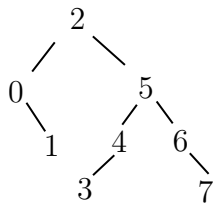
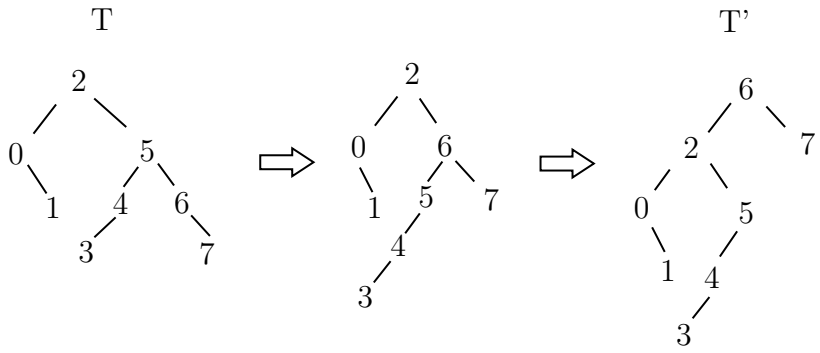
# Question 1:

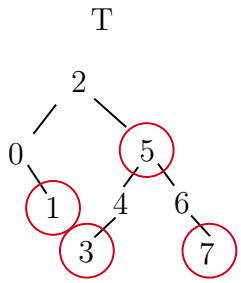
## 1.1




## 1.2




## 1.3



## 1.4

- You cannot make T' into a RB tree because the length of left subtree (6-2-5-4-3) is more than twice of the length of right subtree (6-7) so it violates the rule.

# Question 2:

## 2.1

- You cannot always keep this tree balanced. For example, if a tree contains only 1, lots of 1, then in order to make it symmetric and "balanced", you have to choose a 1 as root and move half of the 1 into the right subtree, then it will violate the rule that every element on the right is bigger than its parent, so the tree cannot be always balanced.

## 2.2

- Start from root.
- If element == root, retrieve the element and the left subtree recurses;
  If element < root, left subtree recurses;
  If element > root, right subtree recurses
- When the element is null (no node), return

## 2.3

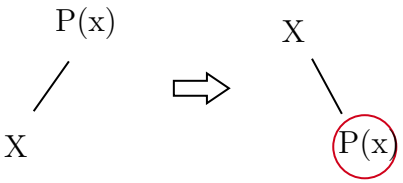
- First we know that left subtree is smaller or equal to parent, and right subtree is bigger than parent, so the elements with same value can only stay in one subtree.
- Compare the element and root (current parent), so we can know where all the duplicated element are.
- Once we know the subtree for them, we can recurse on it. As long as we find a duplicated element, we retrieve it. And continue till we reach the null element, which means nothing left in that subtree so we can return.

## 2.4

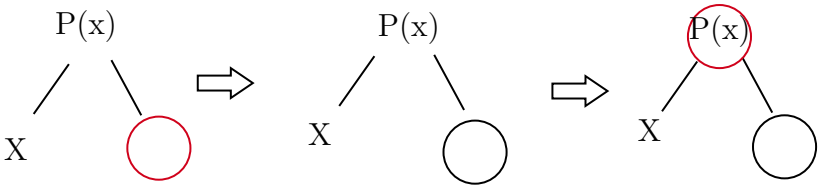
- This function recurses from the root to the leaves and retrieves the element when necessary so the total runtime is:
- O(height of the tree + number of duplicated elements)

## Question 3:

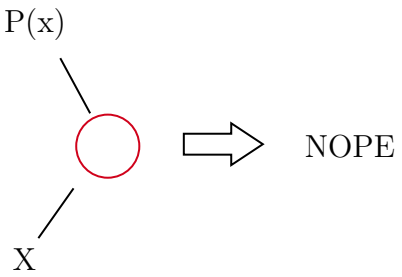
- A) Just right rotate P(x) and change P(x) into red note:




- B) Change the right child of P(x) to black, then change P(x) to red:

