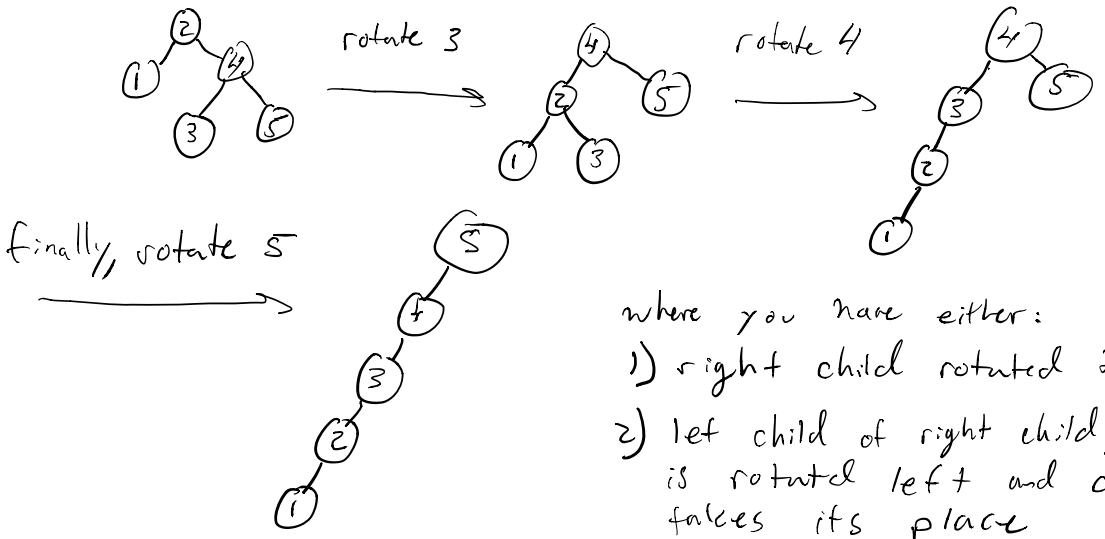
- Each rotation takes O(1) time. Any binary tree can become a left or right-child-only tree. This is done at the worst-case n-1 times by rotating all nodes not already on the left most path.

ex. 1



where you have either:
1) right child rotated to the left
2) left child of right child, parent is rotated left and child takes its place
3) left child of left child which is already left most path, or turns into case 2.

- In the worst case (n-1 rotations) every node needs to be rotated except for one node which is already in the left most position. A 3 node tree at a right-child-only setup will only need a max of 2 rotations to transform it into a left most path. Thus: $3-1=\underline{2}$

- Comparing to two trees would require transforming T_1 into a worst-case left-only tree has n-1

rotations, then checking the operations needed to equal T_2 gives another worst-case of $n-1$ to rotate all the way back. Thus summed we have:

$$(n-1) + (n-1) = \boxed{2n-2 \leq 2n}$$