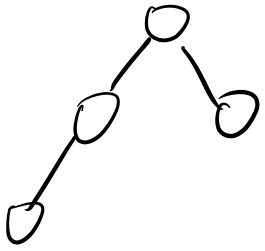
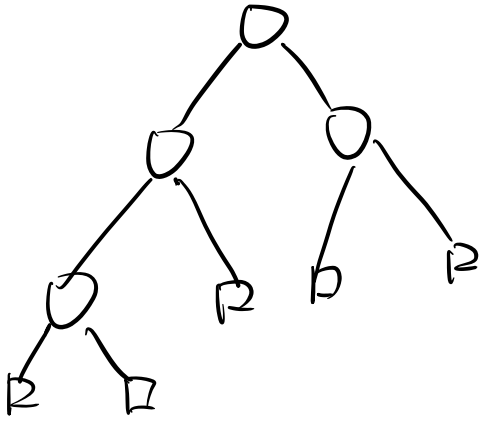
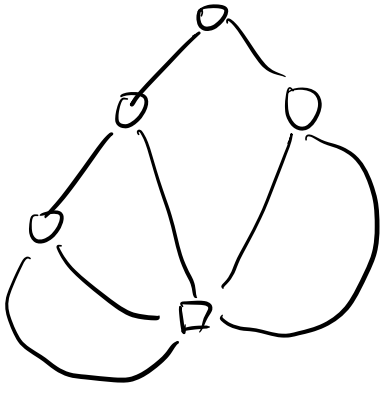
extended binary tree





□: NIL external node

○: internal



A red black tree is a BST whose extended version satisfies the following properties.

(1) every node is either red or black

(2) the root is black

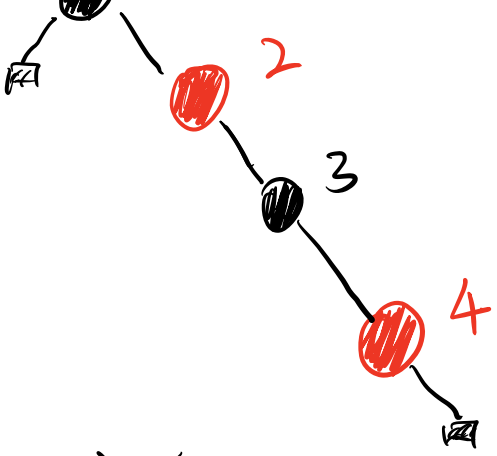
(3) every leaf (NIL) is black

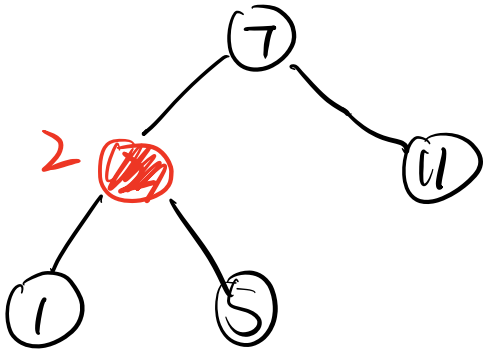
(4) the children of a red node must be black

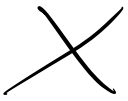
(5) for each node v, all descending path from v to leaves contain the same # black nodes.

(excluding v) black height of v : bh(v)

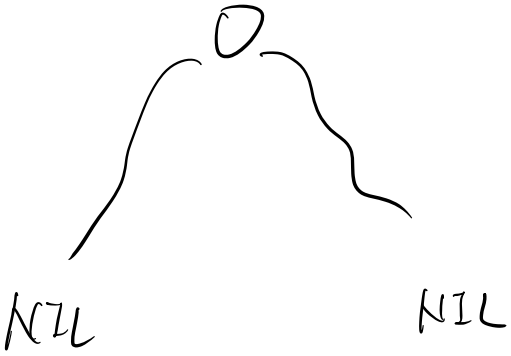
bh(T) = bh(root)







$bh(T) \geq h(T)/2$



Lemma

A red black tree T (extended version) with n internal nodes has height at most $2\lg(n+1)$

Proof:

for any $u \in T$.

T_u: the subtree rooted at u, size(T_u) = # internal nodes of T_u

Will show

$size(T_u) \geq 2^{bh(u)} - 1$ for any u.

Let u = root of T

$size(T) \geq 2^{bh(T)} - 1 \geq 2^{h(T)/2} - 1$

$\Downarrow$

$$h(T) \leq 2\lg(n+1)$$

By induction on height of T_u

Base case: $h(T_u) = 0$

$size(T_u) = 0$

$bh(u) = 0$

$size(T_u) \geq 2^{bh(u)} - 1$

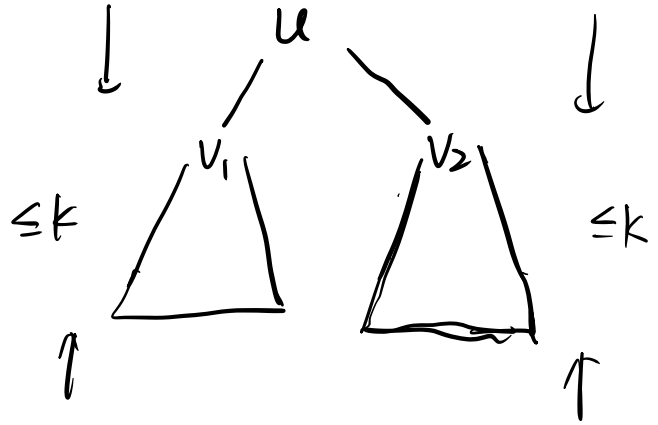
Inductive hypothesis:

Assume that all T_u with height $\leq k$,

$$size(T_u) \geq 2^{bh(u)} - 1$$

Inductive step:

When height of $T_u = k+1$



$bh(v_1) \geq bh(u) - 1$

$bh(v_2) \geq bh(u) - 1$

$$size(T_u) = 1 + size(T_{v_1}) + size(T_{v_2})$$

$$\geq 1 + 2^{bh(u)} - 1 + 2^{bh(u)} - 1$$

$$= 2^{bh(v_1)} + 2^{bh(v_2)} - 1$$

$$\geq 2^{bh(u)-1} + 2^{bh(u)-1} - 1$$

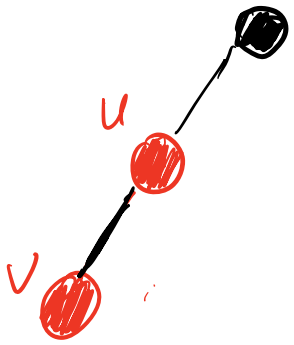
$= 2^{bh(u)} - 1$

Corollary

find key $O(\lg n)$

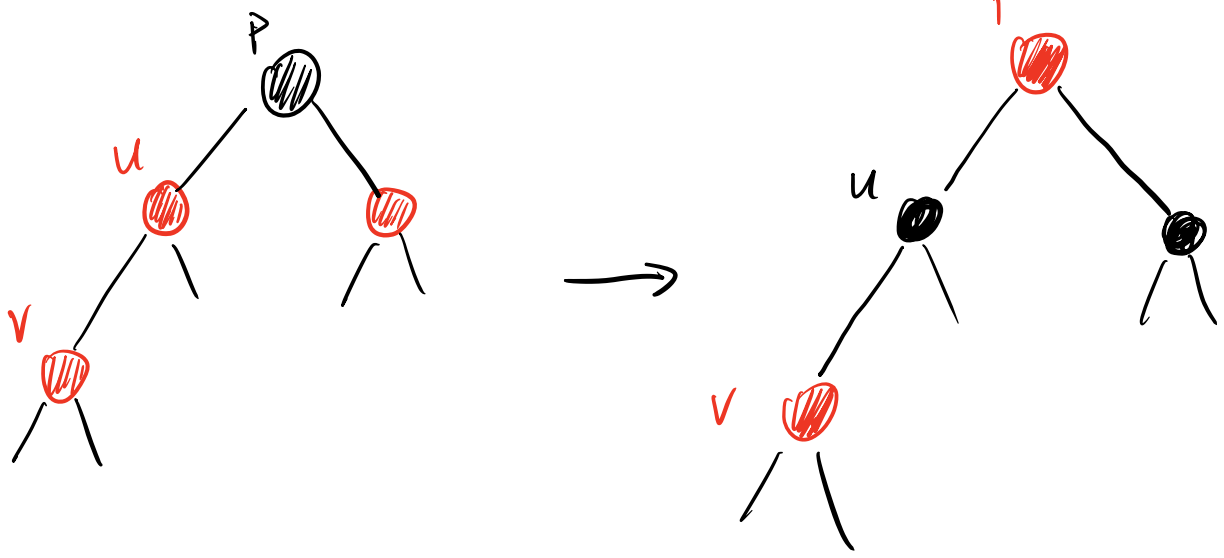
Insertion

as in BST, and mark as red



u is a left child | right child

case 1: sibling of u is red



If parent of p is black.

