Data structure:

1. store data
2. support operation

	BST
find key	$O(h)$
insertion	$O(h)$
deletion	$O(h)$
⋮	

$\Rightarrow$ complete binary search tree $h \to \min.\ O(\log n)$

- Problem: after insertion or deletion $\rightarrow$ no longer complete

  return to complete BST : $O(n)$

- looser condition : Balanced binary tree.

  $|h_L - h_r| \le 1$ balance factor of

  BBST or AVL Tree.

AVL Tree:

Lemma:

A balanced binary tree with n nodes must have a height of $O(\log n)$

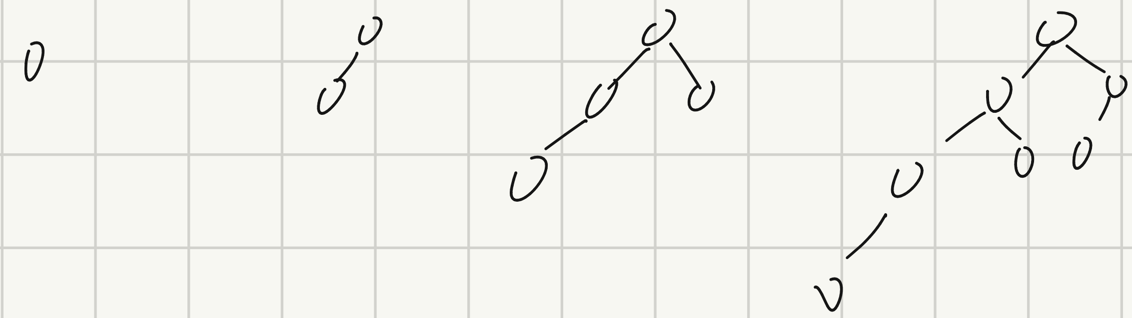
Proof:

any BBT of height h has at least c^h nodes

$n \ge c^h$ $h \le \log_c^n$

$n(h)$ = # nodes in the smallest BBT of height h.

$n(1)=1$ $\quad n(2)=2$ $\quad n(3)=4$ $\quad n(4)=7$ $= n(3)+n(2)+1 = 7.$

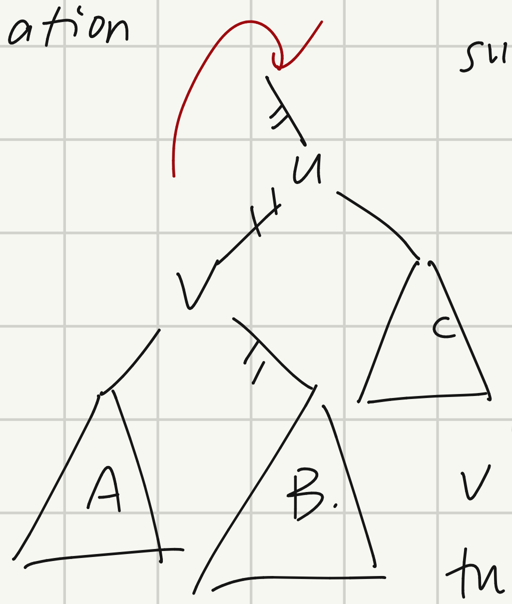


$n(h) = n(h-1) + n(h-2) + 1 \quad h \geq 2. \qquad n^h \approx \left(\frac{\sqrt{5}+1}{2}\right)^h.$

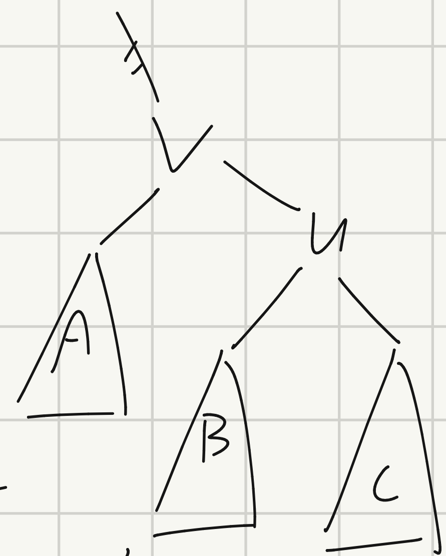
$\therefore \; h \leq \log_{1.618} n$

Rotation

subtree root u.



$\xrightarrow{\text{right rotation}}$



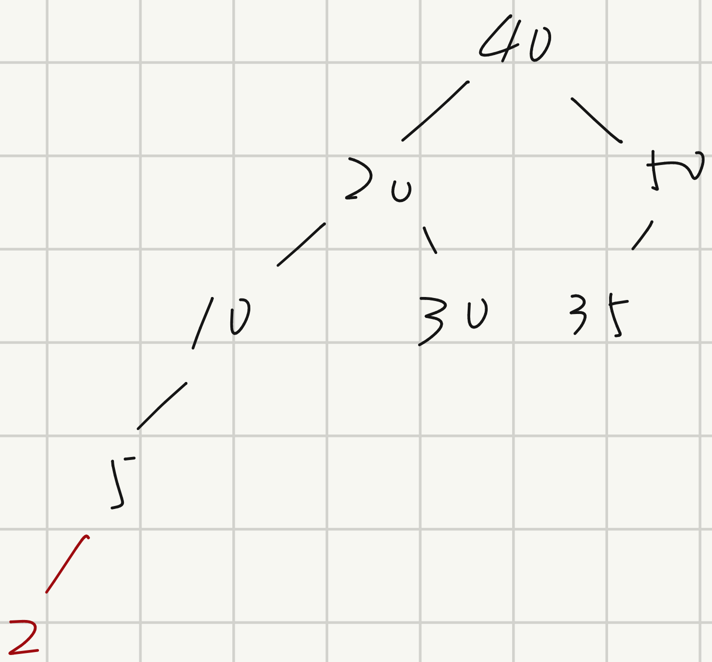
(described as lift v, it has at most two children, so change B's parent).

O(1) change 3 pointers

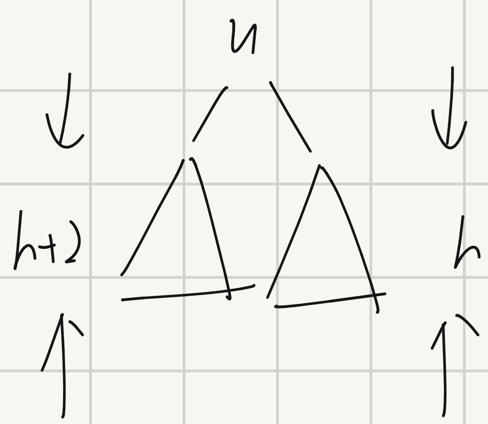
if BST $\Rightarrow A < v < B < u < c \xrightarrow[\text{rotation}]{\text{right}}$ remain BST property.

Insertion.

