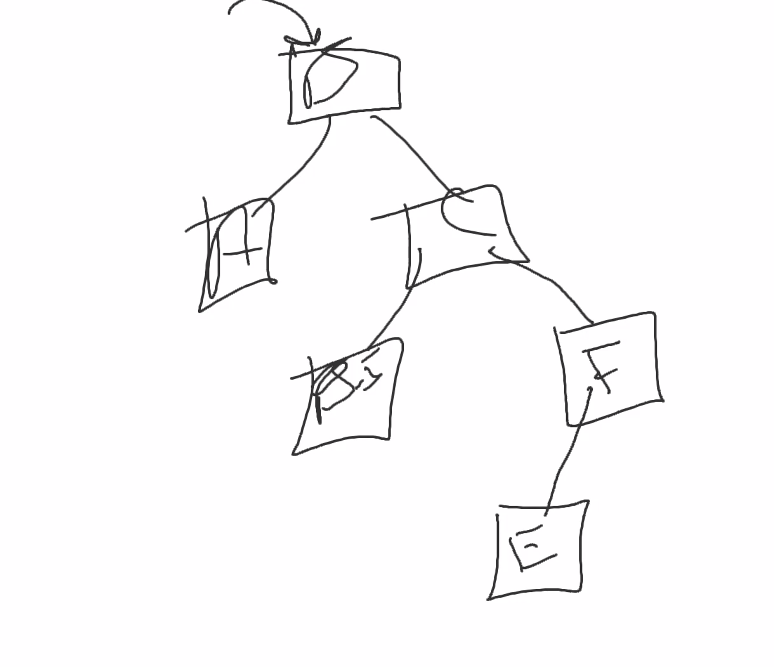
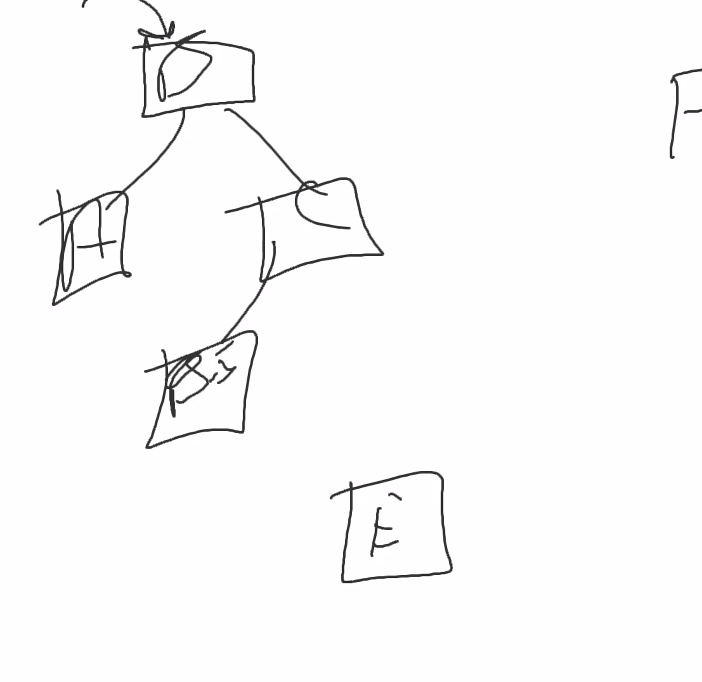