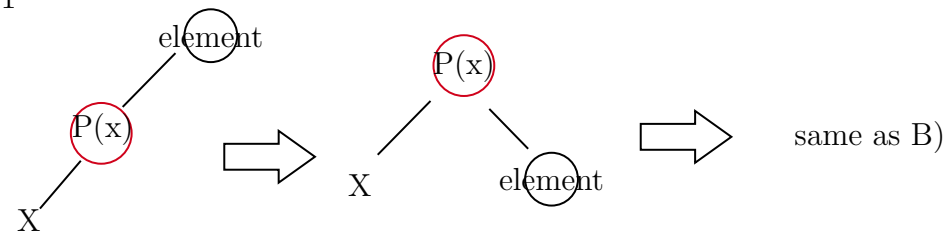


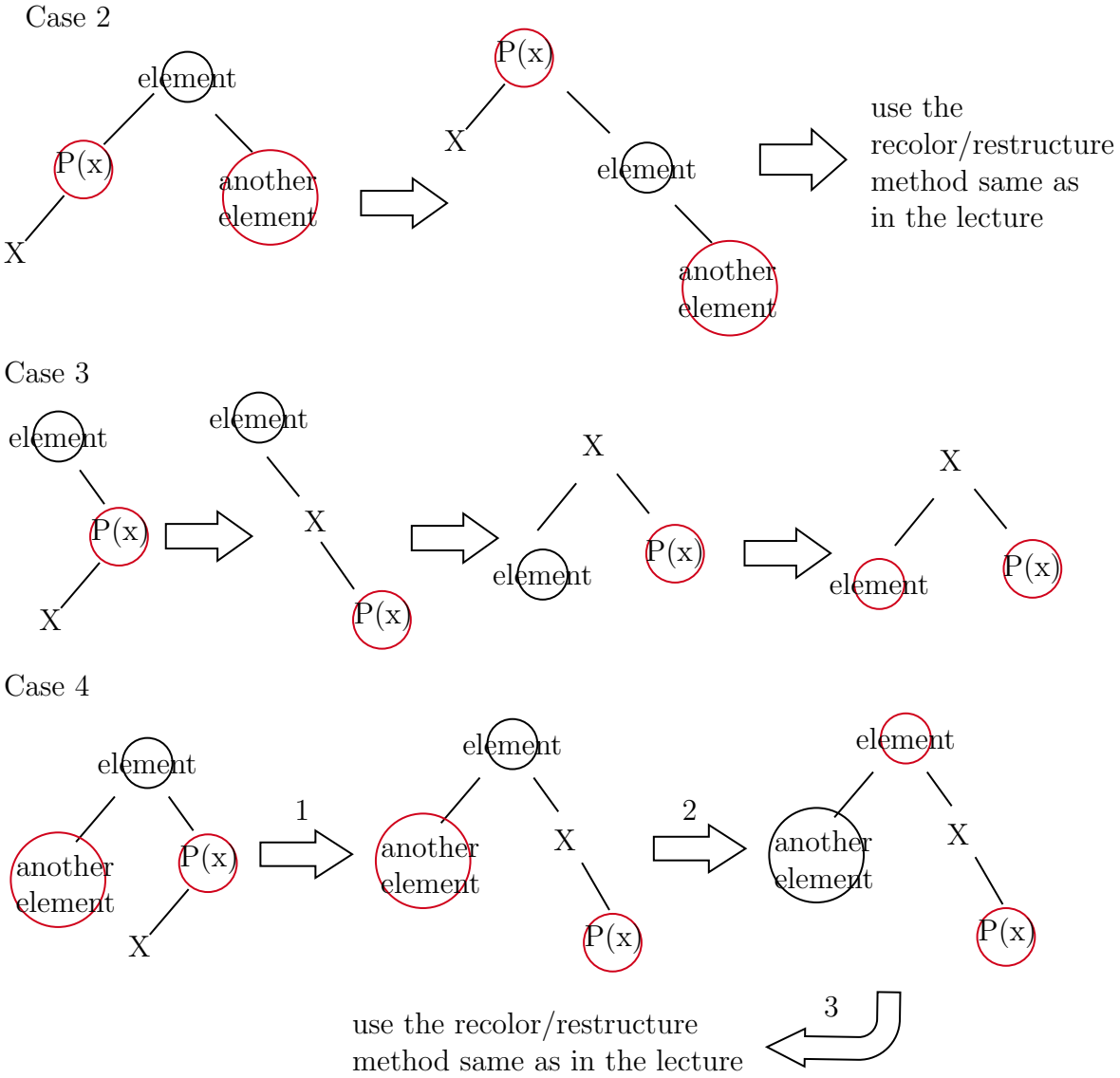

- If P(x) has red parent, do recolor/restructure same as in the lecture
- C) This tree violates the rule: each subtree has the same black height, so recolor/restructure is unnecessary:



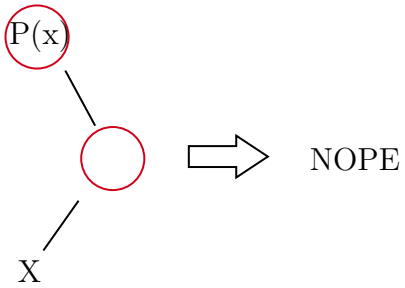

- D) It depends:

Case 1






- E) This tree violates the rule: red node must have black parent, so recolor/restructure is unnecessary:




- This tree violates the rule: each subtree has the same black height, so recolor/restructure is unnecessary: