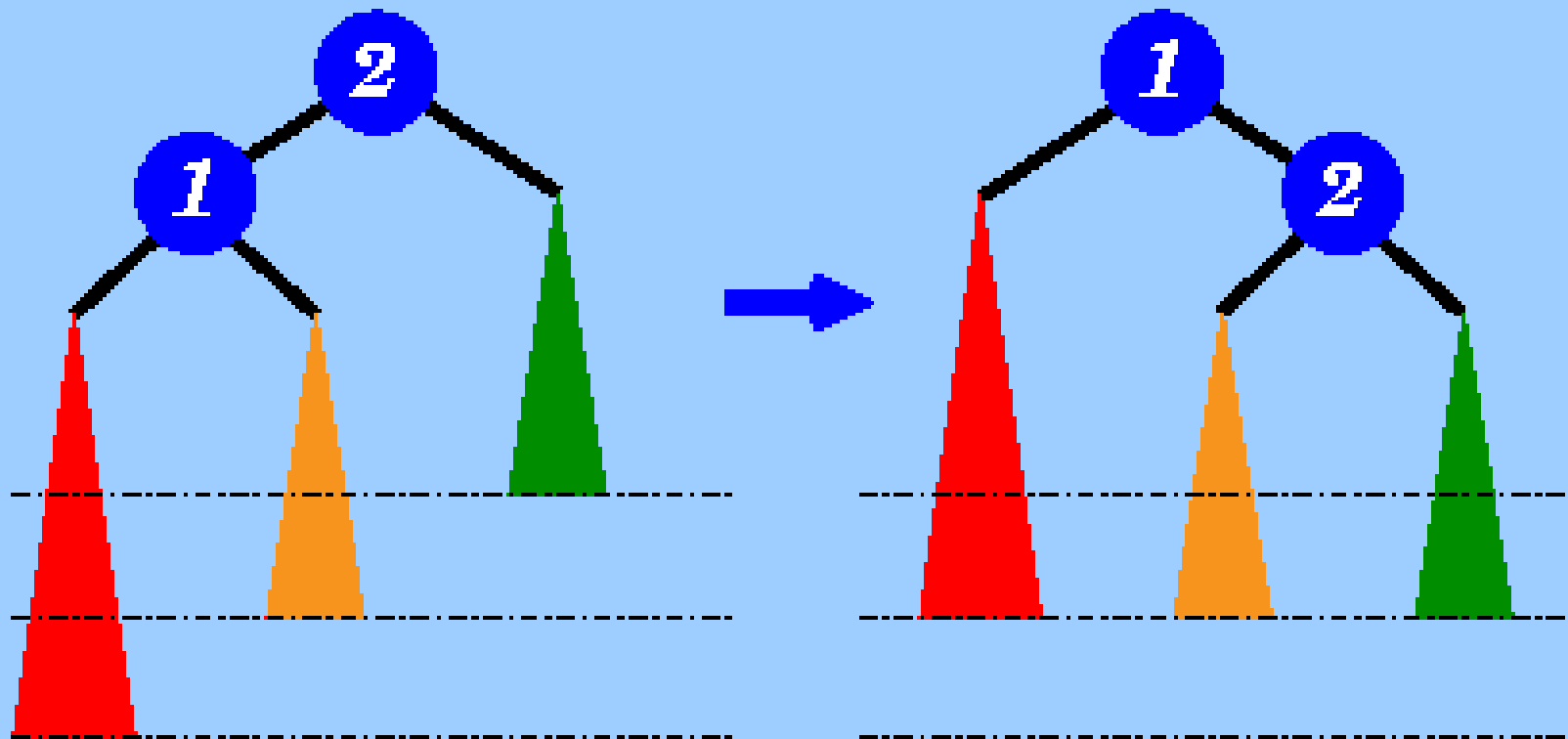
NO

Left sub-tree has height 3 but right sub-tree has height 1

Insertion and Deletions

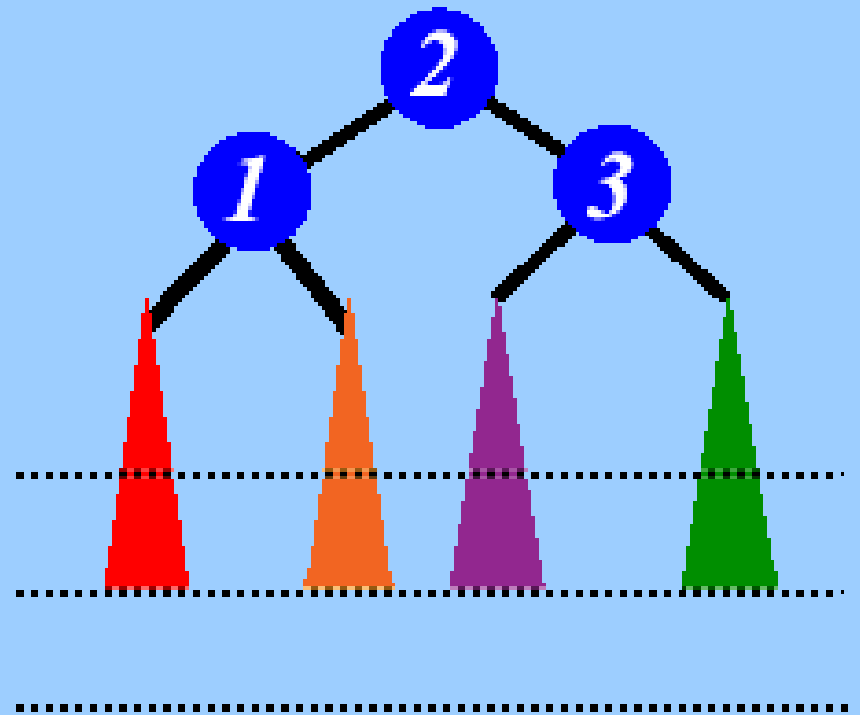
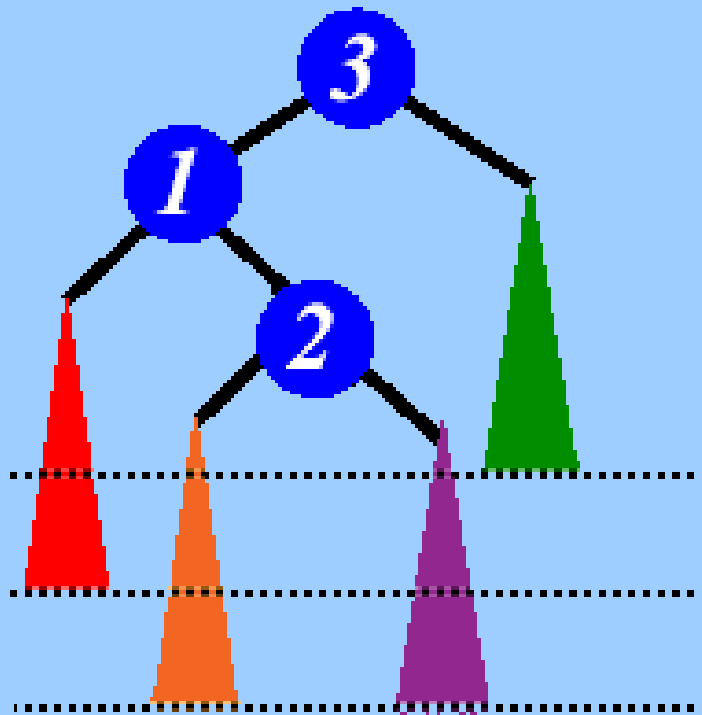
- It is performed as in binary search trees
- If the balance is destroyed, **rotation**(s) is performed to correct balance
- For insertions, one rotation is sufficient
- For deletions, $O(\log n)$ rotations at most needed

Single Rotation



left sub-tree is two level move ① up a level and
 deeper than the right sub-tree ② down a level

