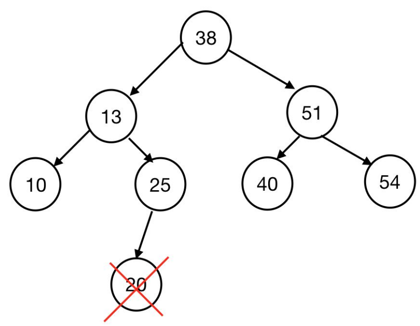
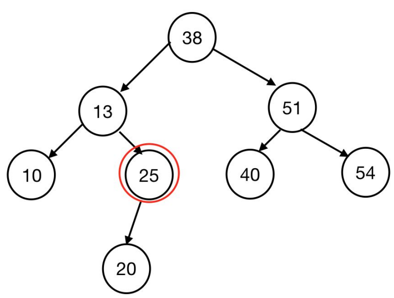
CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

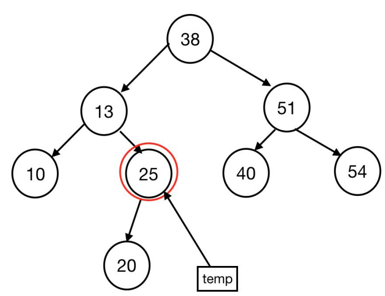
By Wenjie

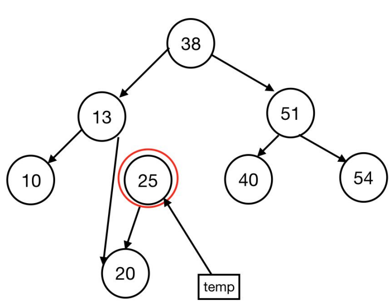
- **Remove in BST**
    - First of all, we need to find the element that we want to eliminate.
    - Then we start the removal steps:
        - If the node we want to delete is a leaf, we can just delete it.
            - Eg. _remove(20)

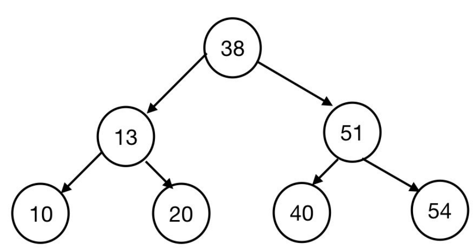


        - If the node we want to delete has only one child, we can delete as like in the linked list - just replace that node with it's child.
            - Eg: _remove(25)





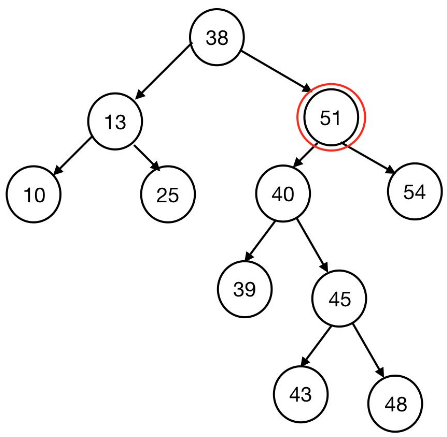


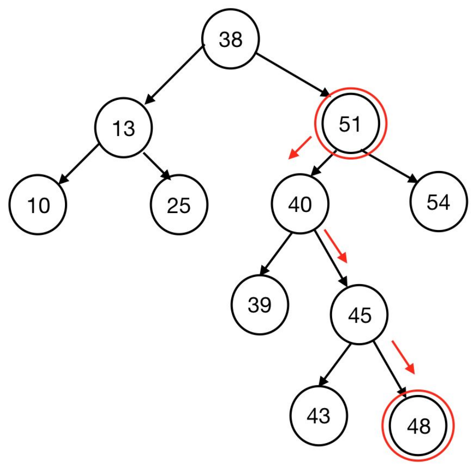


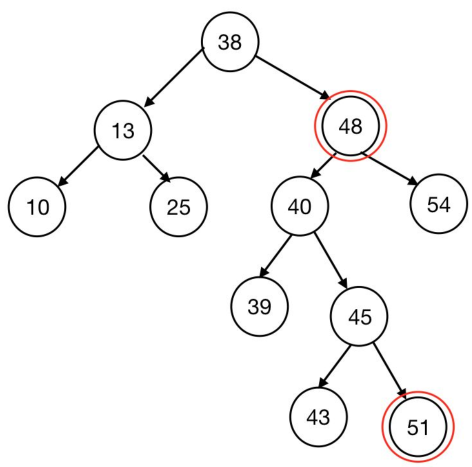
CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

By Wenjie

- If the node we want to delete has two children:
  - First to find inorder predecessor (IOP) - the largest node in the left subtree as well as the rightmost node of the left subtree
  - Swap IOP and the node we actually want to delete
  - The node is now a leaf, so we can remove it.
    - Eg: _remove(51)







- Find / insert / delete : O(h)

- **The relationship between H and N**
  - We observe every operation on BST in terms of the height of the tree.

- m(n) = max number of nodes
  $= \{0, \quad h = -1$
  $1+2m(h-1), \; h > -1$
  $= 2^{h+1} - 1 \qquad //O(2^h)$

- **Proof by induction**
  - Base case: $m(0) = 2^{0+1} - 1 = 2 - 1 = 1$
  - Inductive step: $m(h) = 1 + 2m(h-1)$
    $= 1 + 2(2^{(h-1)+1} - 1)$
    $= 1 + 2 * 2^h - 2 = 2^{h+1} - 1$
- Therefore we have $(2^{h+1} - 1) \le n$. By solving it we have $h \le \log(n+1) - 1$
- The height of the tree is depend on the order of which data to insert
- For every BST, O(log n) is the lower bound running time and O(n) is the upper bound running time.