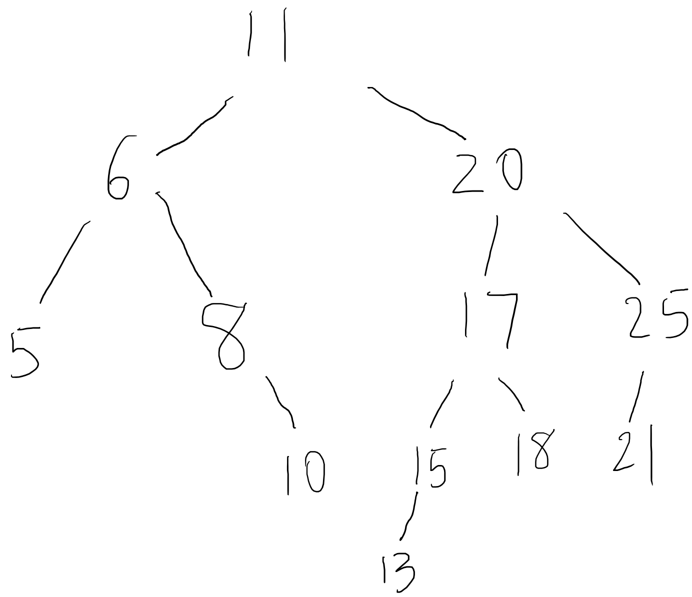
CSCB63 TUT2 WEEK8
andrewyk.leung@mail.utoronto.ca
https://snoopysnipe.github.io/ta/b63f20/

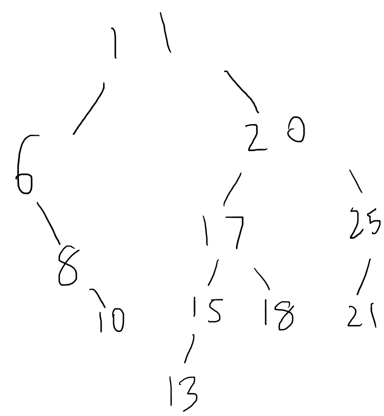
1) AVL Deletion example

2) AVL Augmentation

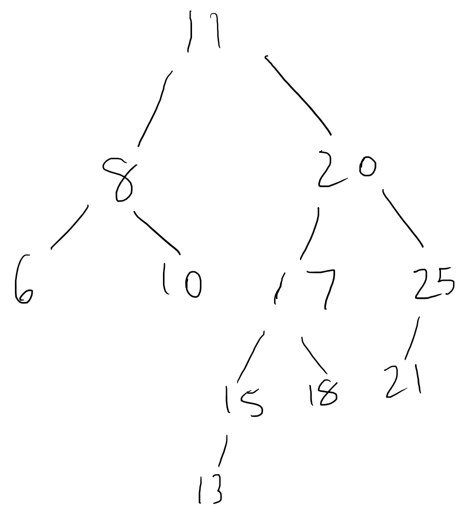
1)



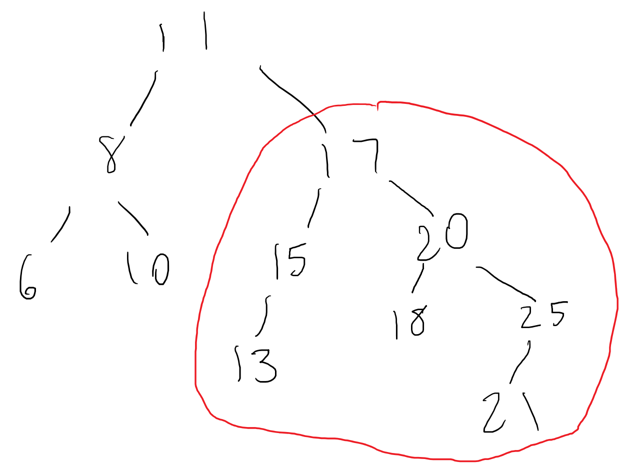
Delete 5

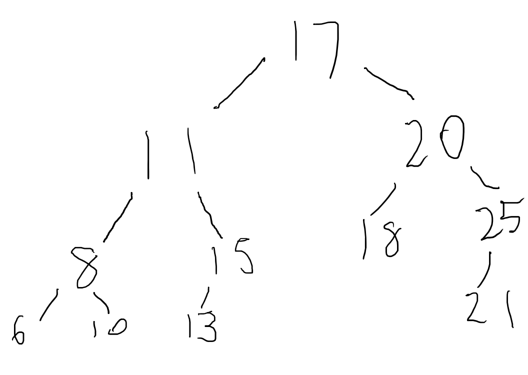


Rebalance at 6, CCW



Rebalance at 11, CW at 20, CCW at 11





2) AVL Augmentation

↳ Idea: Support new operations that keep insert and delete $O(\lg n)$. Add extra fields to nodes to help with that. (Usually, operation is $O(1)$ and defers most work to insert and delete)

Notation: $\lg n \equiv \log_2 n$

ex: closest(): return the/a closest pair of elements in the tree

e.g: 10, 20, 22, 30, 33, 35, 39, 44
closest() returns either (20, 22) or (33, 35)

Solution:

extra fields:
1) cp: a closest pair in this subtree
2) min: the minimum key in this subtree
3) max: the maximum key in this subtree

how to update node x:

1) cp: options — ① x.left.cp
② x.right.cp
③ x.key, predecessor of x.key (x.left.max)
④ x.key, successor of x.key (x.right.min)

O(1) to read and compare

2) min : x.left.min O(1)

3) max : x.right.max O(1)