Double Rotation

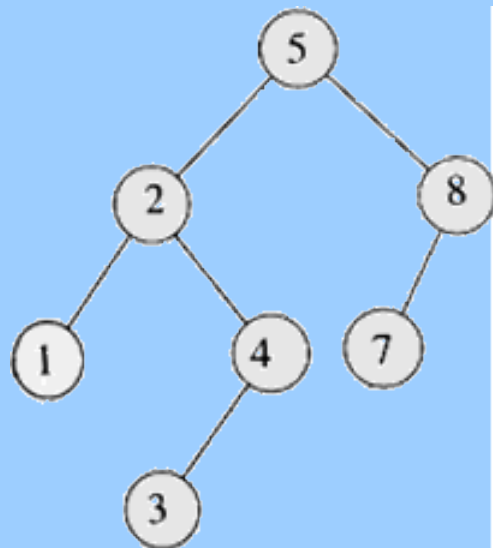


Left sub-tree is two level deeper than the right sub-tree

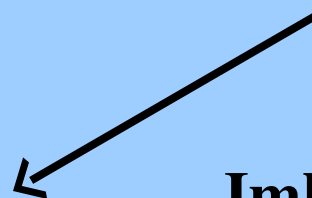
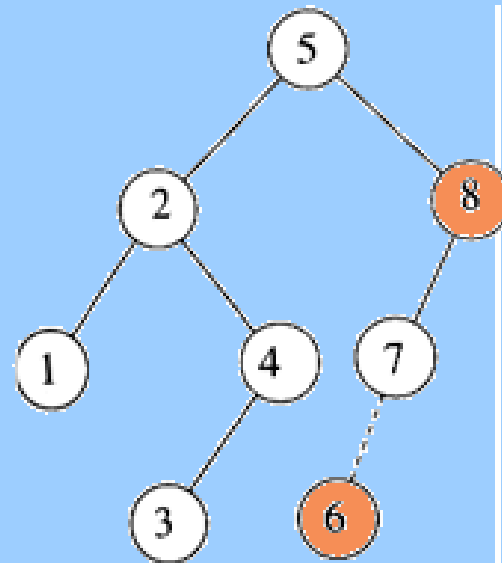
Move② up two levels and
③ down a level

