Removal from BST

Michael Wollowski

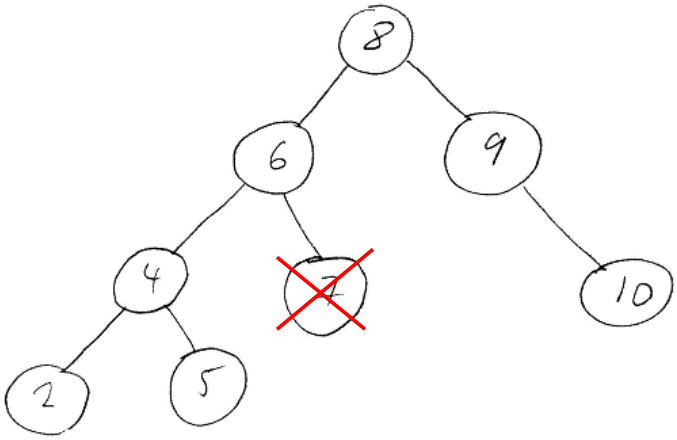
Three Cases

- When removing a node from a BST, there are three cases.
- The node to be removed is:
  1. A leaf
  2. Has once child
  3. Has two children

Case 1: Removing a Leaf

- A leaf node has NO children
- If the node to be removed is a leaf, just remove it.
- In Java: Set the link to it to *null*.

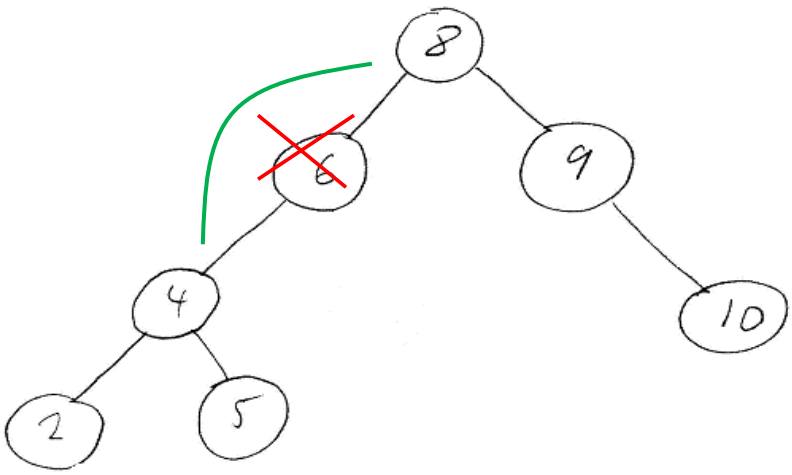
Example: Removing a Leaf: 7



Case 2: Removing a Node with 1 Child

- If the node to be removed has one child, just remove it and make its grandchild the new child.
- In Java: Set the link to the child so that it points to its grandchild.

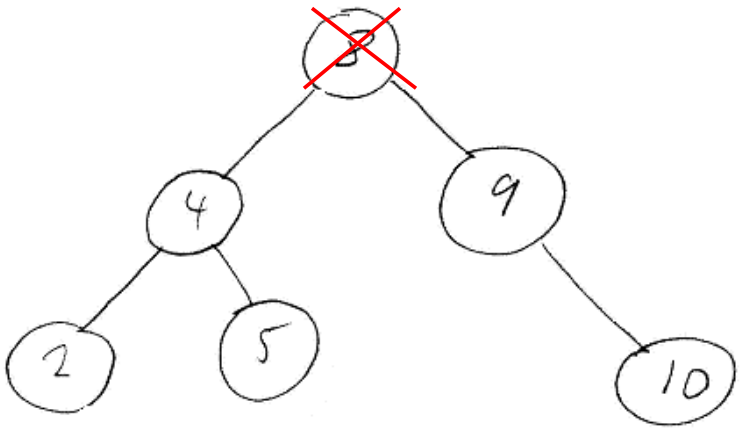
Example: Removing a Node with 1 Child: 6



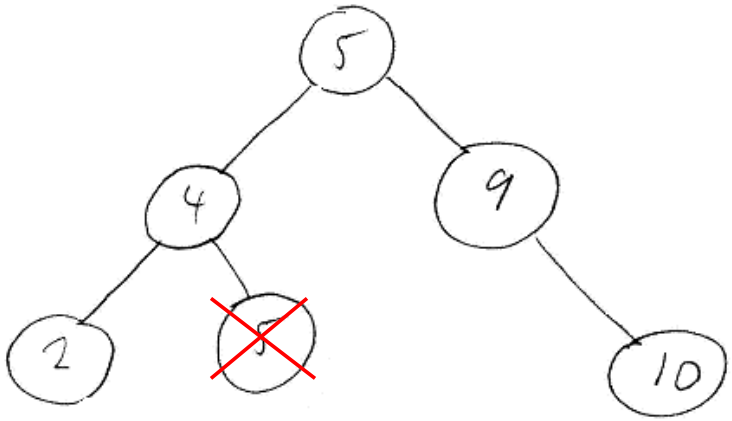
Case 3: Removing a Node with 2 Children

- If the node to be removed has two children, find the largest child in the *left* sub-tree.
- Copy that value into the node with the two children.
- Remove the node that contains the copied value.
- NOTICE: You could instead find the *smallest* value in the right sub-tree.
- NOTICE 2: Our test cases assume you work with the left sub-tree.

Example: Removing a Node with 2 Children: 8



Example: Removing a Node with 2 Children: 8



Example: Removing a Node with 2 Children: 8

- What if 5 has a child?
- Where would it be?
- Do we need to worry about it?

Return Type/Exceptions

- This method returns TRUE, if the element was successfully removed and FALSE otherwise
- This method throws an *IllegalArgumentException* if an attempt is made to remove “null.”