



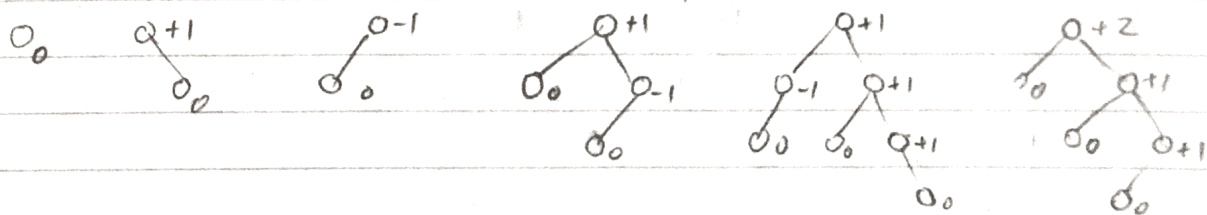
Housekeeping • Friday's tutorial is moved to room BA1180.

ADT	Ops	Data Structure
• Dictionary	• Insert	- BST
	• Delete	• <u>"Balanced Trees"</u> :
	• Search	- 2-3 trees - red-black trees
		- B-trees, - Adelson-Velski-Landis trees

• Problem with BSTs:

Suppose we start with  and add 4, 5, 6...  result: height $\Theta(n)$
ops $\Theta(n)$:~

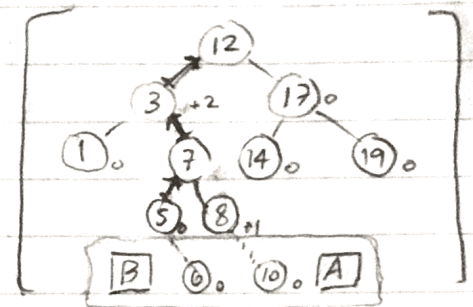
- Height(v): # of edges on longest path from node v to leaf +
- BalanceFactor(v): Height(v .rightchild) - Height(v .left)
- ALL Tree \equiv BST such that \forall nodes v , $BF(v) \in \{-1, 0, 1\}$
 - -1) v is "left-heavy" 0) Balanced 1) v is "right-heavy"
 - empty AVL has height -1, BF -1,
 - single node AVL has height 0, BF 0
 - balance factor stored in every node



- AVL of m nodes has height $\Theta(\log m)$ [$\leq 1.44(\log m + 2)$]
- can do inserts, deletes, while maintaining tree balance in $O(\log m)$
lecture tutorial

Example: (i) $S = \{1, 3, 7, 12, 14, 17, 19\}$

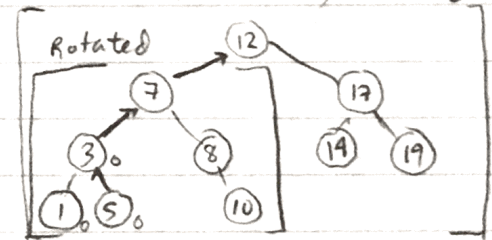
(ii) `INSERT(8);` add 8 as with normal BST, then adjust BF of ancestor nodes, all BFs $\in \{-1, 0, 1\}$ so done!



(iii) `INSERT(5);` //, adjust BF of ancestors.

A (iv) `INSERT(10);`

- adjusting BFs after insert creates a BF of +2 at $v=3$, and 3's right i.e. subtree $v=7$ is right-heavy
- **Left rotate** subtree $v=3$, changes three pointers (arrows)
- rotation & reset BFs in constant time
- rotation restores balance & minimizes height of rotated subtree while maintaining BST property



(iv) `INSERT(6);`

- adjusting BFs results in subtree 3: +2 and subtree 7: -1
- not same sign! must do double rotation
- rotate left heavy subtree 7 to make it right-heavy, then left rotate right-heavy subtree as in [A]