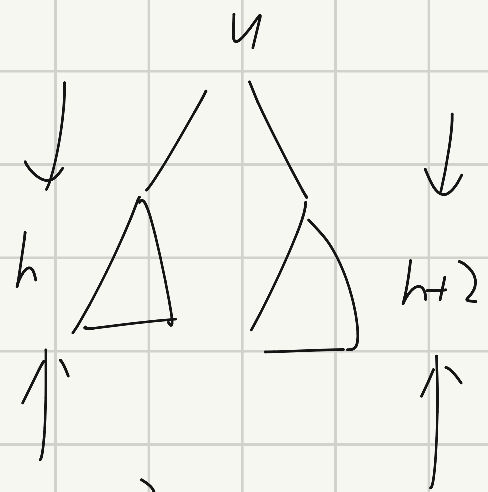
1. insert as BST (its ancestors failed balanced) ↳ $|h_l - h_r| \leq 2$.
2. restore the balance



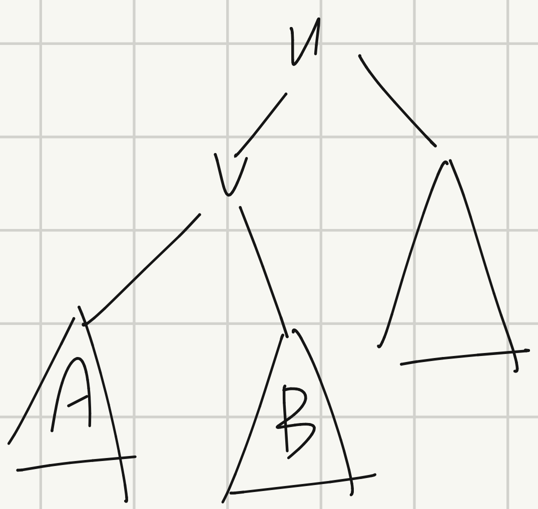
the lowest unbalanced node : u



L case



R case



$$\begin{cases} h_A = h+1 \\ h_B = h \end{cases}$$

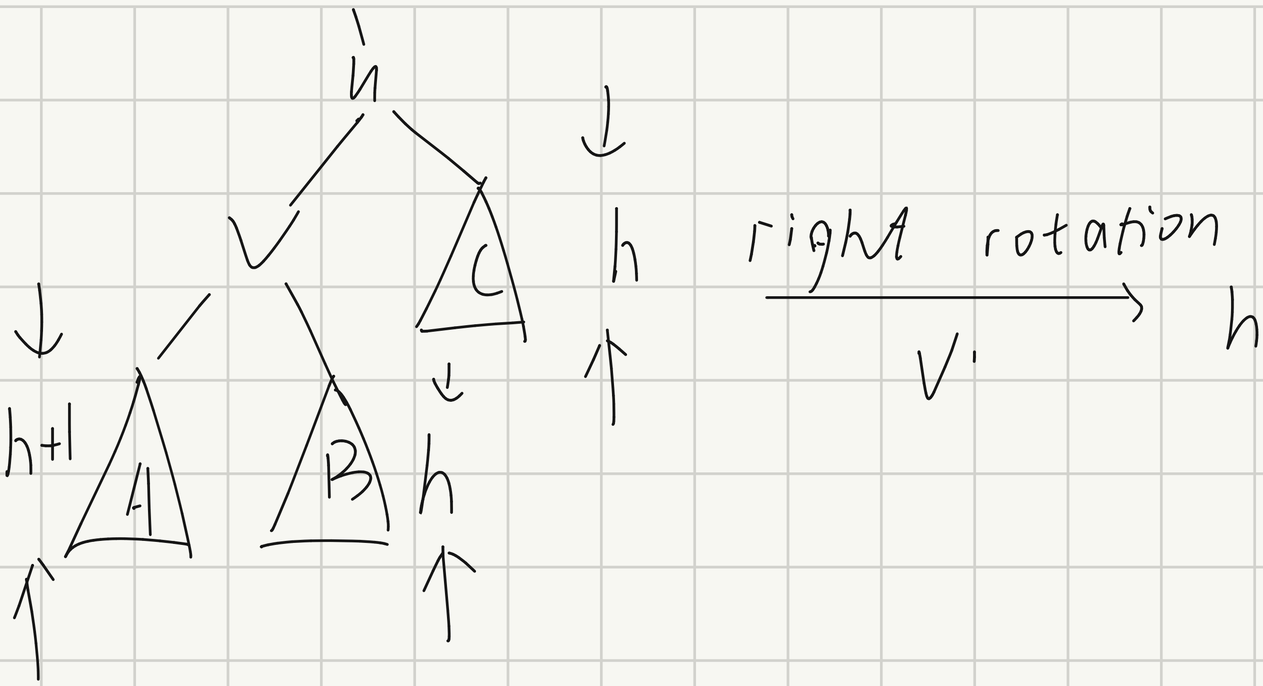
LL case

or

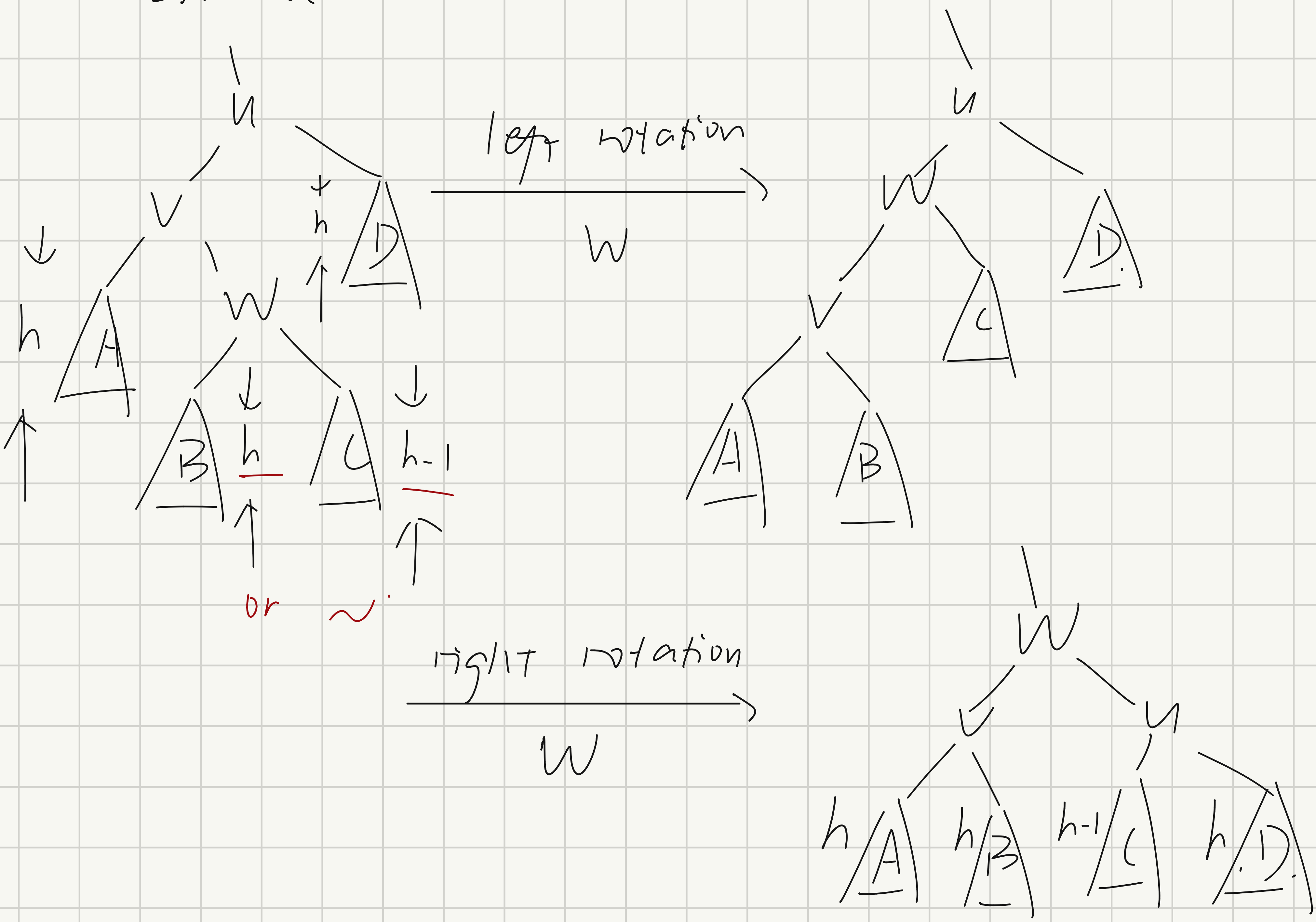
$$\begin{cases} h_A = h \\ h_B = h+1 \end{cases}$$

