Deletion

Red Black Tree – T

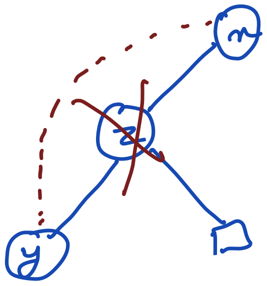
Delete (T, z) → delete z from T

Deletion in Binary Search Tree

Case 1: z – leaf

just delete z.

Case 2:- z has only one child

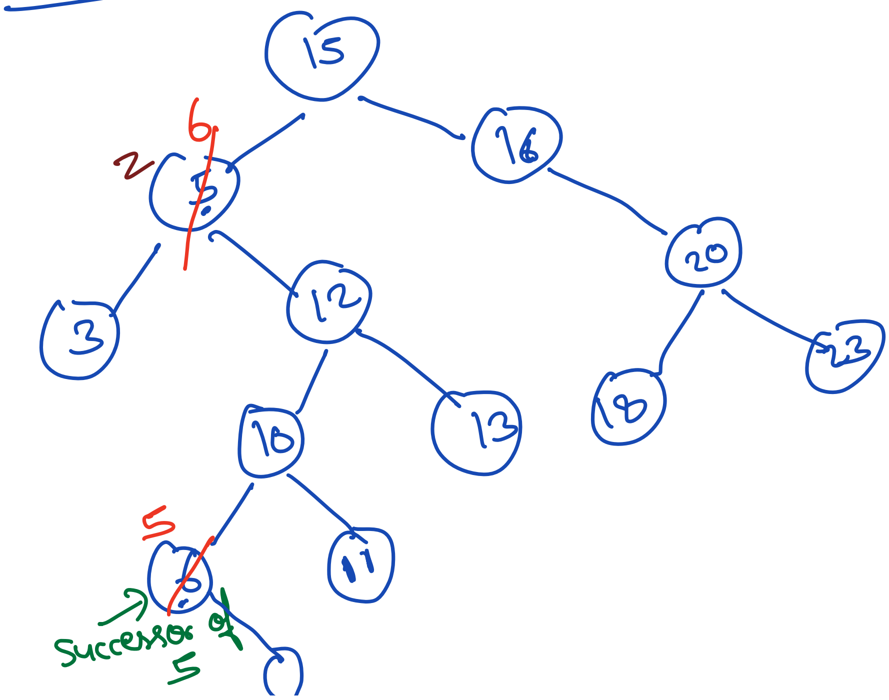


delete z

x is the parent of y.

y is left child of x.

Case 3:- z has two children.



replace key of z with its successor.

Reduced to Case 1 or Case 2

5 10 12

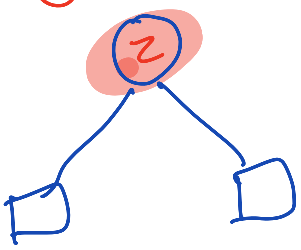
7

~~5~~ 6 - - - - .

Deletion in Red Black Tree

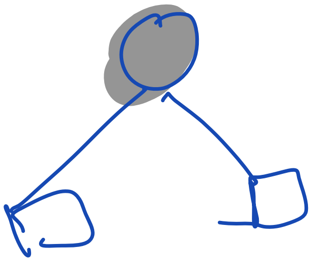
(1) Delete as we do in BST

Case 1:



