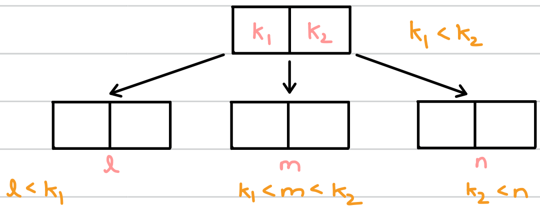
SEARCH TREES

2 - 3 TREES

- Multiway Search Tree (M Way Search Tree)
  - M → degree
- Degree 3 (2-3 trees are Multiway Search Trees with degree 3)
- B Trees (These are height Balanced Search Trees)
- Rules
  - All leaf nodes at same level
  - Every node must have $\lceil \frac{n}{2} \rceil$ children

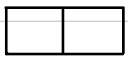
$$\left\lceil \frac{3}{2} \right\rceil = 2$$



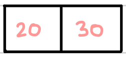
  - Cannot have duplicates

CREATION OF 2-3 TREE

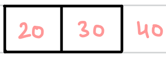
KEYS : 20 , 30 , 40 , 50 , 60 , 10 , 15 , 70 , 80 , 90

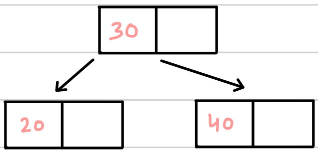


20 , 30