LR case

LL case



LR case



=> notice the rotation odd, at most 2 times

odd: nodes below the lowest imbalanced node start $\longrightarrow$ after insertion $\longrightarrow$ after rotation

$h+2$ $\qquad$ $h+3$ $\qquad$ $h+2$

Deletion:

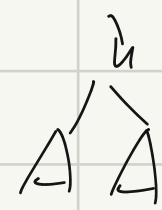
1. delete as in BST
2. restore the balance

Observation:

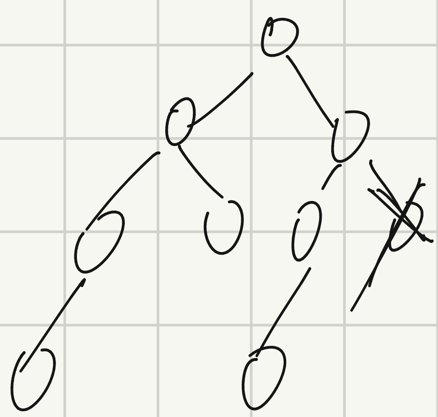
BST deletion in BBST is essentially removing a leaf.

delete (u)

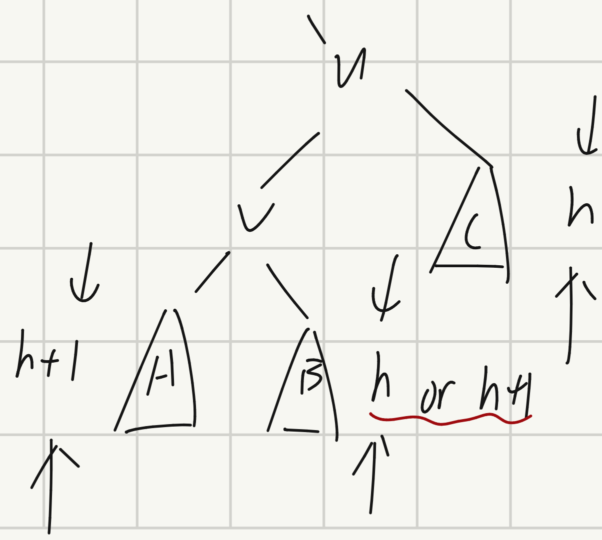
1. u is a leaf 2 u has only one child 3 u has two children



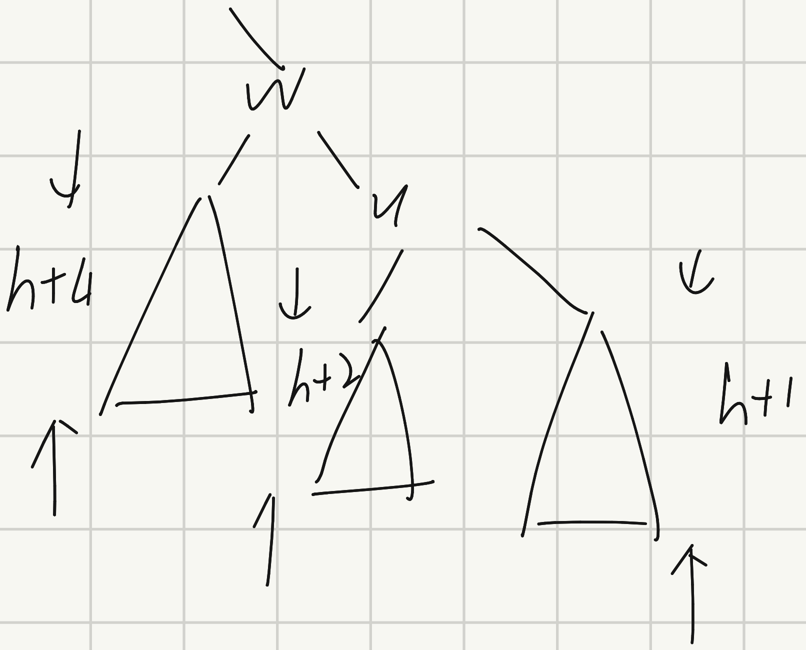
occur at most one unbalanced node.