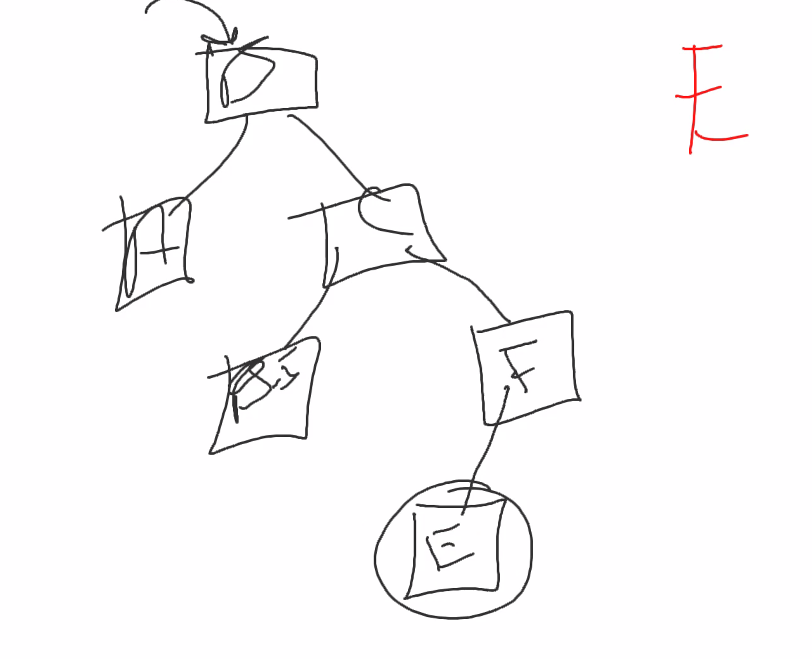
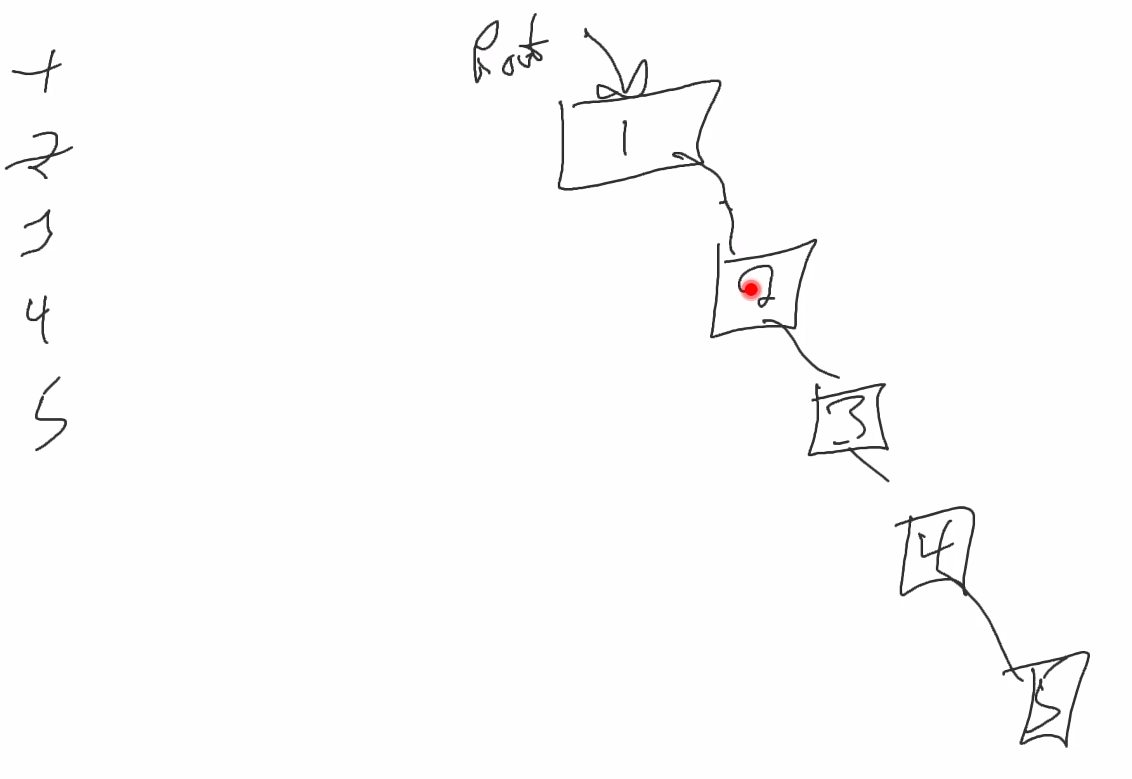