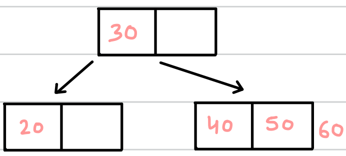


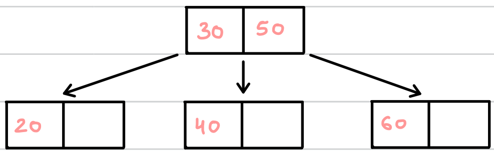
40



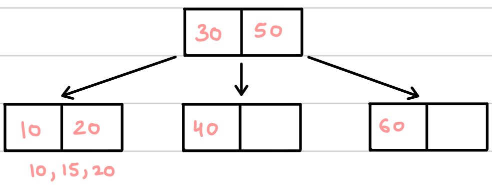


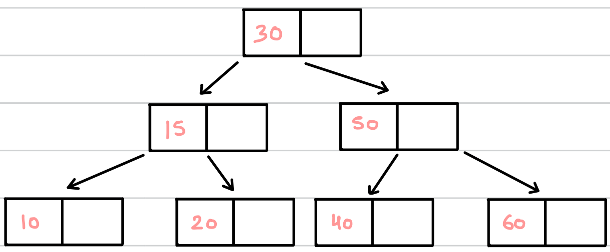
50 , 60



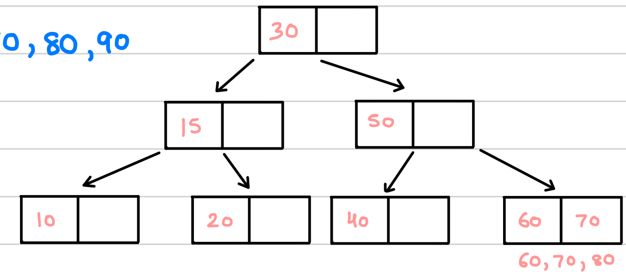


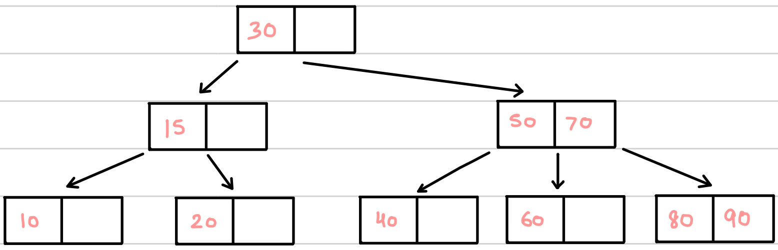
10 , 15





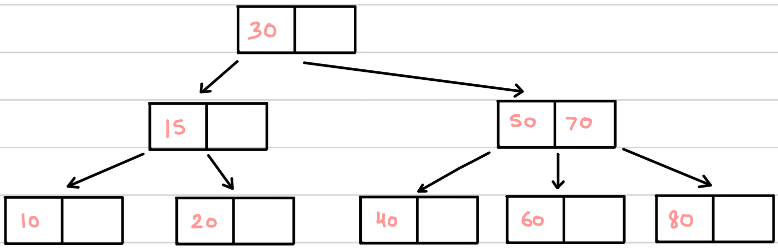
70, 80, 90



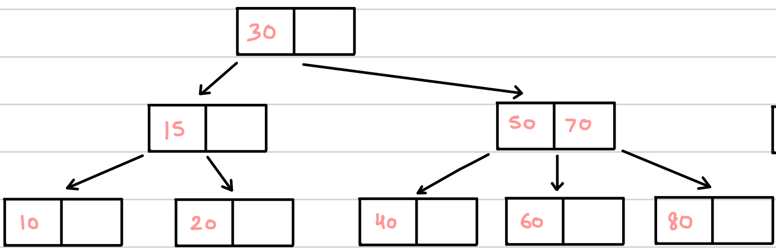


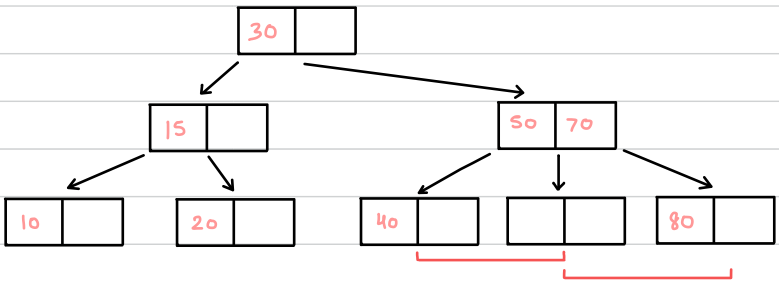
DELETING FROM 2-3 TREE

CASE 1 : Simply Delete
90

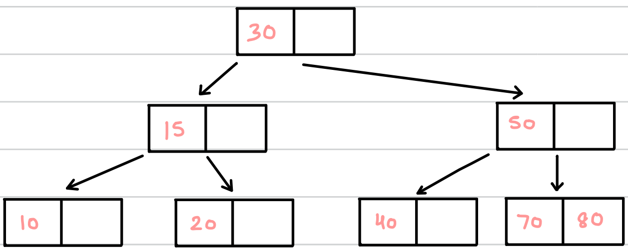


CASE 2 : Delete and merge
60



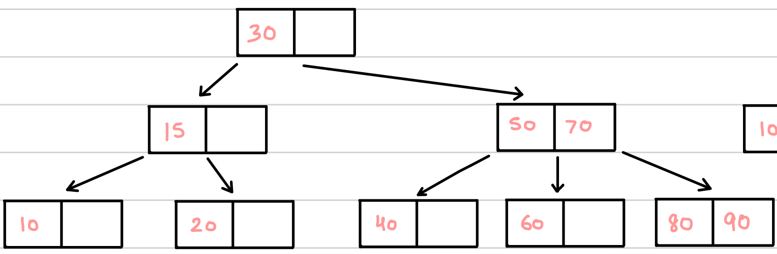


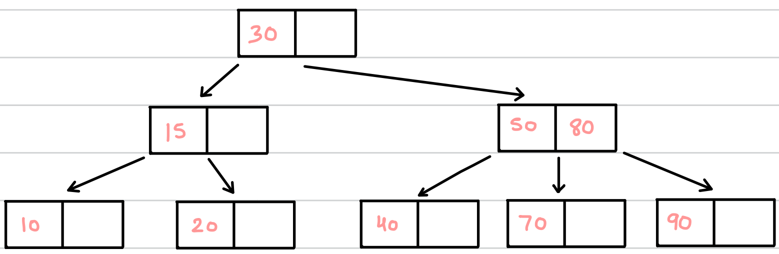
Join with either left sibling or right sibling