done

If p is ~~root~~ the root,

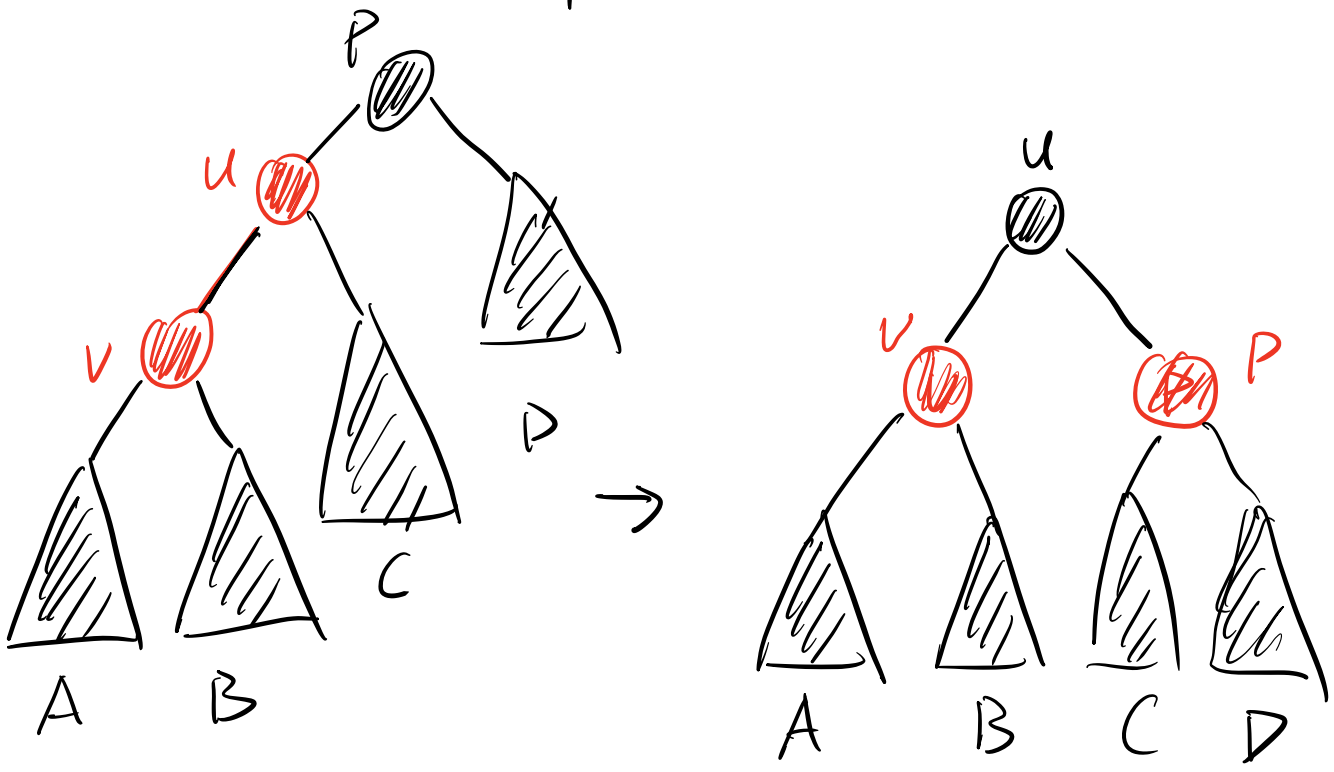
label p as black, done.