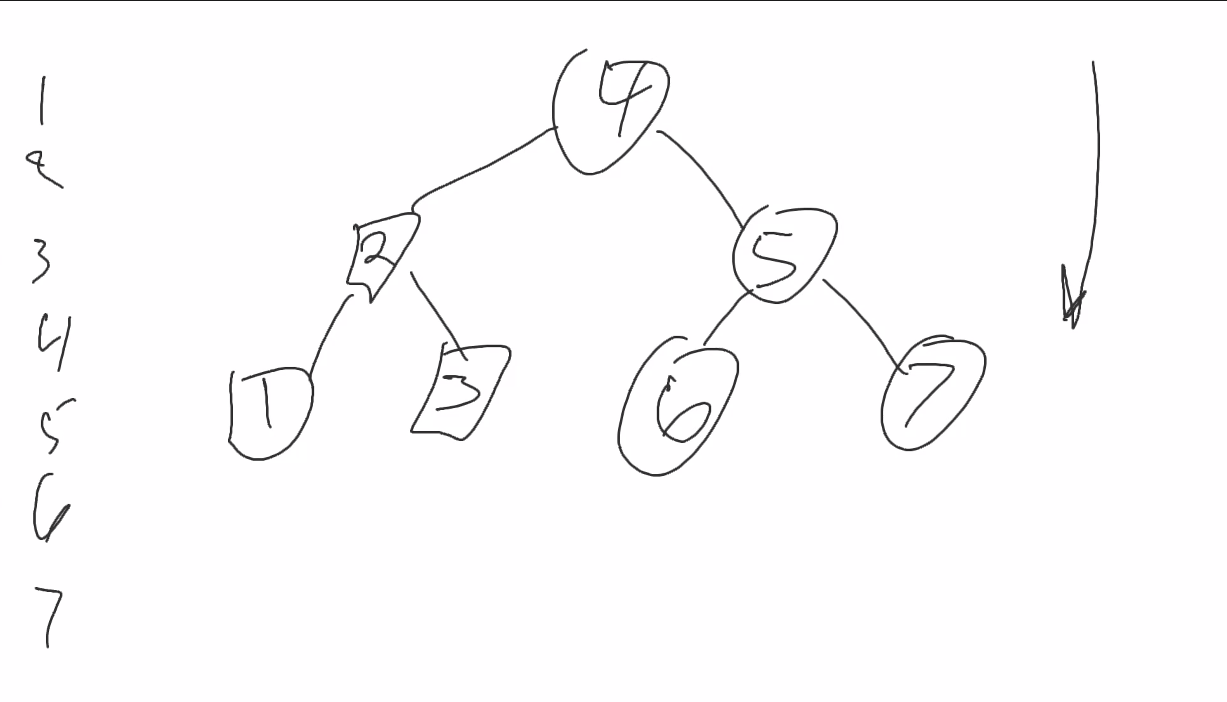
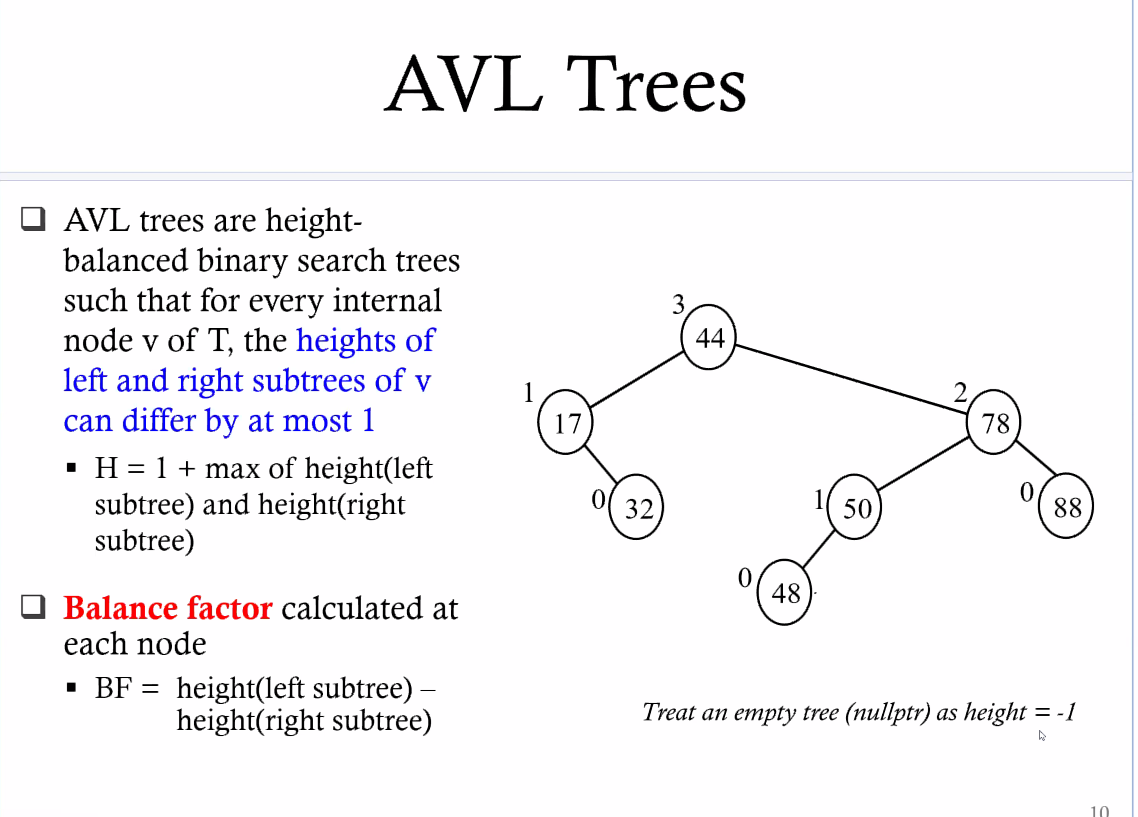