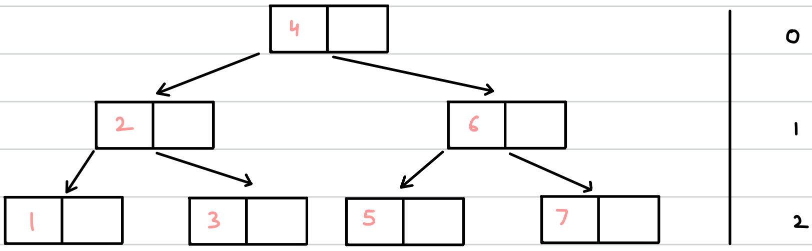
Merging with right sibling

CASE 3 : Borrow
60





ANALYSIS



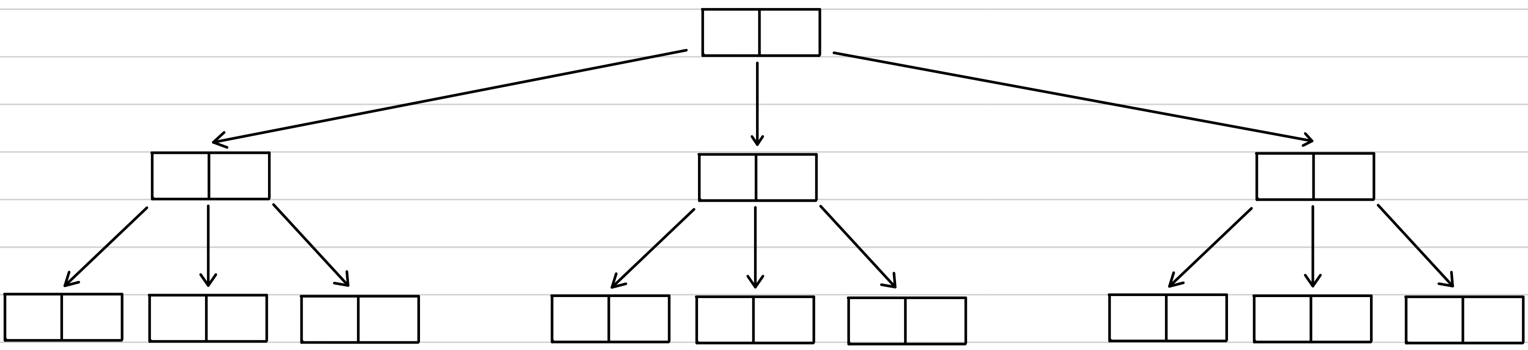
Tree with min nodes for given height

Minimum $n = 1 + 2 + 2^2 \ldots$

$= 2^{h+1} - 1$

Max $h = \log_2 (n-1) - 1$

$= O(\log_2 n)$



Max $n = 1 + 3 + 3^2 \ldots..$

$$= \frac{3^{h+1} - 1}{3 - 1}$$

Min $h = \log_3 [n(3-1) + 1] - 1$

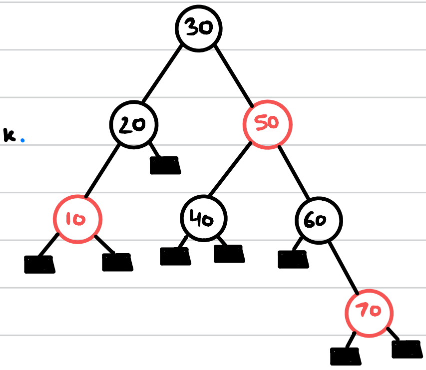
$O(\log_3 n)$

Minimum as well as maximum height is $\log n$

These trees are used for DBMS softwares because a node can have more than one value

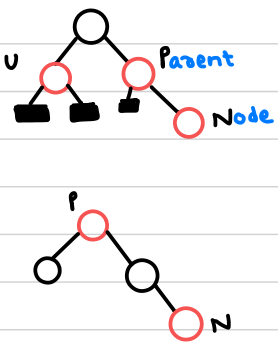
RED BLACK TREE

- It is a height balanced Binary Search Tree, similar to 2-3-4 tree.
- Every node is either Red or Black.
- Root of a Tree is Black
- NULL is also Black.
- Number of Blacks on paths from Root to leaf are same.
- No 2 consecutive Red, Parent and Children of red are Black.
- New inserted Node is Red.
- Height in $\log n \le h \le 2\log n$



CREATION OF RED BLACK TREE

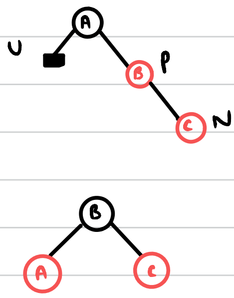
Uncle is Red (for Node)