If parent of p is red

the violation is propagated upwards

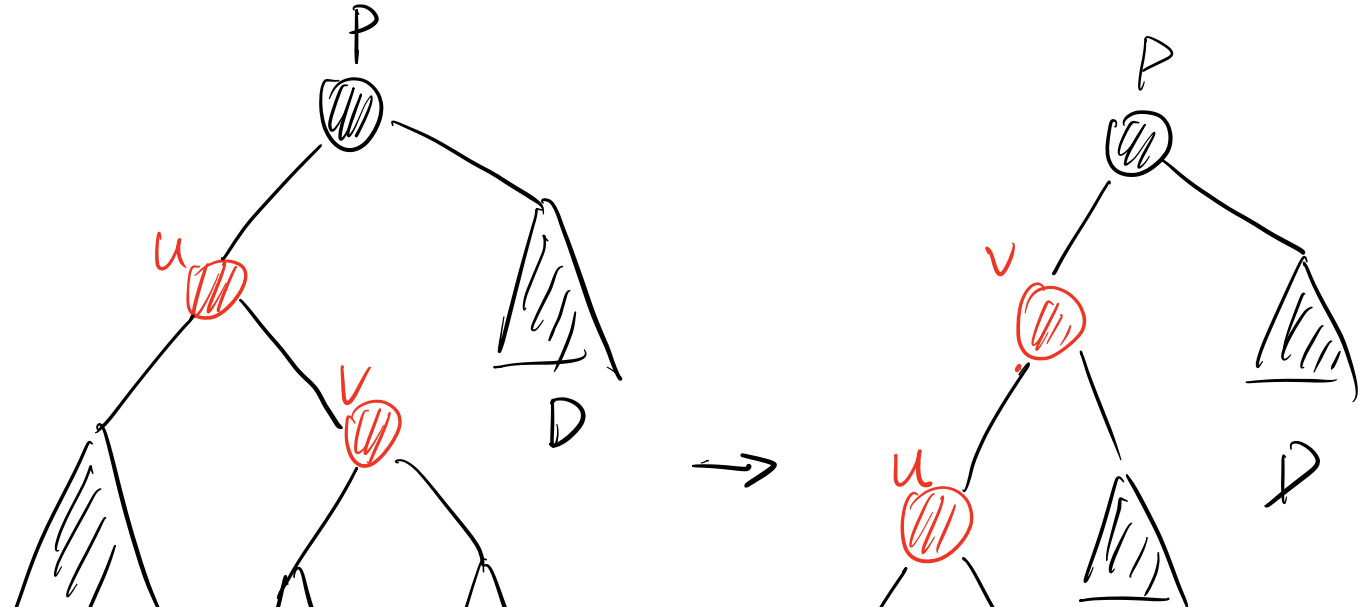
case 2. sibling of u is black

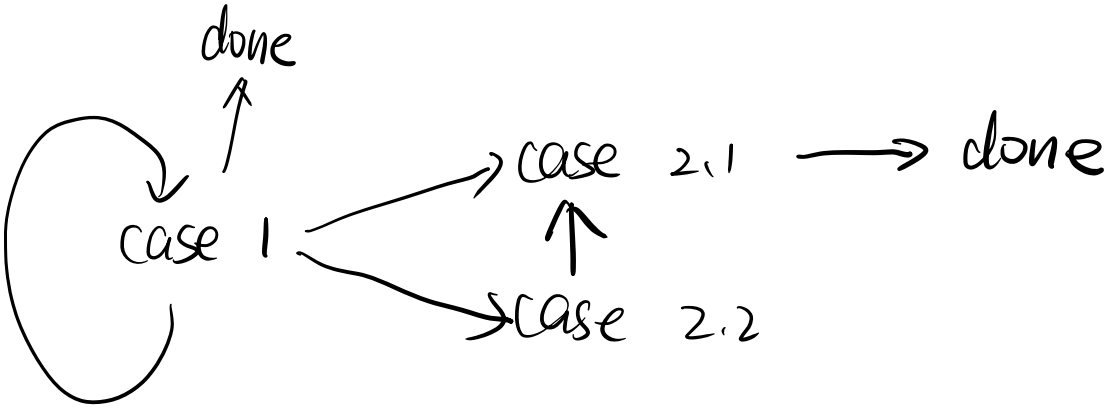
case 2.1 v is the left child of u



done.

case 2.2 v is the right child of u





Insertion: $O(1) \cdot O(\lg n) + O(1) = O(\lg n)$

rotations 2

Deletion

as in BST

deleted node has at most one child (excluding NIL)