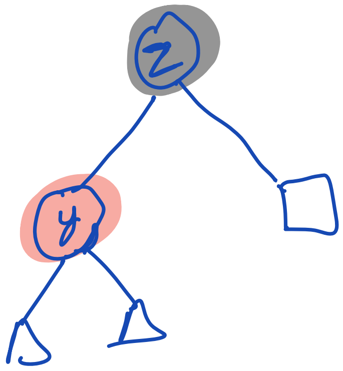
delete z.
it is still a
RB- tree

Case 2:-



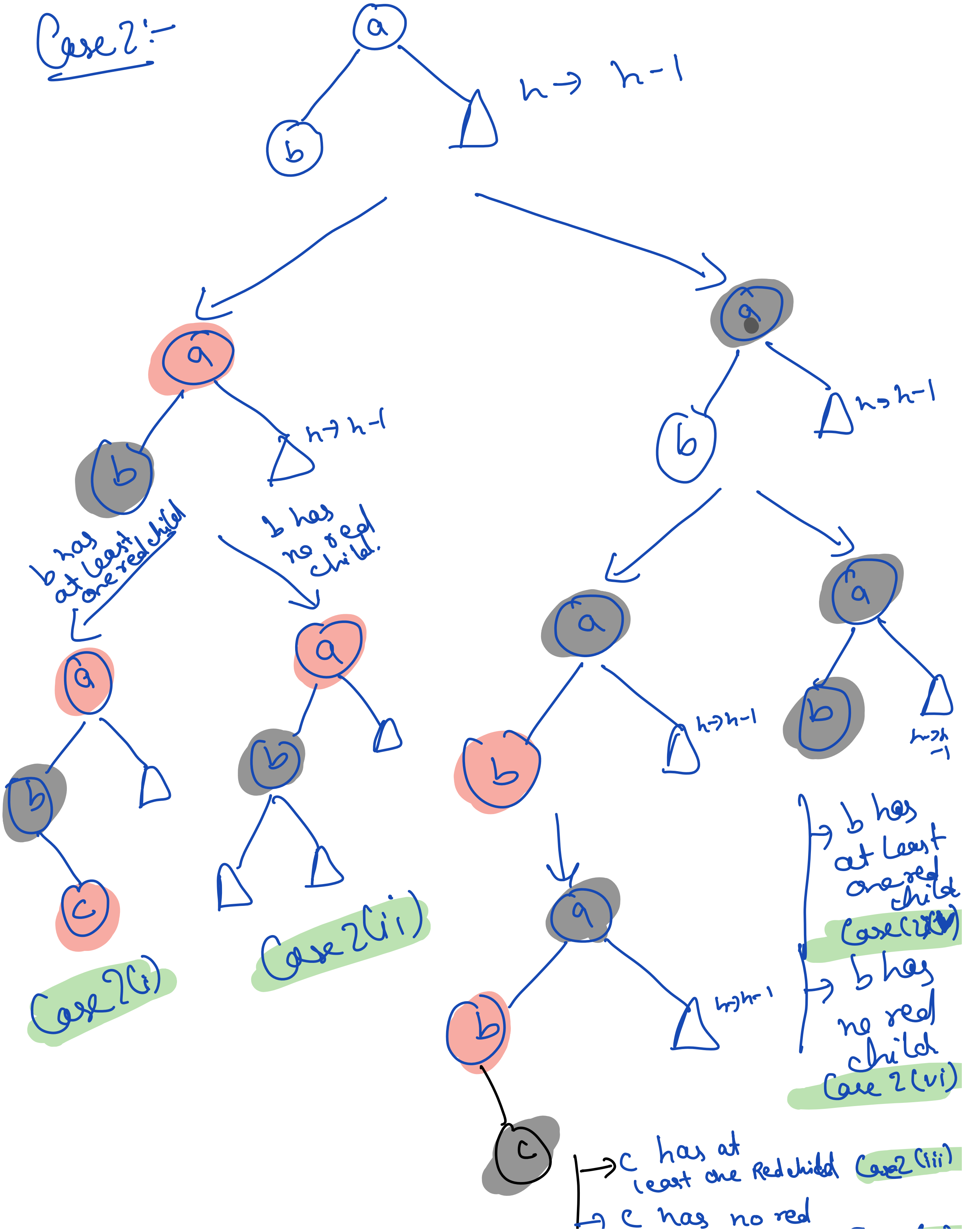
delete z.
black height
will decrease.