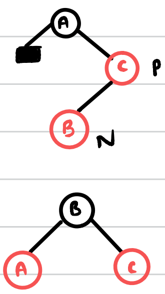
RECOLOURING

Uncle is Black

Zig-Zig (LL/RR)



Zig-Zag (RL/LR)



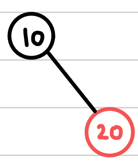
ROTATION

KEYS : 10, 20, 30, 50, 40, 60, 70, 80, 4, 8

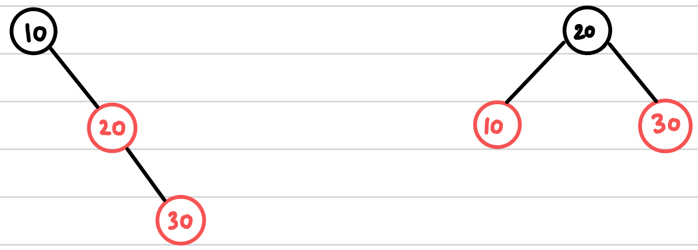
INSERT

10

20



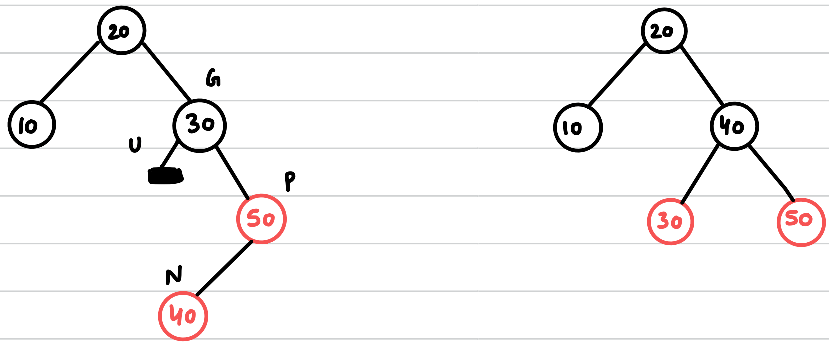
30