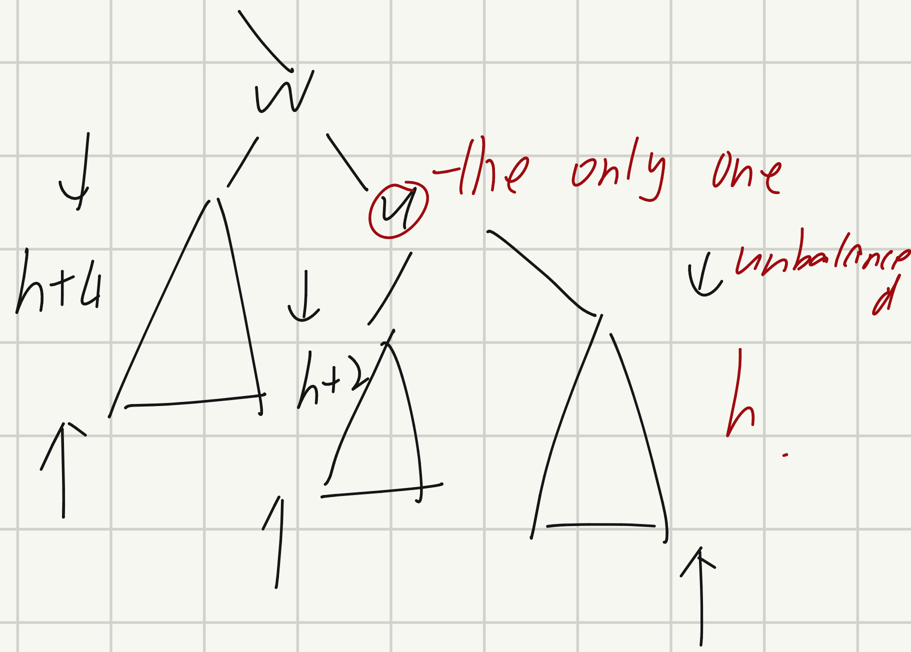
L case



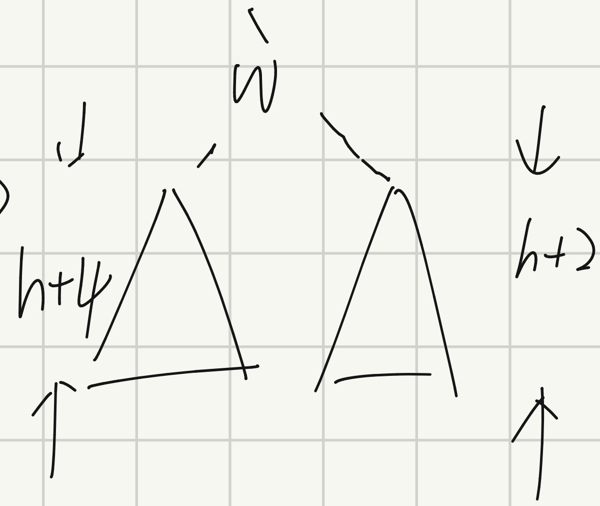
LL case



$\xrightarrow{\text{delete}}$



$\xrightarrow[\text{repair } u.]{\text{rotate } u.}$

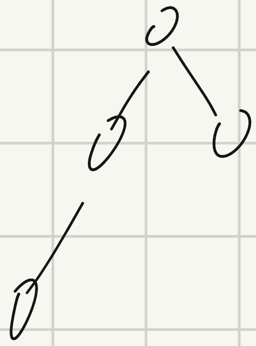


(after rotation, height ↓)

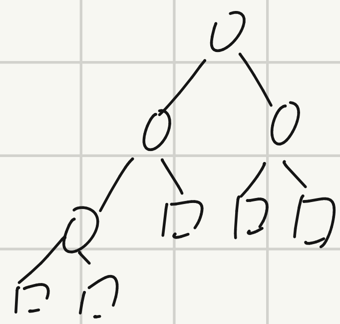
∴ unbalanced node goes upward.

$O(\log n)$ rotations

∴ deletion: $\underbrace{O(\log n)}_{\text{delete}} + \underbrace{O(1)}_{\text{each rotation}} \cdot \underbrace{O(\log n)}_{\text{rotation times}} = O(\log n)$



extended version



□ : NIL external nodes

○ : internal nodes

□ = ○ + 1

Paths have disparity lower than twice.

⇓

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

<1> node color: red or black.

<2> root is black.

<3> leaves (NIL) are black.

<4> children of red must be black ⇒ red << black.

<5> ※ for each node v, all descending paths from v to leaves contain the $\underbrace{\text{same number}}$ of black nodes

(excluding v)

→ black height of v: $bh(v)$

$bh(T) = bh(root)$

Tree.

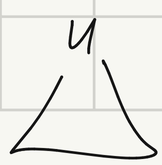
$<4> + <5> \Rightarrow h(T) \leq 2bh(T)$

<6> diff between l and r is less than twice, so rate ~ . . . . triple

Lemma: A RBT (in extended version) with n internal nodes has height of at most $2\log_2(n+1)$

Proof: for $u \in T$

Define: T_u:



size(T_u) = # internal nodes

Will show $size(T_u) \geqslant 2^{bh(u)} - 1$ for any u *

if true

$size(T) \geqslant 2^{bh(T)} - 1 \Rightarrow bh(T) \leq \log_2(n+1)$

$\Rightarrow h(T) \leq 2\log_2(n+1)$

Induction:

Base case $h(T_u) = 0$

u: $\square$ $size(T_u) = 0$ $bh(u) = 0$

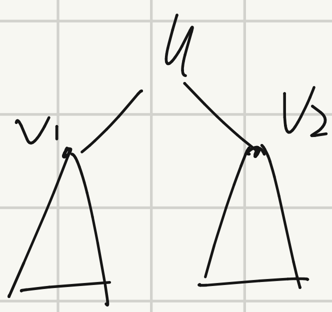
$size(T_u) \geqslant 2^{bh(u)} - 1$

Inductive hypothesis:

Assume that for all T_u with $h(T_u) \leq K$

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

Inductive step when $h(T_u) = K+1$



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

(hypothesis) $\geqslant 1 + 2^{bh(v_1)} - 1 + 2^{bh(v_2)} - 1$

$bh(v_i) = bh(u)$ u is red

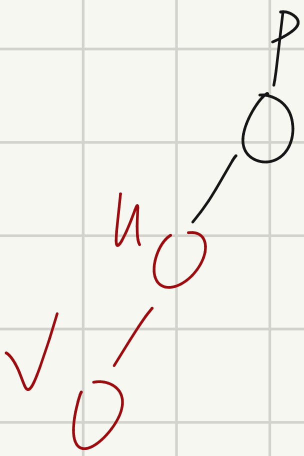
$bh(v_i) = bh(u) - 1$ u is black

$\geqslant 2 \cdot 2^{bh(u)-1} - 1$

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

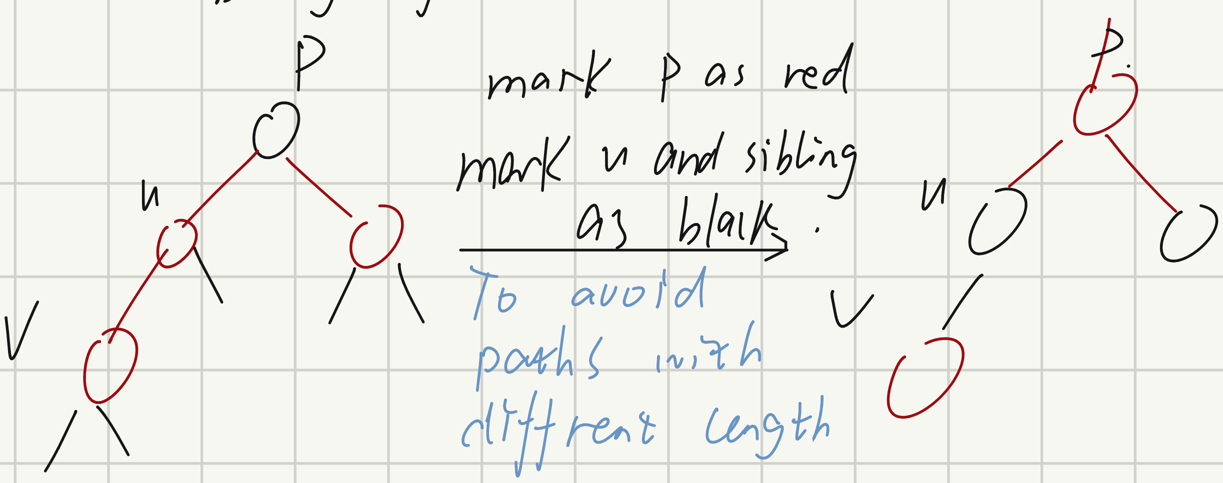
Insertion:

1. insert.
2. mark the new node red.



u is left child | u is right child

case 1: sibling of u is red.