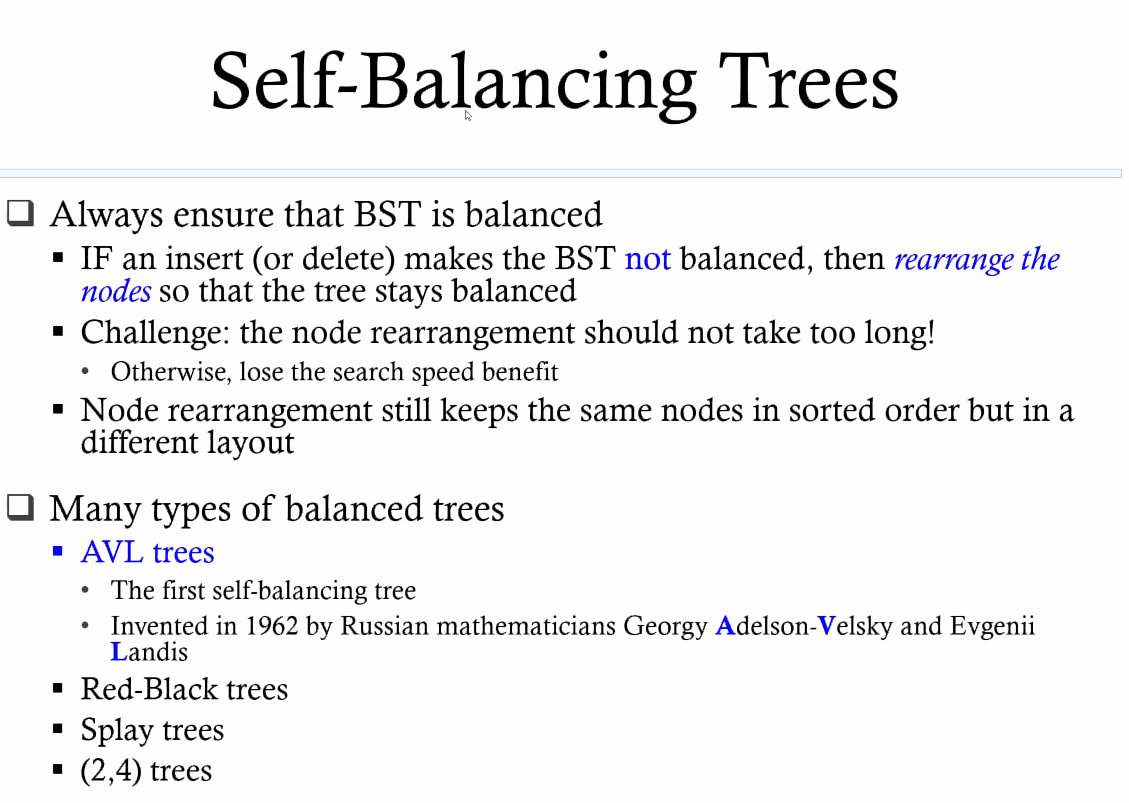
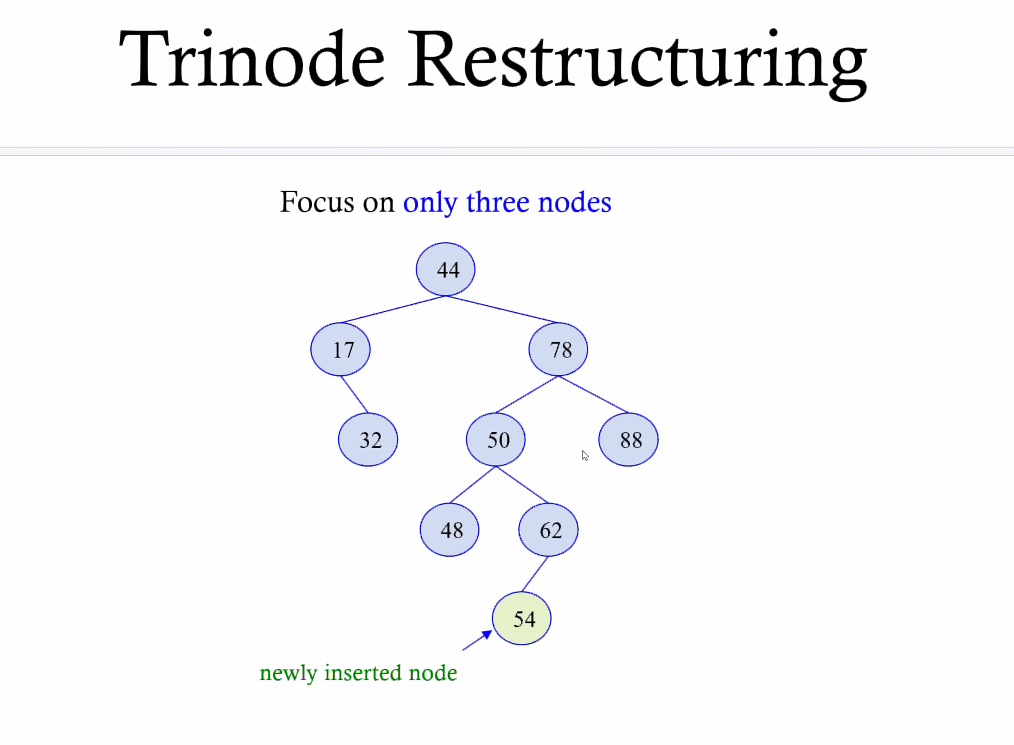