Case 3:-



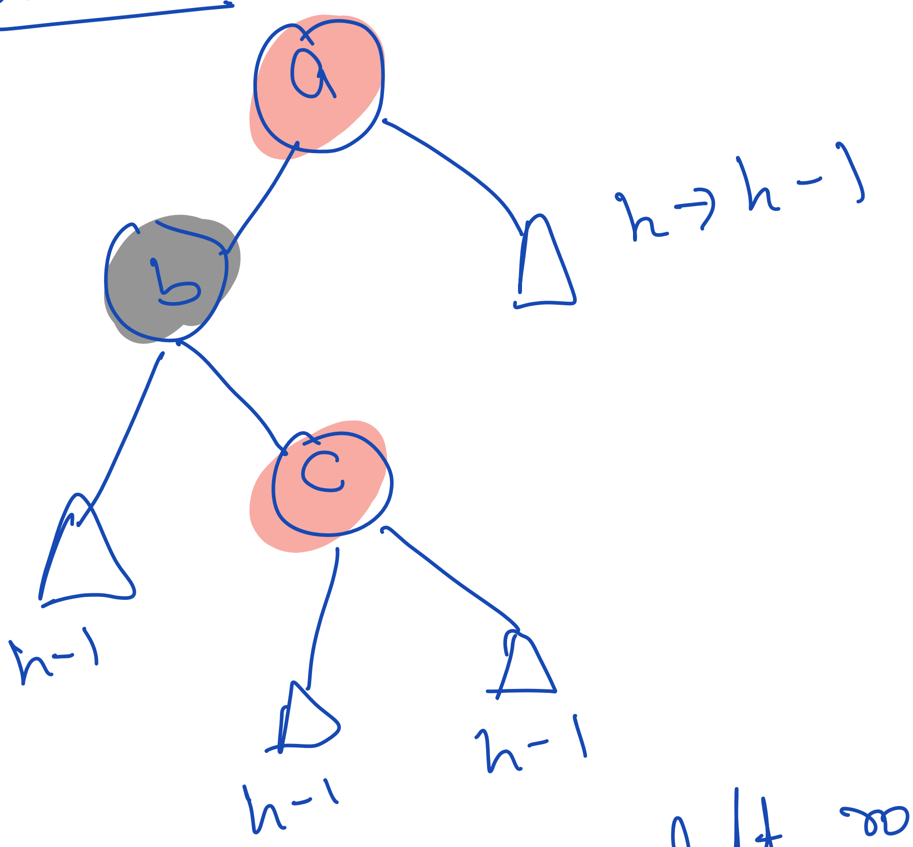
⇓
change the
color of y.

Case 2:-

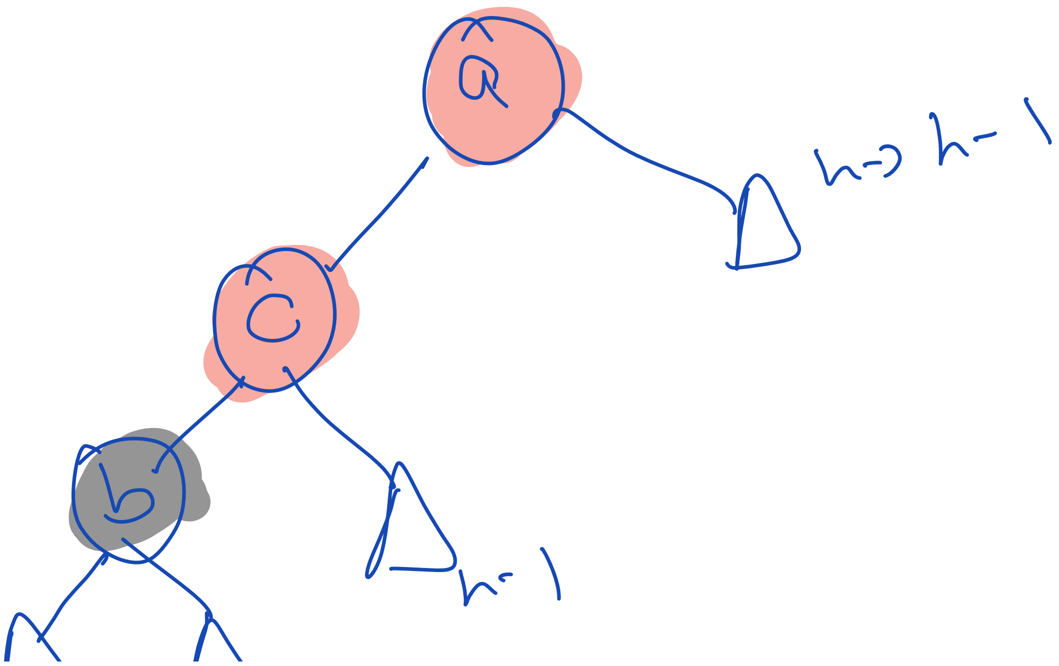

a
b
h → h-1
a
b
h→h-1
b has at least one red child
b has no red child.
a
b
c
Case 2(i)
a
b
Case 2(ii)
a
b
h→h-1
a
b
h→h-1
a
b
h→h-1
b has at least one red child
Case (2(v))
b has no red child
Case 2(vi)
a
b
h→h-1
c
c has at least one Red child
Case2 (iii)
c has no red

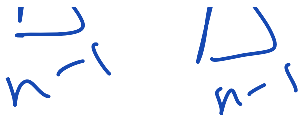
child Case 2(iv)