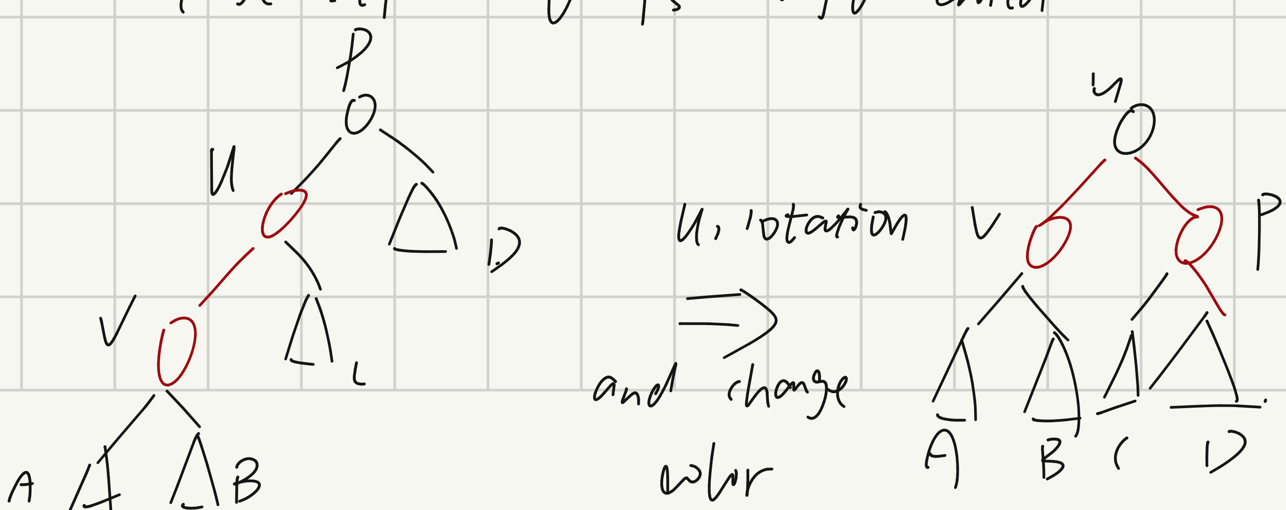
a) P is root, mark P as black. done.

b) parent of P is black, done.

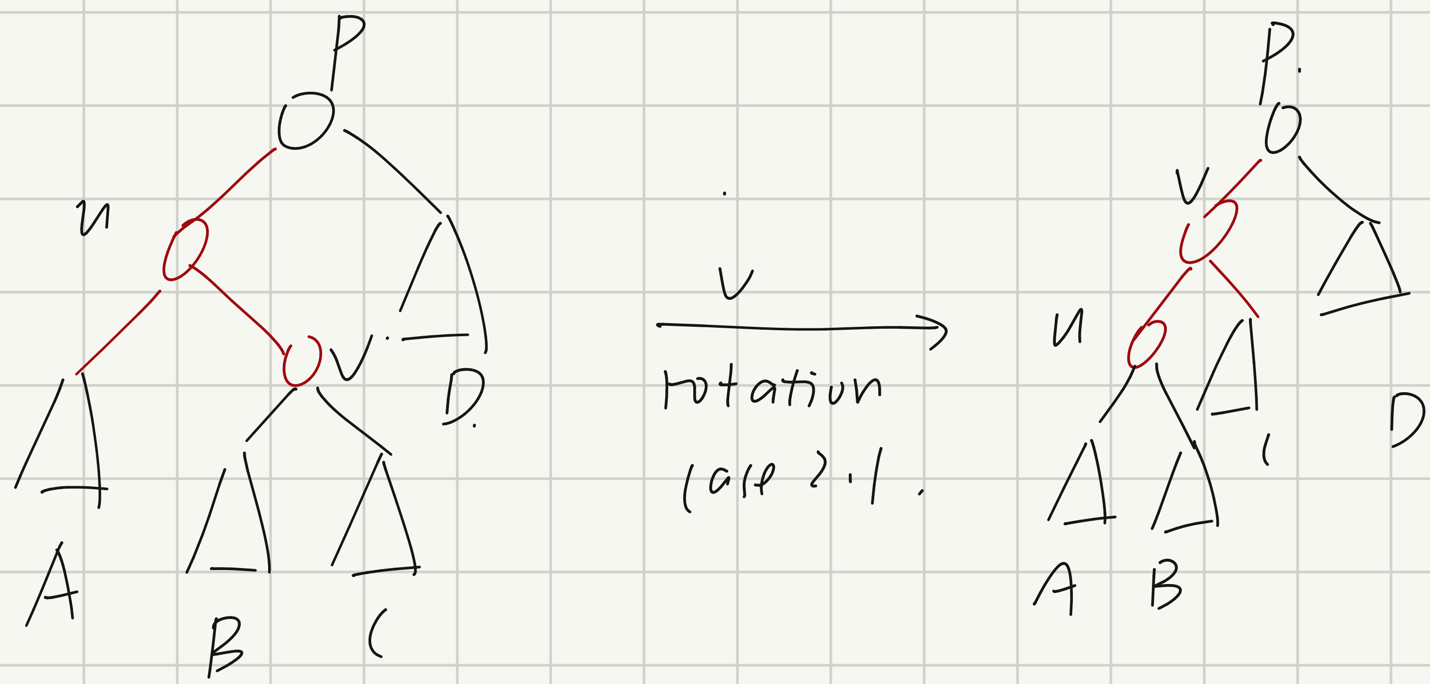
c) parent of P is red, violation goes upwards

Case 2. sibling of u is black.

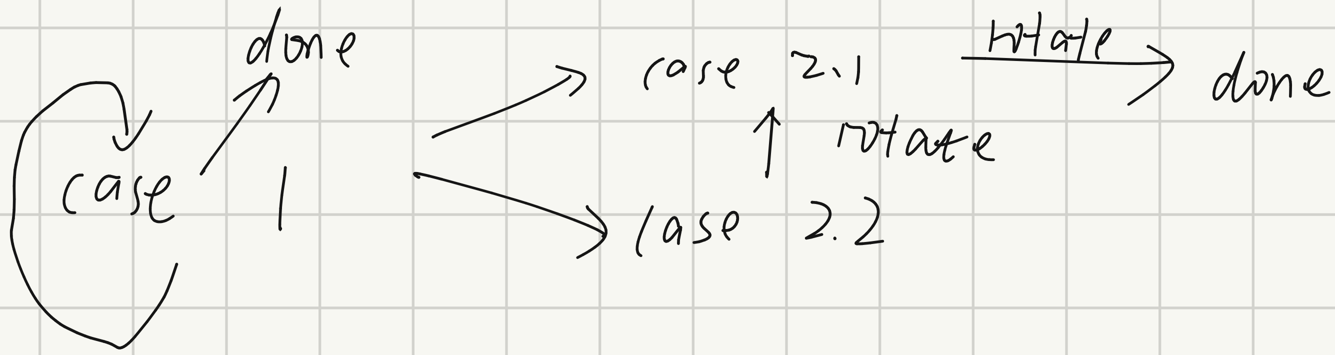
case 2.1 v is left child



case 2.2   v is right child of u.