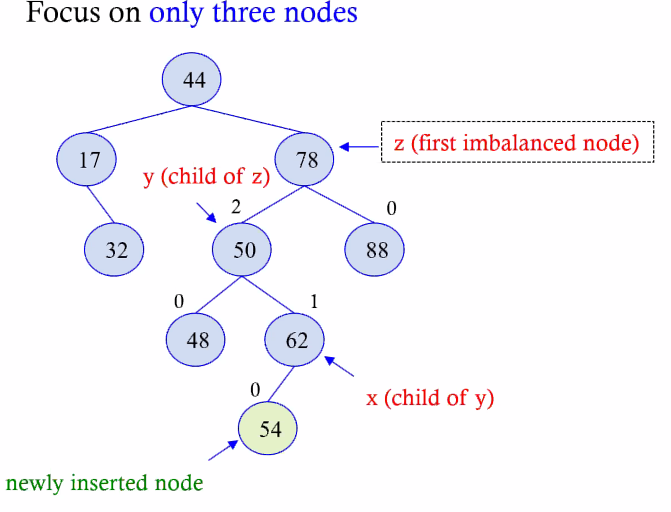
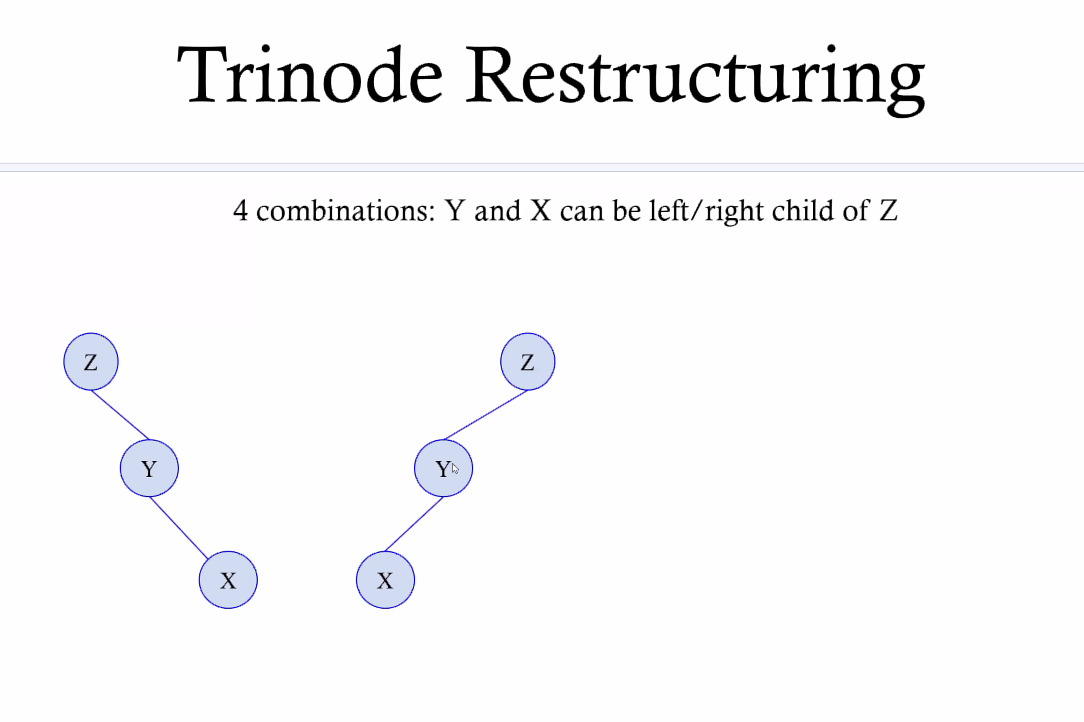
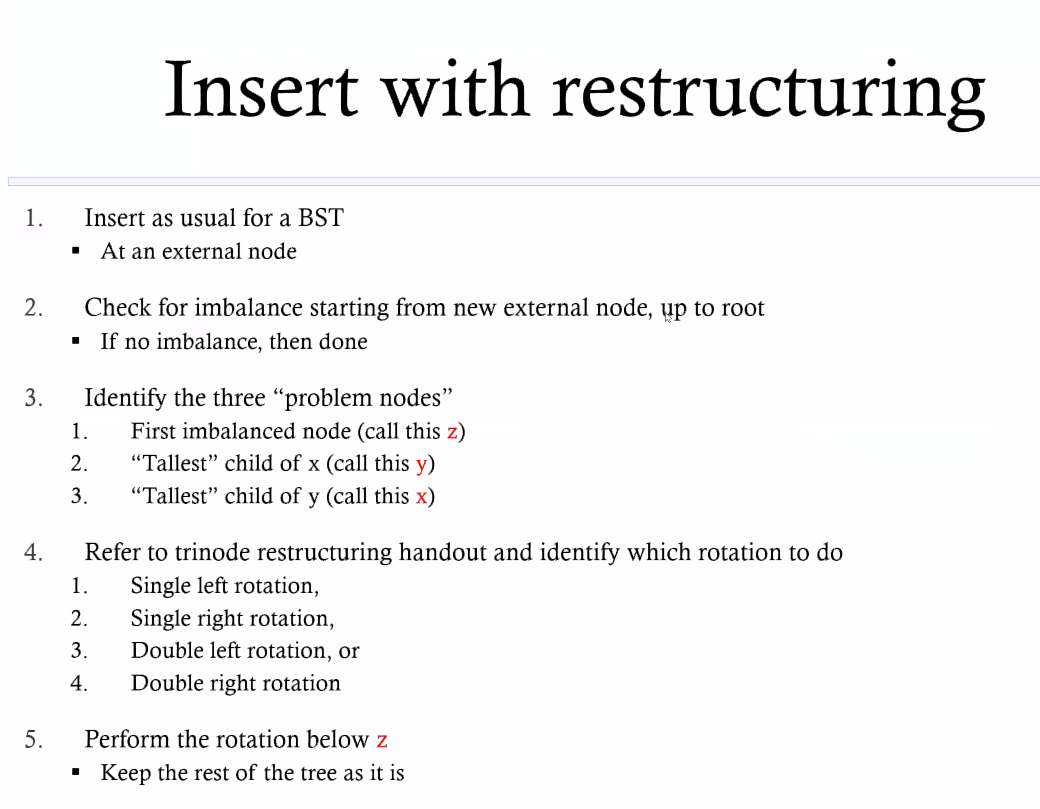
Lecture 20