Case 2(i)

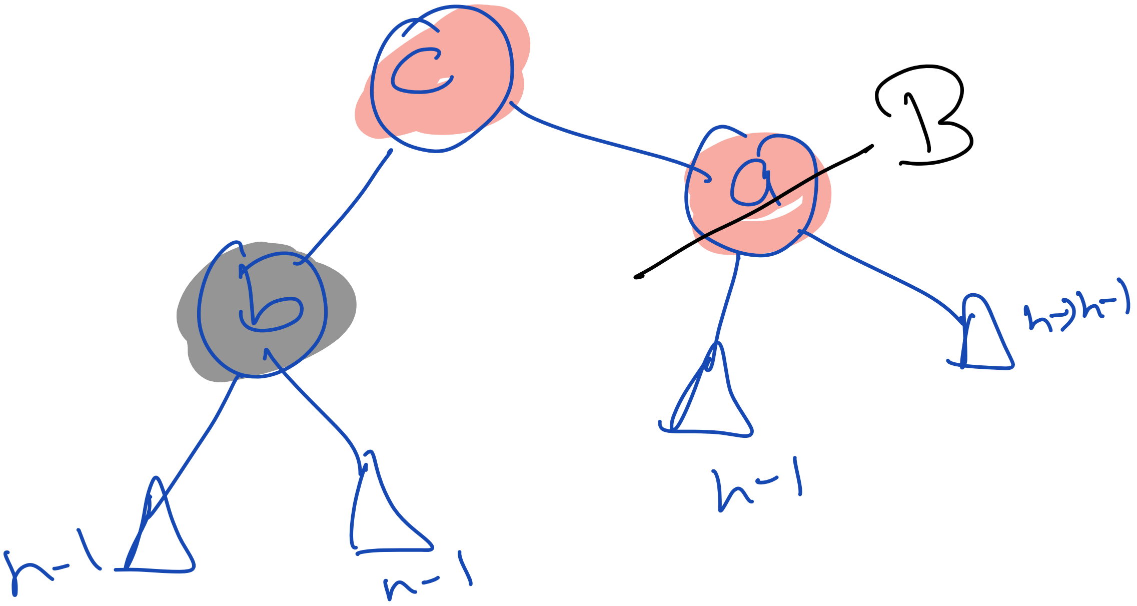


⇓ left_rotate (b)

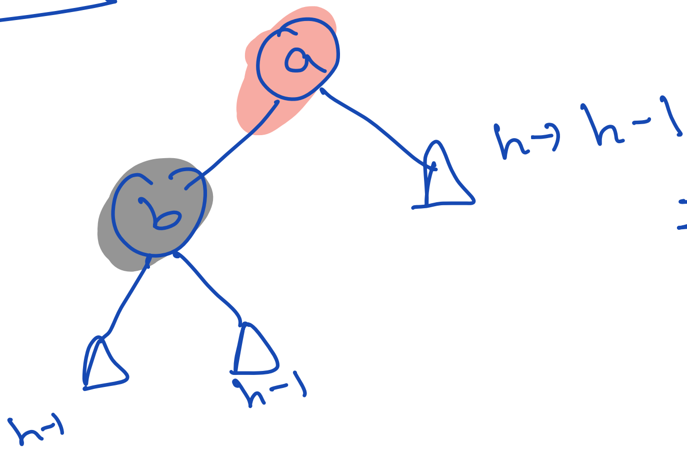


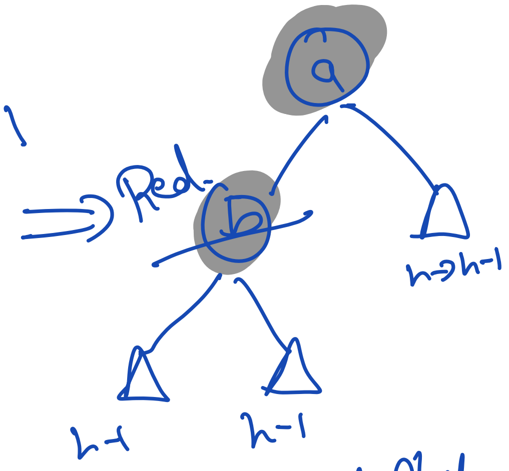


⇓ right_rotate(a)



Case 2(ii)





It is a Red Black Tree

→ Black depth is same for all external nodes

→ b does not have any red children due to assumption.