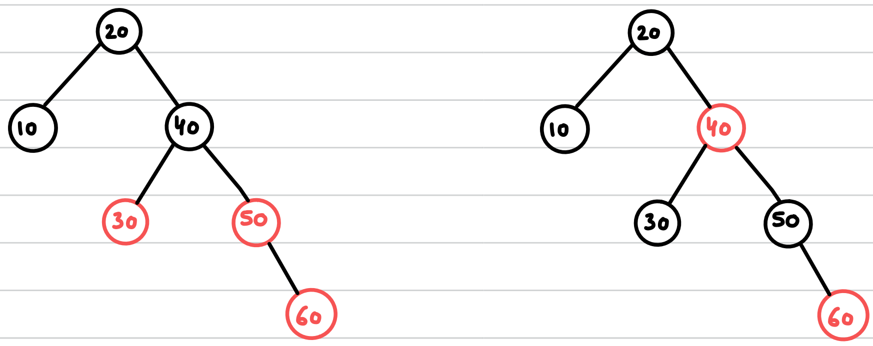
50

Root must be black

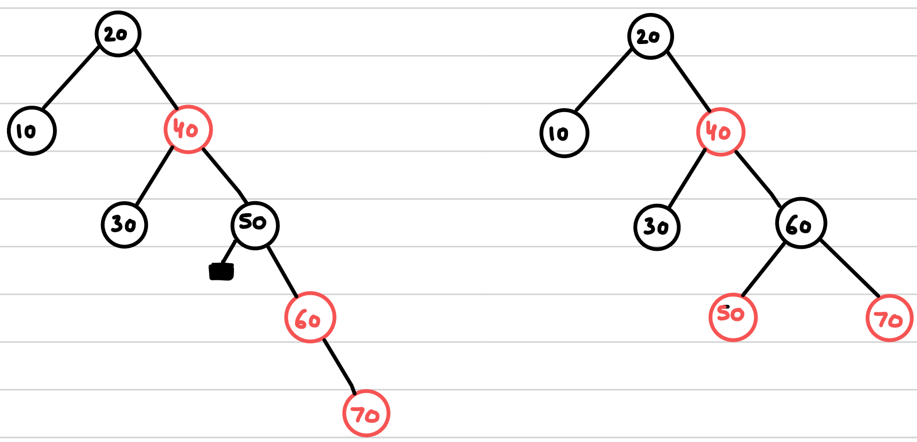
40



60

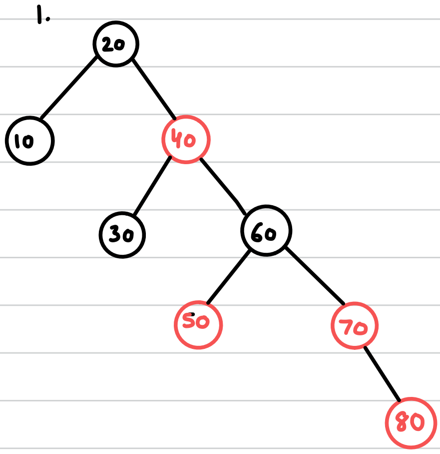


70

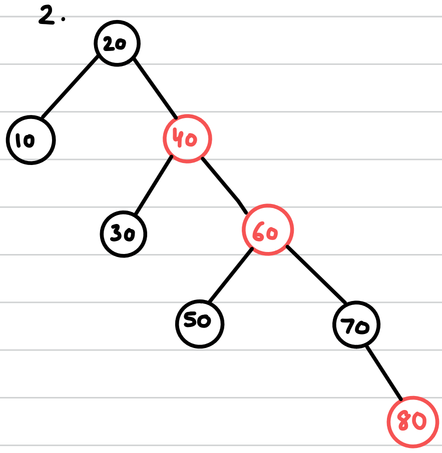


80

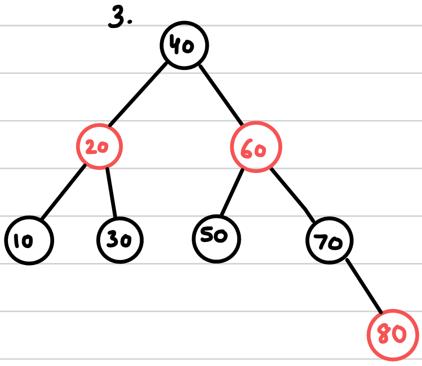
1.



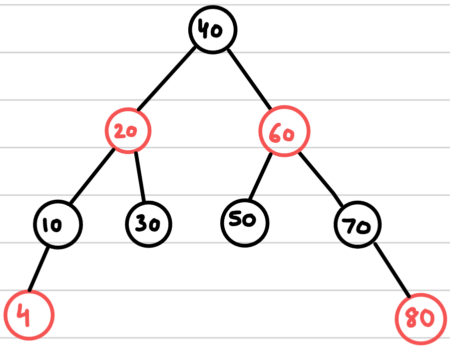
2.



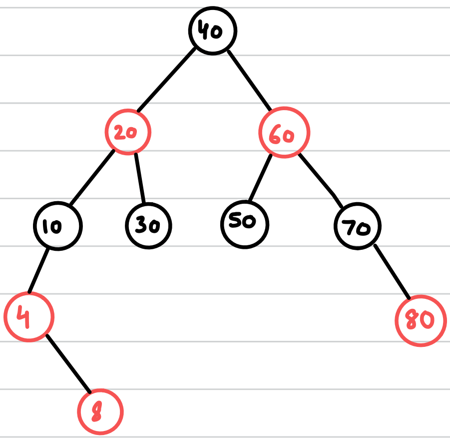
3.