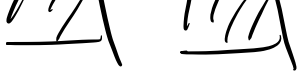
If the deleted node is red,

done

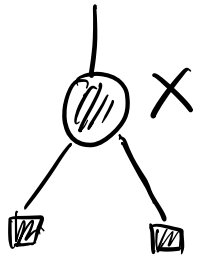
else (deleted node is black)

if it has a red child,

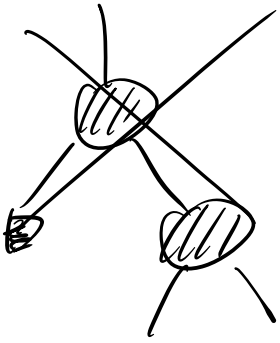
mark the child as black



else if its children are all black,

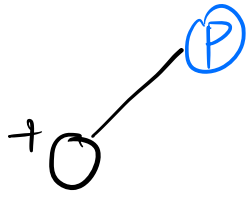


double black



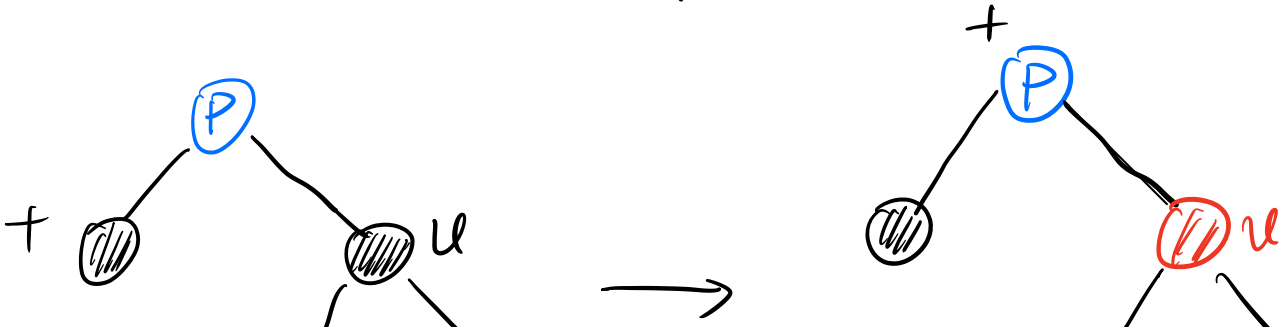
child

the double black node is left | right



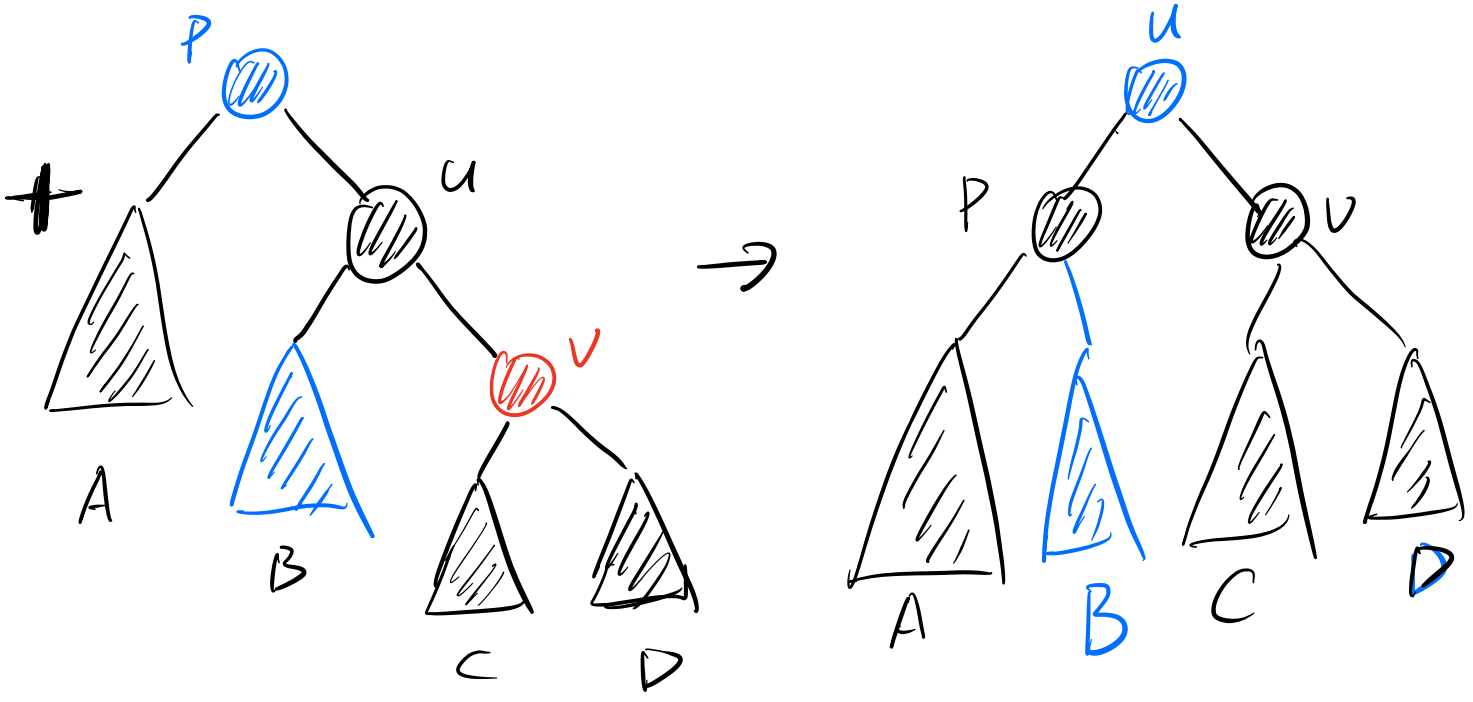
case 1 the sibling of double black is black