CPSC 131

11/9/2020

1. Binary Search Tree
   1. Tree ABC
      1. What we’re doing: Recursing to the right of the Binary Search Tree but there is no tree   
         
      2. So we’re **inserting** E below F; we’re forming a tree here
      3. When we need to **erase**, we can erase E because it has no children or erase a node with one child or one with 2 children  
         
         1. It really comes down to three different forms of options
         2. Take the parent pointer and, instead of point to parent, point to grandparent
      4. **Finding**
         1. We go through each branch until we get to E
         2. We copy the contents of B.5, replace B with B.5, and then delete B. Then delete the successor.
2. Root
   1. It’s a binary search tree because it’s in order from greater to less then  
      
      1. Surprisingly no different from a doubly linked list
3. What if we want our binary search tree to look like this?  
     
   1. This complete, perfect, full tree. IT is also self-balancing.
      1. This means that the hiehgt of the Left Subtree is 1
4. AVL Tree
   1. AVL is a person’s initials
   2. But most of all, they are binary search trees  
      
   3. Self-Balancing Trees  
      
   4. Trinode Restructuring  
      
      1. First, find the offending node  
         
      2. Then you can fix the imbalance
   5. The Patterns used to fix the imbalance  
      
   6. Insert with restructuring  
      
   7. Trinode Restructuring Cheat Sheet  
      