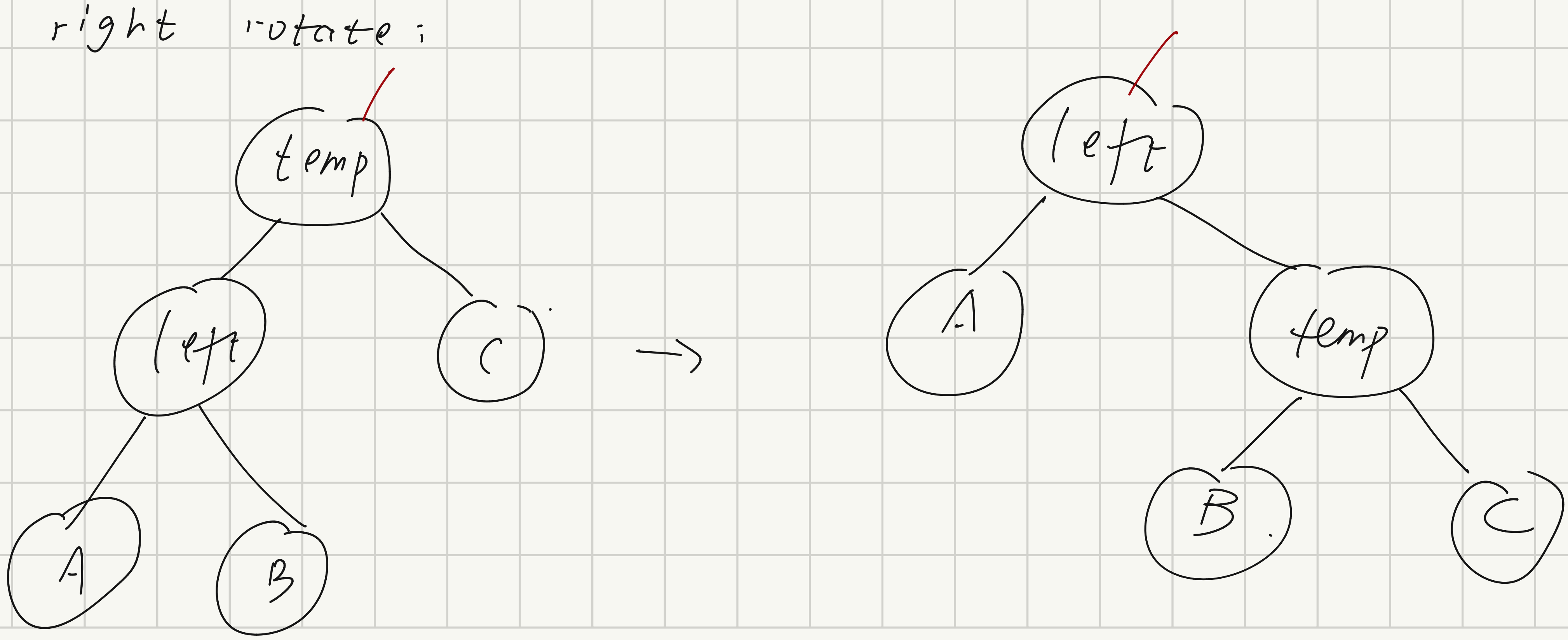


rotation change one level



$$O(\log n) + O(1) \cdot O(\log n)$$

right rotate:



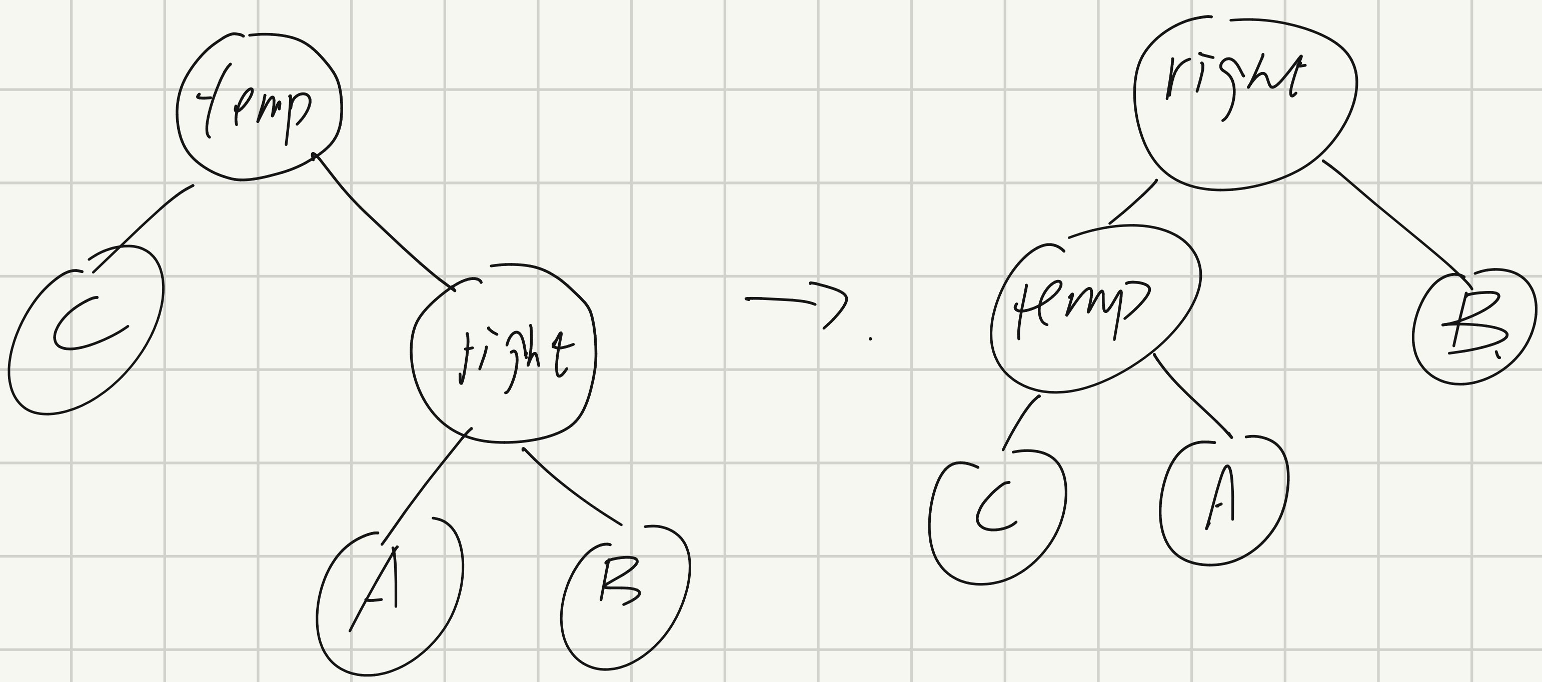
whose parent changed ? : ① temp.
② left
③ B

=> 先 B 再 left 后 temp.

