Height balanced AVL trees

Adelson, Velski 1962

Motivation –

BST’s have a worst case of

O(n) (when it’s really a linked list)

We’d like to keep the tree height O(log(n)),

hence the need to balance the tree

Height balanced AVL trees

Options

* Don't balance (O(n) worst case)
* Strict balance - keep it perfect
* Pretty good balance – keep it close (close == log(n))
* Adjust on insert/delete - “self-adjusting”

Height balanced AVL trees

**2-4 trees**

**Splay trees**

**Red black trees**

**Weight balanced trees**

**Lots of graduate degree theses**

Height balanced AVL trees

Balance factor of a node =

height(left subtree) - height(right subtree)

can not differ by more than 1

store the heights in the nodes

Height balanced AVL trees

Insert operation may cause balance factor to become 2 or –2

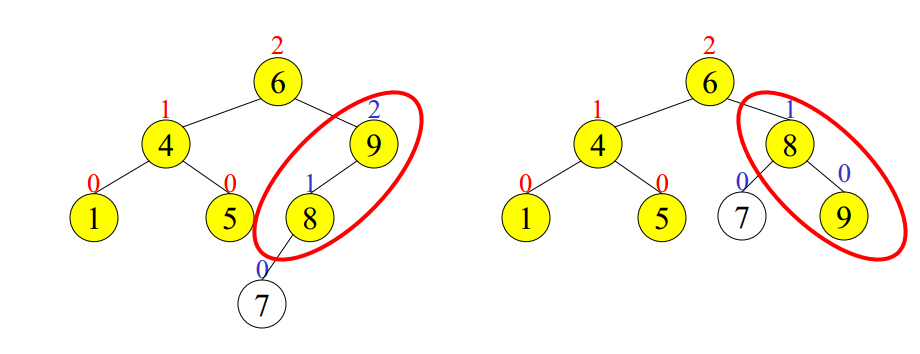
only nodes on the path from insertion point to root node have possibly changed in height

after the Insert, go back up to the root node by node, updating heights

If a new balance factor (the difference ht(left) – ht(right)) is 2 or –2, adjust tree by rotation

Height balanced AVL trees

Notion of ‘rotation’



Height balanced AVL trees

Insertion algorithm, 4 cases:

Outside Cases (require single rotation) :

Insertion into left subtree of left child of δ

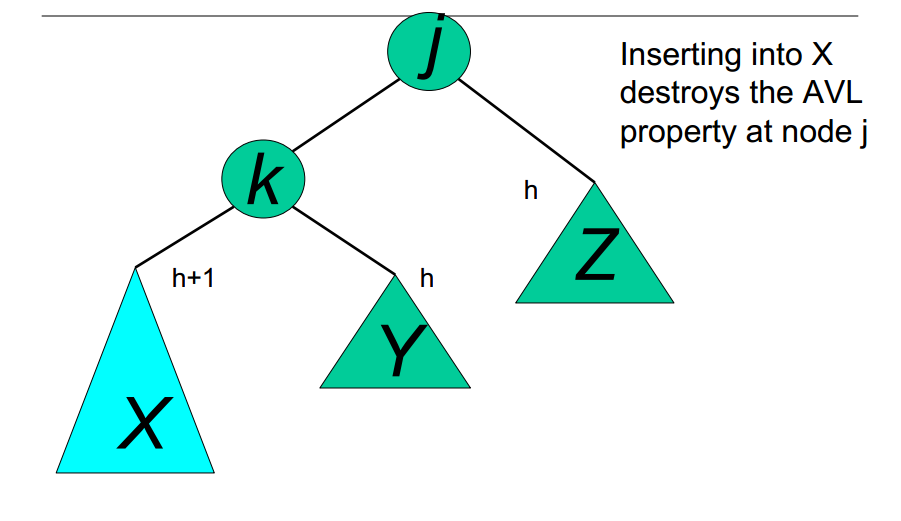
Insertion into right subtree of right child of δ

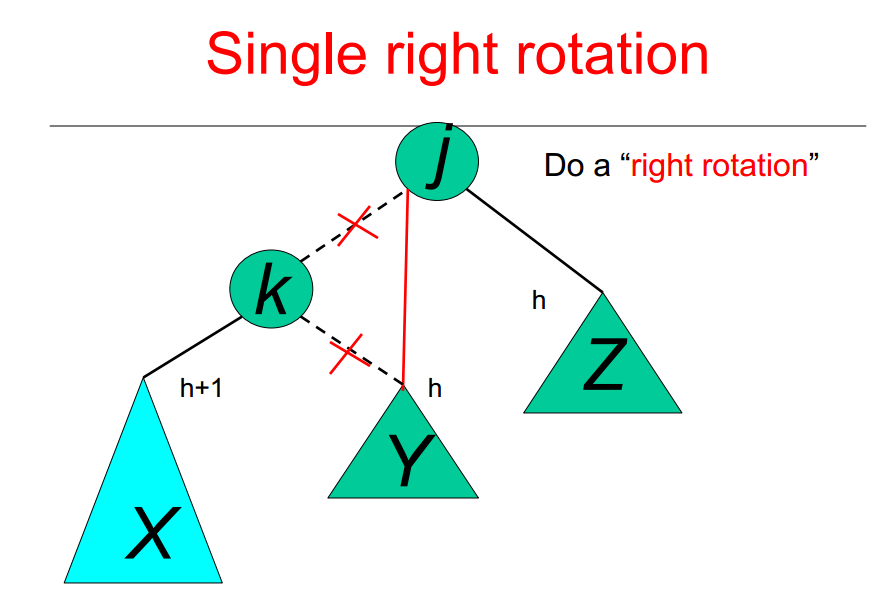
Inside Cases (require double rotation) :

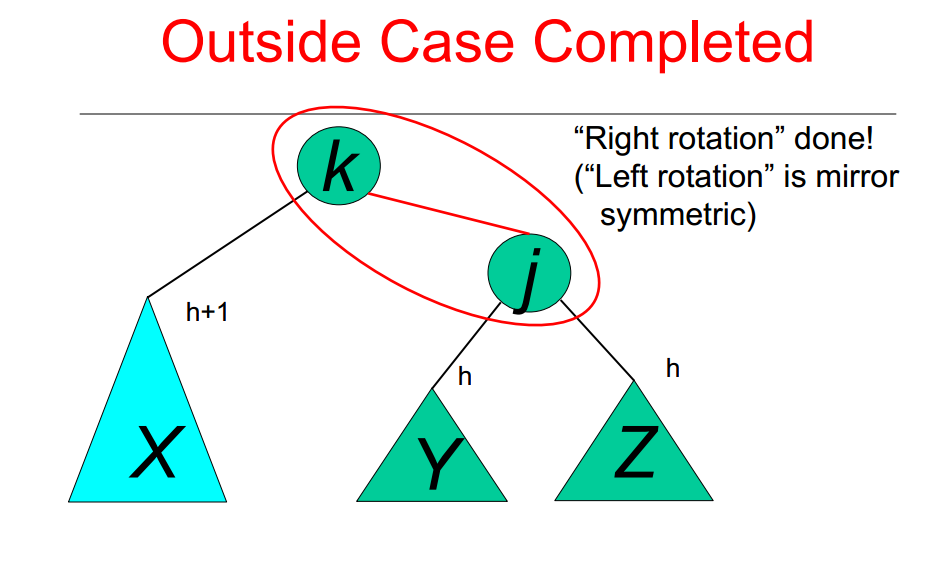
Insertion into right subtree of left child of δ

Insertion into left subtree of right child of δ

OUTSIDE







U washington

INSIDE