case 1.1 the children of u are all black



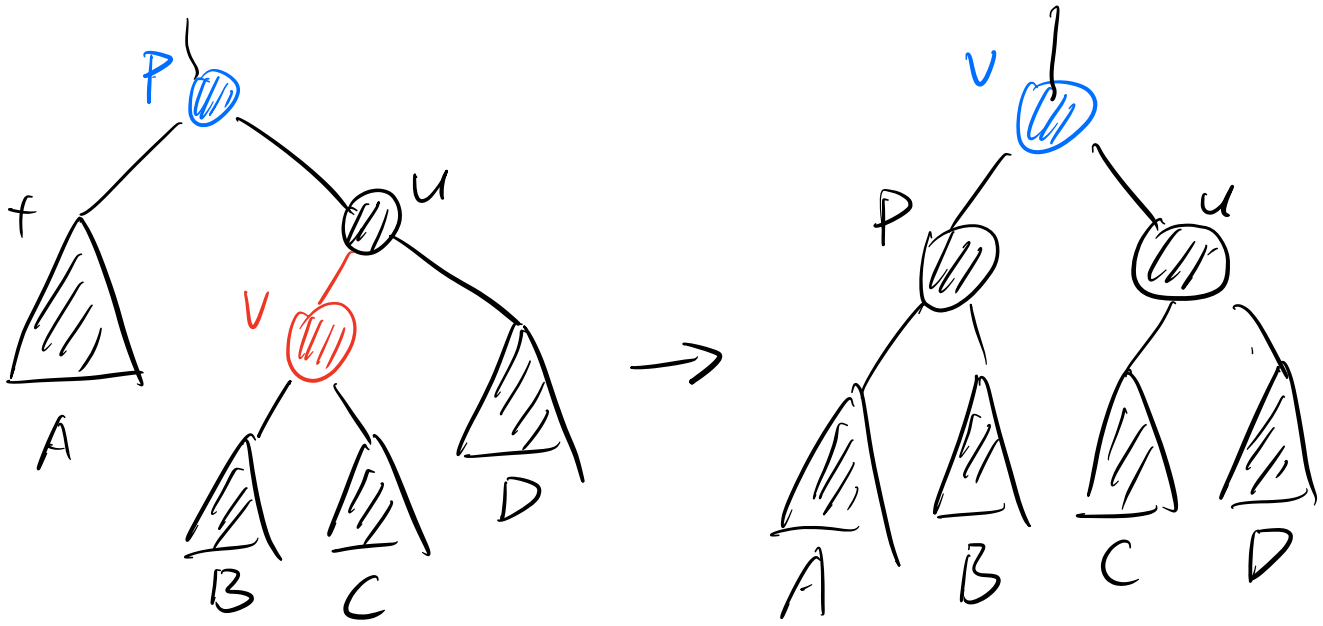
If p was red,
label p as black
If p was a root (black)
label p as black.
If p was ~~a~~ black (non-root)
p becomes double black
(propagated upwards)

case 1.2 the right child of u is red.