∵ 否则 temp -> p 丢失

whose child changed ? ① temp
② left
③ temp -> p

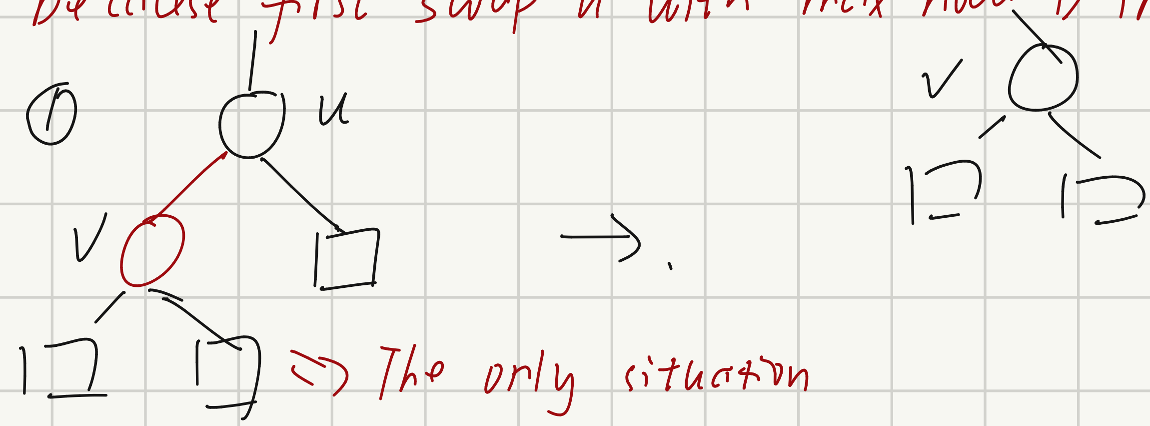
left rotate



RBT: deletion.

1. delete it as in BST. (swap key not color)
(deleted node has one most child excluding new)
because first swap u with max node v in left tree, v's no child / one left child

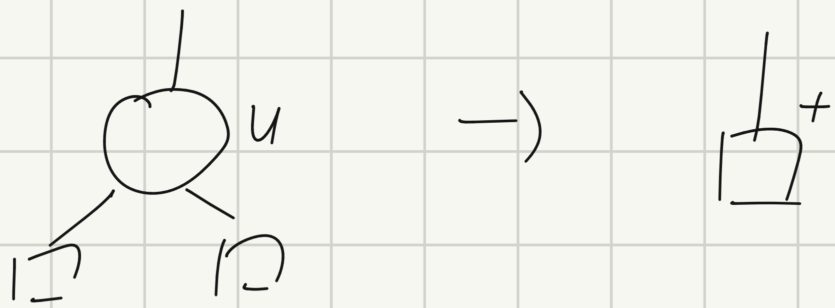
2. ①



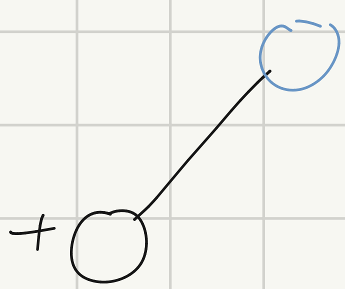
⇒ The only situation when u has a child

②

③



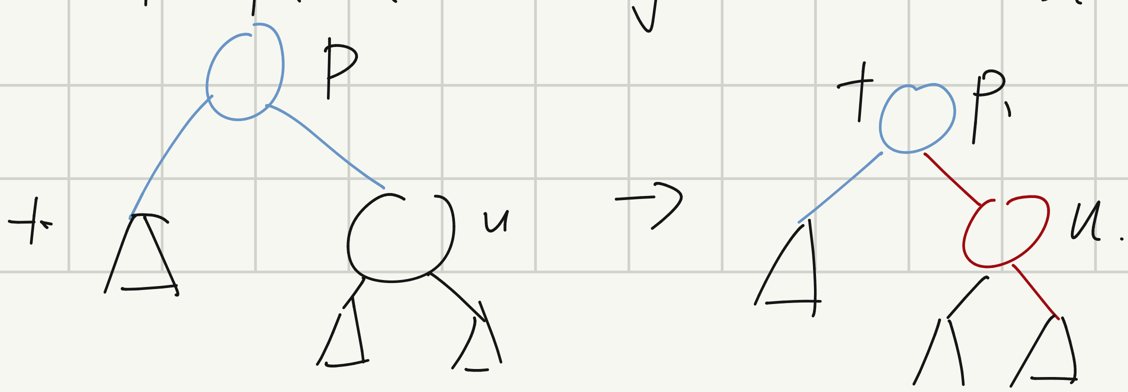
double blank.



db is left child | right child.

case 1: the sibling of db is blank → set u

case 1.1 | children of u are blank.



(a) P is the root ✓

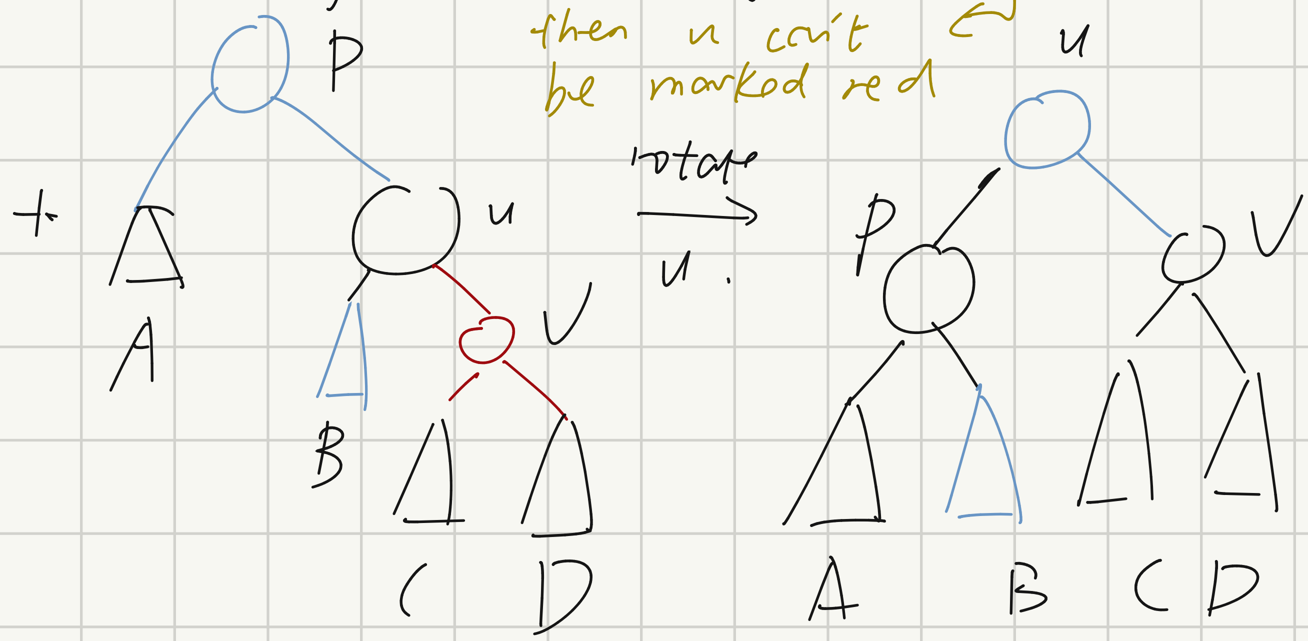
(b) P is red, mark as black.