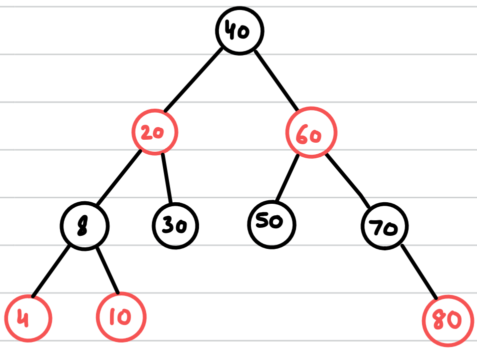


4



8

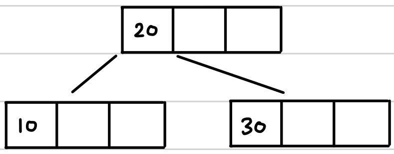


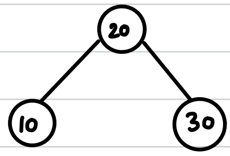


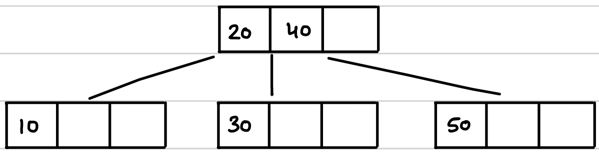
height only red / black = log n

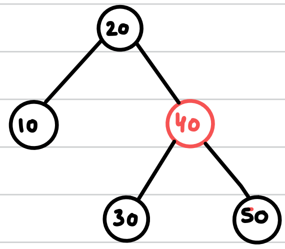
height red and black = 2 log n

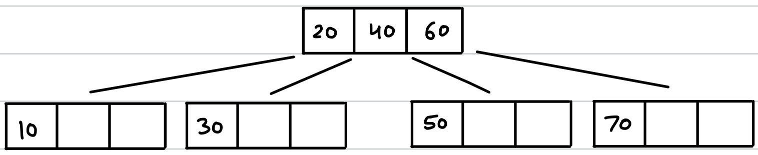
2-3-4 TREES VS RED BLACK TREES

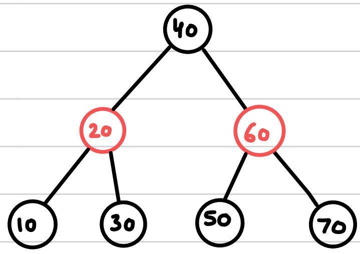










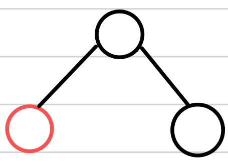


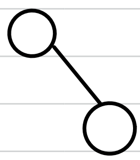
RED BLACK TREE DELETION CASES

CASE 1: Deleted Node is Red Node

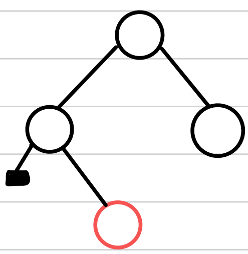
Before Deletion

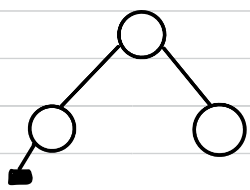
After Deletion

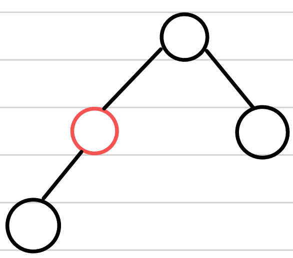


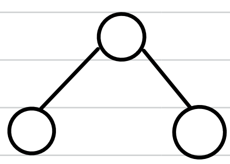


Simply delete as it is a leaf node and red





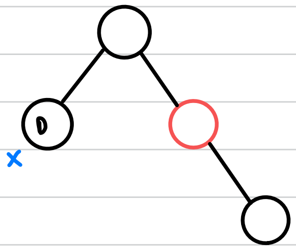




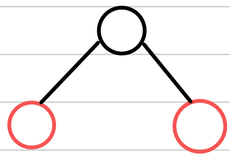
Simply delete because if red node is deleted, the path of black nodes remains unchanged

CASE 2 : Node is black and sibling is red

Before Deletion



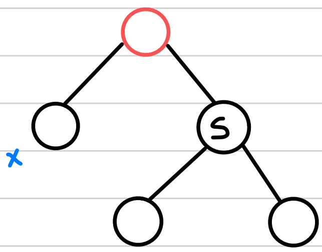
After Deletion



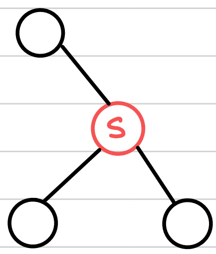
Perform rotation

CASE 3 : Node is black and sibling is also black

Before Deletion

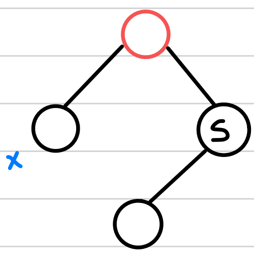


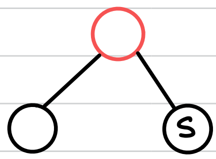
After Deletion



Change sibling to red and parent to black

Recolour





Perform rotation