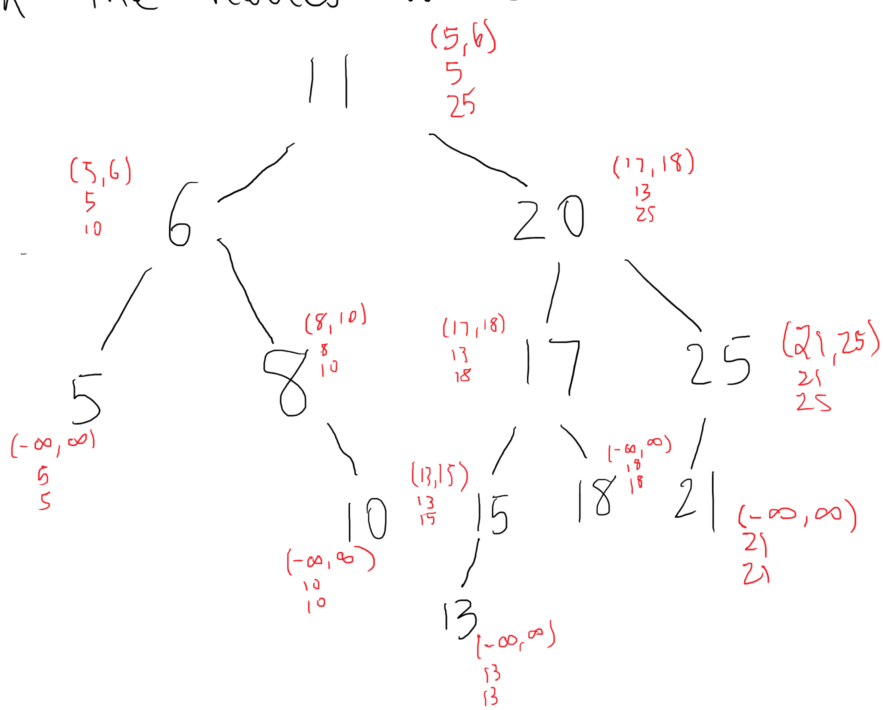
Update during insert and delete for the nodes that are on the path of the ancestors. Insert and delete stays O(lg n).

Algorithm for closest():

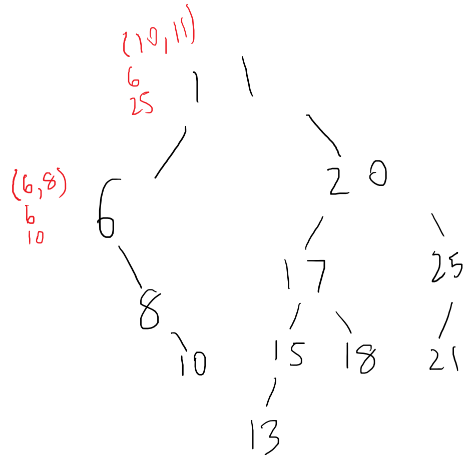
read root.cp

In conclusion, closest() can be done in O(1). But most of the work is done when updating fields in the nodes whenever there is an insert or delete.

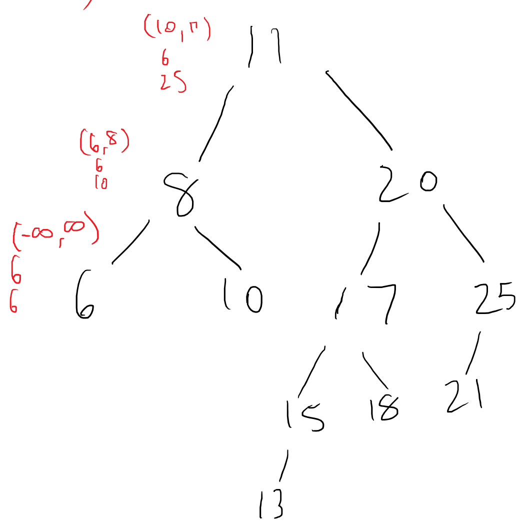
ex:



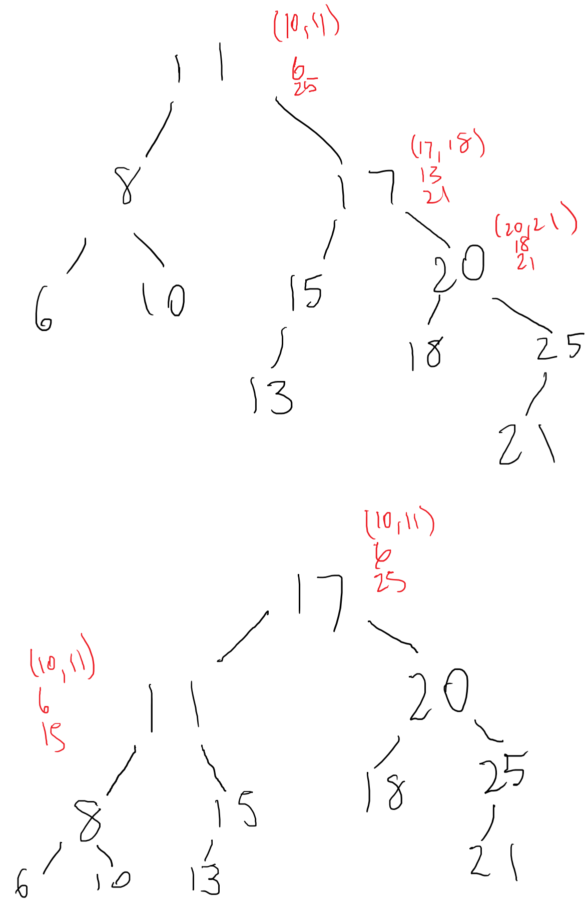
Delete 5



Rebalance at 6, CCW



Rebalance at 11, CW at 20, CCW at 11



Exercise: make sure field values at each intermediate step of other nodes stay the same