(c) P is black (not root)

P becomes db.

case 1.2 right child of u is red.

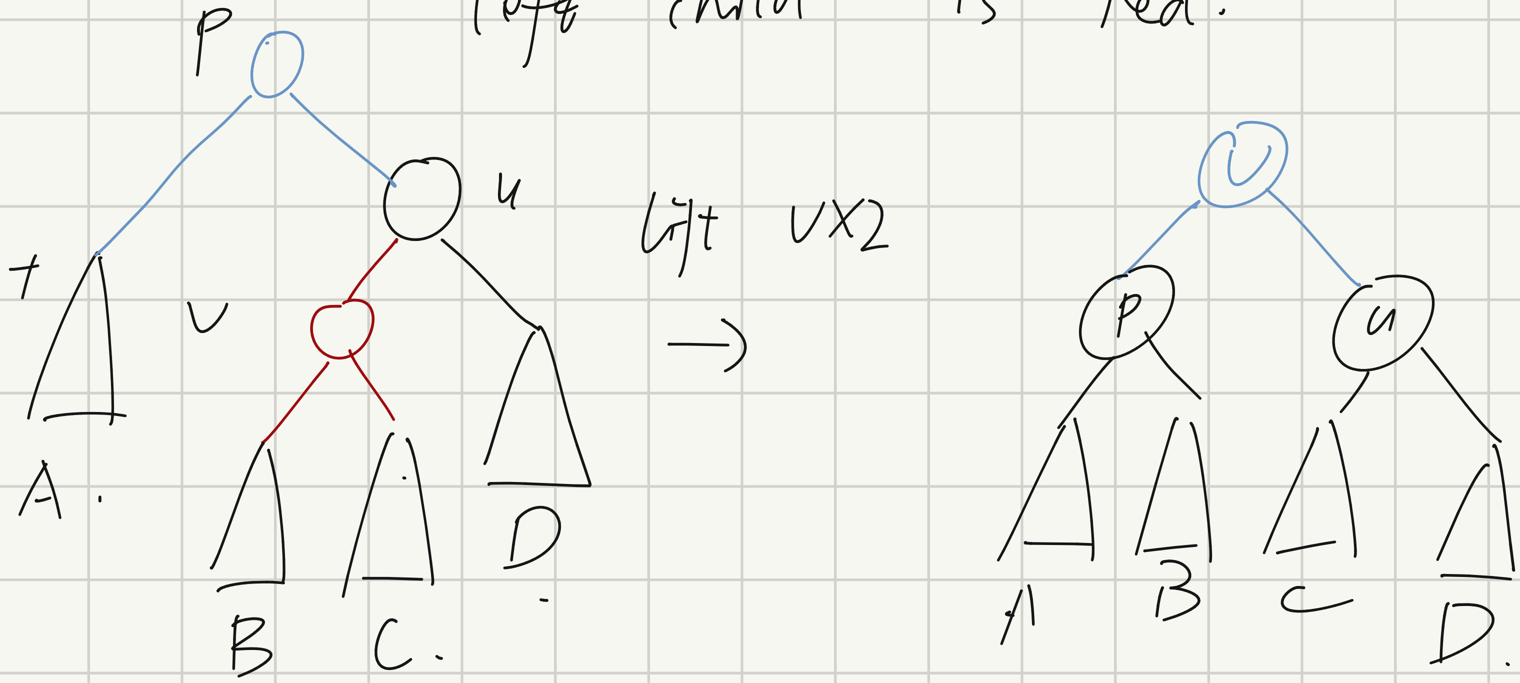
then u can't be marked red



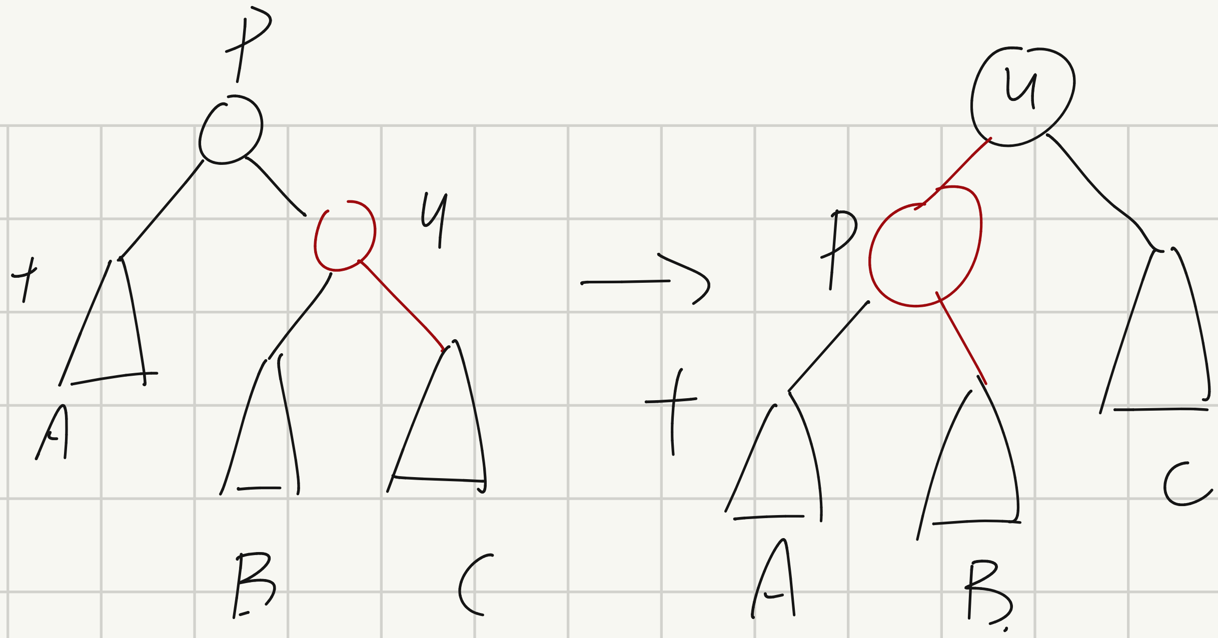
(solve db)

Case 1.3 right child of u is black

left child is red.



case 2: the sibling of 0^+ is red



$\longrightarrow$ return case 1.

case 1.2 $\xrightarrow[\text{rotation}]{\text{single}}$ done

case 1.1 $\longrightarrow$ done

case 1.3 $\xrightarrow[\text{rotation}]{\text{double}}$ done

case 2

single rotation

$\downarrow$

(case 1.1 (b))

$\downarrow$

done

at most 3 times.

$$O(h) + 3\,O(1) = O(\log n)$$

	AVL	RBT
height	$\approx \log_{1.618} n$	$2\log_2 n = \log_{\sqrt{2}} n$.
insertion	at most 2 rotations ✓ (lower)	at most 2 rotations

deletion dlogn) 3

rotations rotations

✓