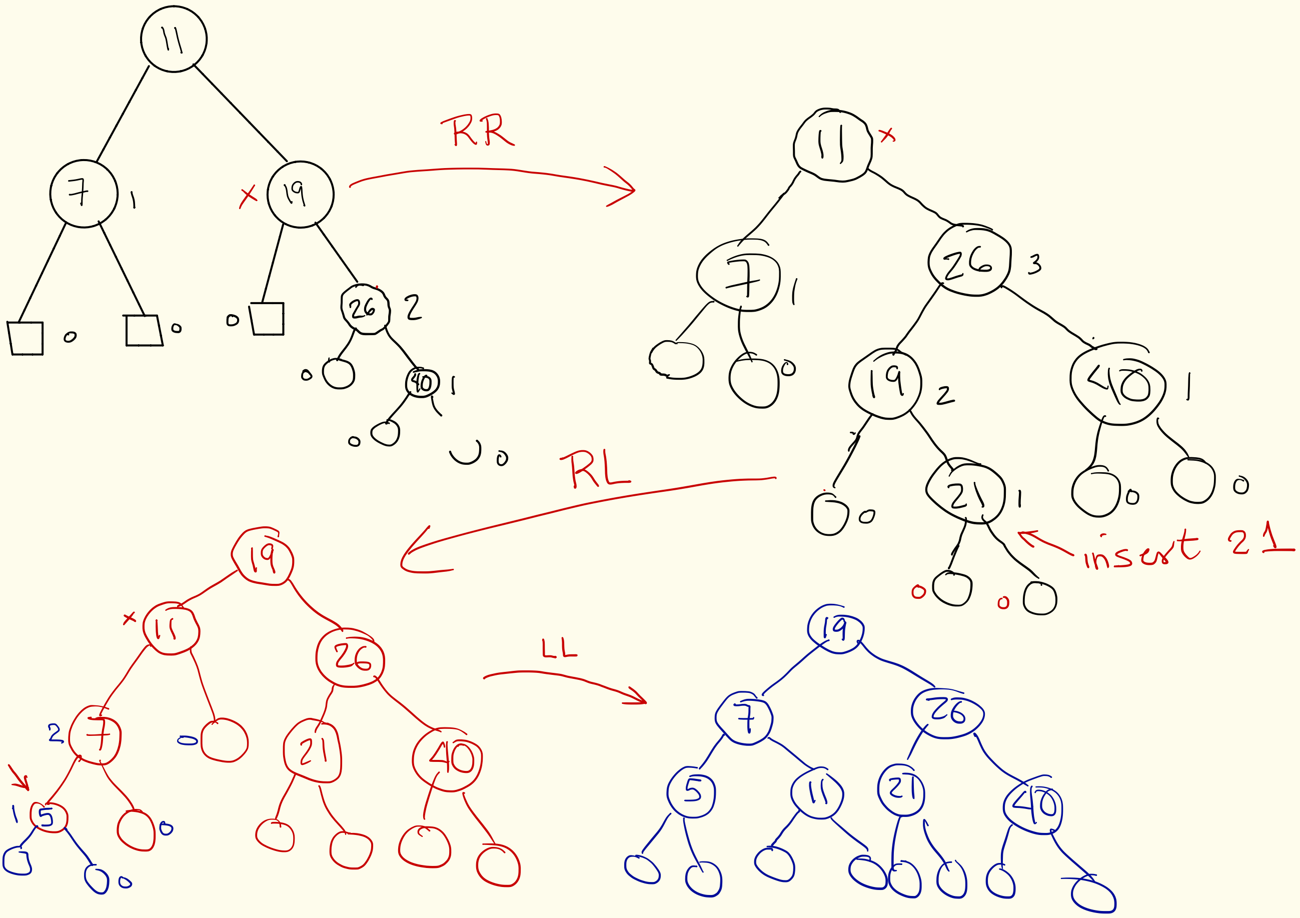
insert 40, 21, 5



Algorithm putAVL (*r*, *k*, data)

In: Root *r* of an AVL tree, record (*k*,data)

Out: {Insert (*k*,data) and re-balance if needed}

```
$O(\text{height})$
put(r,k,data)    // Algorithm for binary search trees
$c_1$
Let p be the node where (k,data) was inserted
while (p ≠ null) and (subtrees of p differ in height ≤ 1) do
    $c_2$  p = parent of p
$c_2 \times \text{height}$
if p ≠ null then rebalance subtree rooted at p by
                 performing appropriate rotation    $O(1)$
```

$f(n)$ is $O(\text{height}) = O(\log n)$

Algorithm removeAVL(r, k)

In: Root r of an AVL tree, key k

Out: {Remove k from the tree and rebalance, if needed}

```
$O(height)$        remove(r, k)   // Same algorithm for BST
$c_1$              Let p be the parent of the removed node
                   while $p \neq null$ do {
$c_2 \times height$    $c_2$   if subtree rooted at p is not AVL then
                                   Rebalance it
                               $p \leftarrow$ parent of $p$
                   }
```

$$f(n) \text{ is } O(height) = O(\log n)$$