Insertio

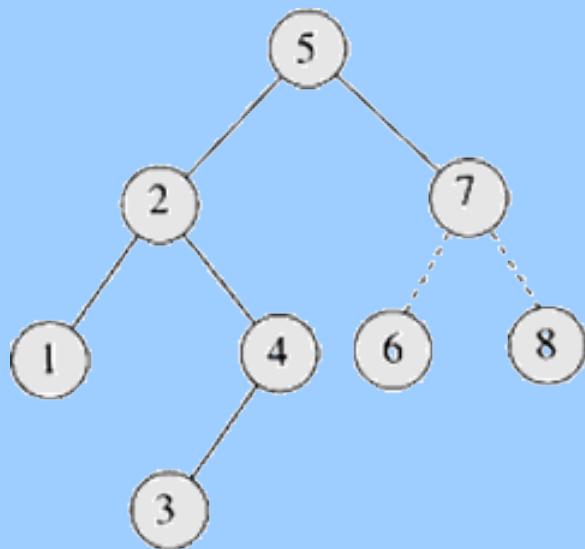
n

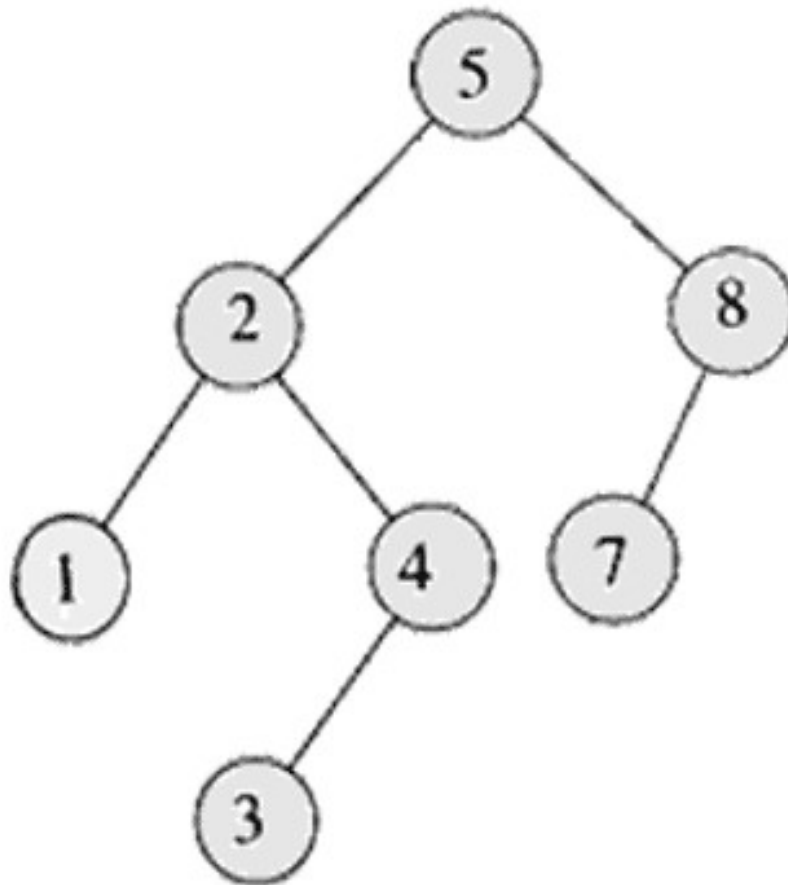


Insert 6



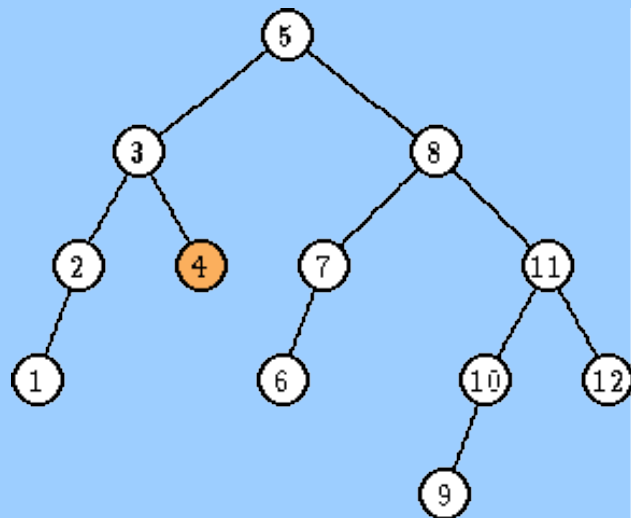
Imbalance at 8
Perform rotation with 7



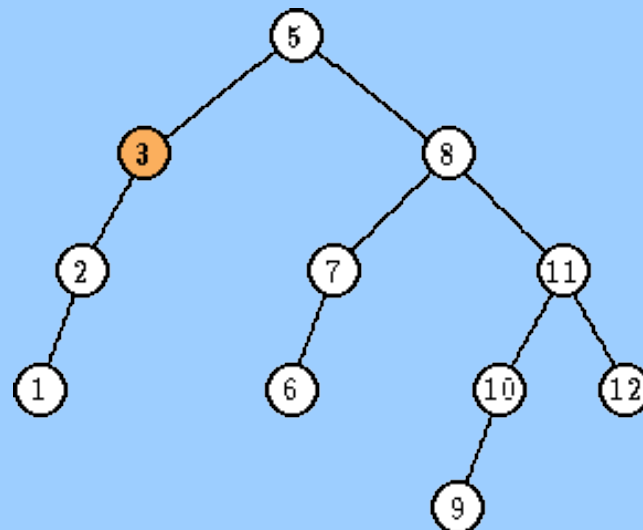


11/STR/MAN/VIS/1/A

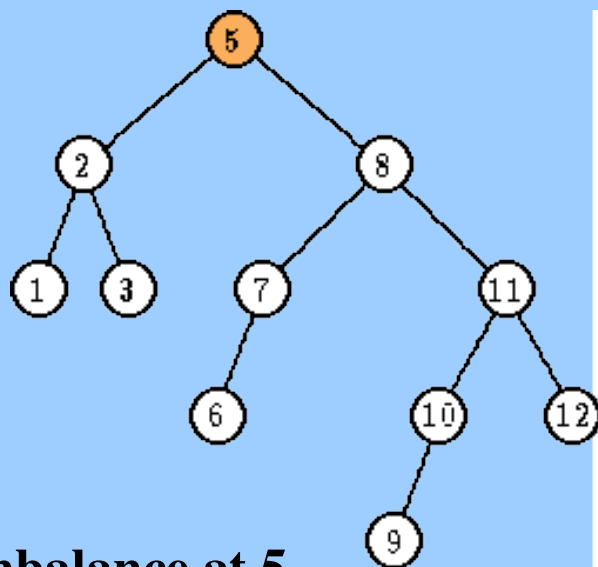
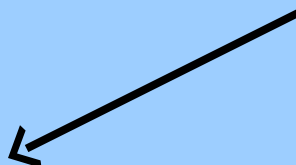
Deletion