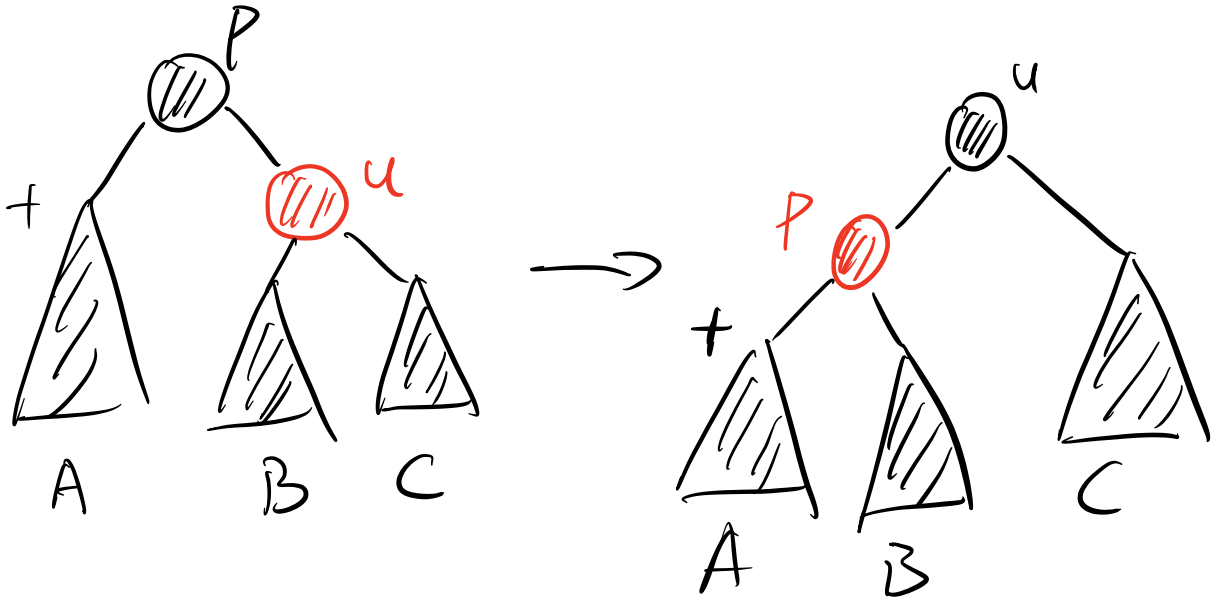
done.

case 1.3 the right child of u is black
the left child of u is red.

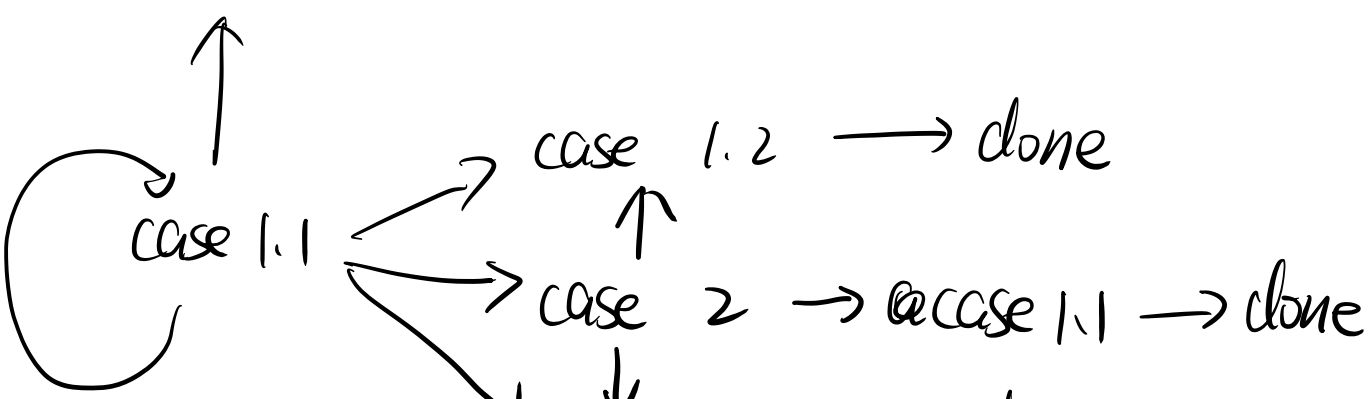


done.

case 2 the sibling of double black node is red



done ⇒ case 1



case 1.3 ⟶ done

Deletion: $O(\lg n)$

rotations ≤ 3

	AVL	RB	
findkey	$O(\lg n)$	$O(\lg n)$	AVL is better
Ins	$O(\lg n)$	$O(\lg n)$	$\simeq$
Del	$O(\lg n)$	$O(\lg n)$	RB is better