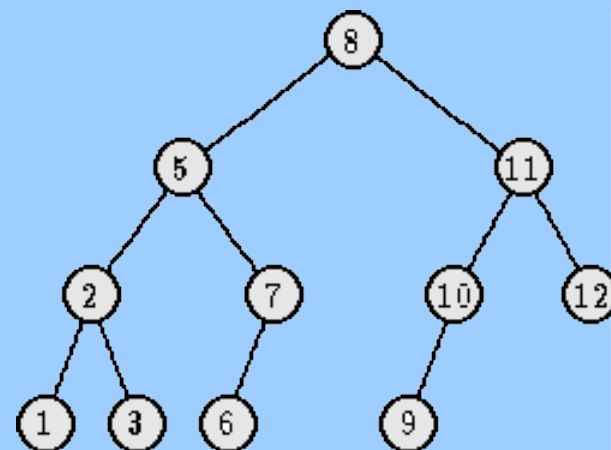
Delete 4

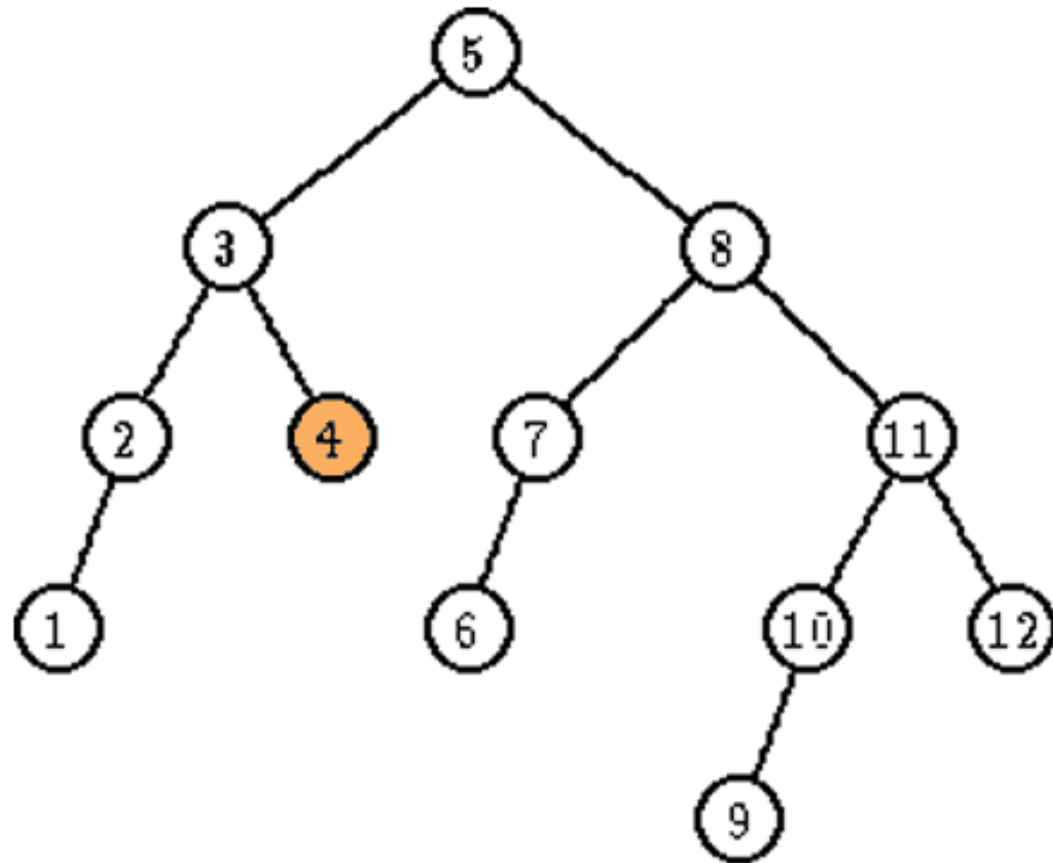


**Imbalance at 3
Perform rotation with 2**



**Imbalance at 5
Perform rotation with 8**





Key Points

- AVL tree remain **balanced** by applying rotations, therefore it guarantees **$O(\log)$** search time in a dynamic environment
- Tree can be re-balanced in at most **$O(\log)$** time