★ Amortized Analysis

Σ actual cost ≤ Σ amortised cost

PUSH → 1 unit
POP → 1 unit
MULTIPOP (K) → K units

★ To do n operations

Naive $O(n^2)$: Multipop (n) used n times

① Aggregate Analysis:

No. of pushes : O(n)

No. of pops : O(n) (includes multipop)

∴ Aggregate cost : $\frac{O(n)}{n} = O(1)$

② Accounting Method:

Amortized cost = actual cost + Credit

∵ # ∴ $\sum_{s \in S'} \text{credit}(s) \geq 0$

Credit	Operation
+1	Push
-1	Pop
-k	Multipop(k)

Σ credit ≥ 0
(# no. of ele. in stack)

(If only k' obj in stack Multipop (k) = k' + (-k') = 0

③ Potential Method:

ϕ: State of Data Structure (DS) $\rightarrow \mathbb{R}$

Potential of the Data Structure:

$\phi(D) = k$ (no. of elements in stack)

Amortized Cost = actual cost + $\phi(D_{after}) - \phi(D_{before})$

$\because \# \therefore \phi(D_{n'}) \geq \phi(D_0)$

Potential at any point is atleast the initial potential

→ $\phi(D_0) = 0$ (# empty stack)

Push Operation

amortized cost = $1 + (s+1) - s = 2$

Pop operation

amortized cost = $+1 + (s-1) - s = 0$

→ Multipop (k)

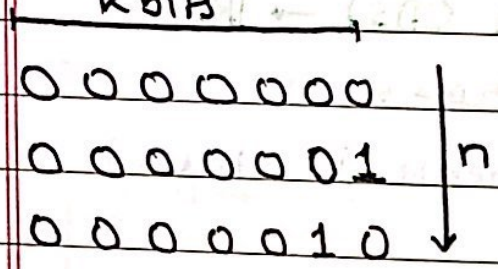
$k + (s-k) - s = 0$

If only k' elements in stack

$k' + (s-k') - s = 0$

∴ Amortized cost is $O(1)$

* Increment a number



* Increment
++

Naive : $O(nk)$ → each bit is changed n times

① Aggregrate :

$$n + \frac{n}{2} + \frac{n}{2^2} + \cdots + \frac{n}{2^{k-1}} < 2n$$

∴ cost is $\frac{2n}{n} = 2$

② Accounting :

Set, reset actual cost : 1

Credit : no. of 1's Reset : -1 Set : +1

0110111 0 $\overbrace{11}^{t_{i-1}}$

0110111 100

Amortized cost = Actual cost + Credit

$= (t_{i-1} + 1) + (-t_{i-1} + 1)$

$= 2$

(reset, set)

1 1 1 1 . . . 1 Actual cost = k

0 0 0 0 . . . 0 Credit = -k

③ Potential :

$\phi(D)$: no. of 1's

$\forall n' \; \phi(D_{n'}) \geq \phi(D_0) = 0$

(# 0, 1 in binary rep)

Amortized cost = Actual cost + $\phi(D_i) - \phi(D_{i-1})$

0110111 $\overbrace{0111}^{t_{i-1}}$ b_{i-1} : No. of 1's

01101111 0000 $\therefore b_i = b_{i-1} - t_{i-1} + 1$

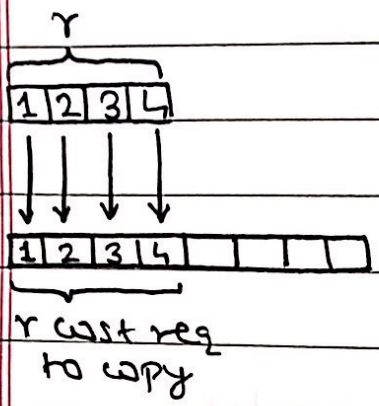
$\therefore AC = (t_{i-1} + 1) + (b_{i-1} - t_{i-1} + 1) - b_{i-1} = 2$

(reset, set)

* If All are ones Actual cost = k, $\phi(D_i) - \phi(D_{i-1}) = -k$

$\therefore AC = 0$

★ Dynamic Array:



★ Cost to insert: 1

★ Double the size of array if it is full

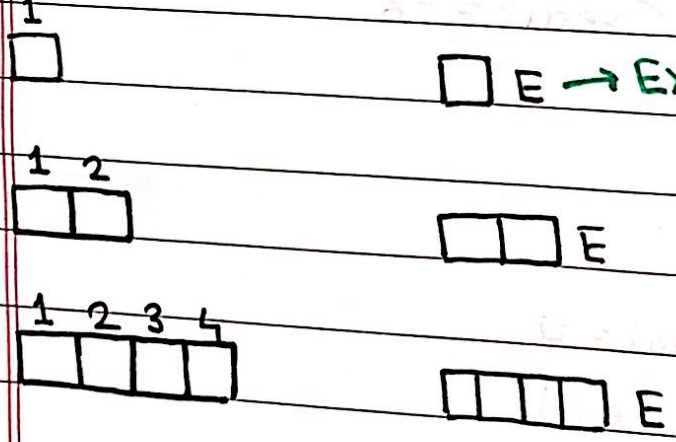
★ Memory allocation zero cost

Naive : $O(n/2 \cdot n)$

↑ size of array before insertion ↑ insert n elem

★ Insert not expands : 1

★ Insert that expands : r+1



① Aggregate:

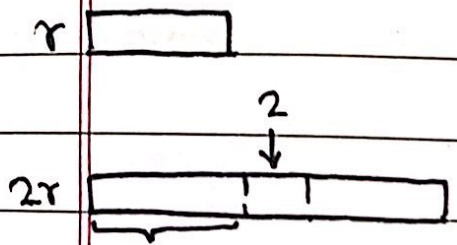
$$n + \sum_{j=0}^{\log n - 1} 2^j \approx 2n-1$$

$\therefore$ cost is $\frac{2n-1}{n} = 2$ $\therefore$ insert is $O(1)$

② Accounting Method:

Insertion : 2 credits

Copy : -r credits



★ Actual cost
$= r+1$

★ Credit
$= -r+2$

$\therefore AC = (r+1) + (-r+2) = 3$

$\sum$ Credit ≥ 0

③ Potential Method:

$\phi(D_i) : 2n(D_i) - s(D_i)$

no. of slots occupied ; size of DS

$\phi(D_0) = 0$

$\therefore \phi(D') = 2n(D') - s(D') \geq 0$

If $2n - s < 0$ then $n < s/2$ (Then why did you expand the DS)

→ Actual cost $= r+1$

$\phi(D_{after}) - \phi(D_{before}) = [2(r+1) - 2r] - [2r - r]$
$= 2 - r$

$\therefore AC = r+1+2-r = 3$

★ No expansion:

Actual cost $= 1$, $\phi(D_{after}) - \phi(D_{before}) = [2(n+1) - s] - [2n - s] = 2$

$\therefore AC = 1+2 = 3$

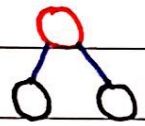
✱ Red-Black Trees:

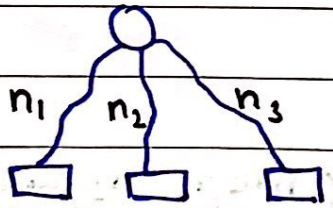
→ Self Balancing BST

→ Extendend BST:

- Every leaf node has a dummy empty node.

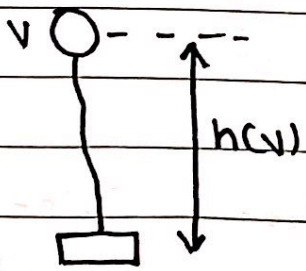
→ Property:

1) Red or Black
2) Root Black
3) Every leaf Black
4)  (No two consecutive red)
5) From a node to a leaf node equal no. of black nodes.



$n_1 = n_2 = n_3$

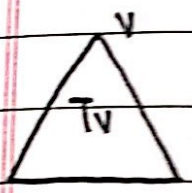
→ Black height of the tree / node: no. of nodes (black) in the path after that node to the leaf.



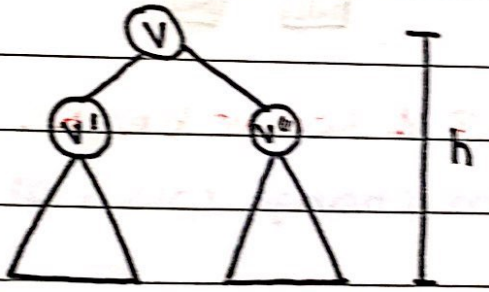
black height $\geq h(v)/2$

$\therefore bh(v) \geq h(v)/2$

* No of internal nodes in $T_v \geq 2^{bh(v)} - 1$



Basis: height of node is 0, only one node i.e. black



→ $v', v'' = B$

∴ no. of internal nodes $\geq (2^{bh(v')} - 1) + (2^{bh(v'')} - 1) + 1$

$\geq 2.2^{bh(v')} - 1$

$\geq 2.2^{bh(v')+1}/2 - 1$

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

→ $v' = B$ & $v'' = R$

∴ no. of internal nodes $\geq (2^{bh(v')} - 1) + (2^{bh(v'')} - 1) + 1$

(# ⑤ & Def of bh) $\geq (2^{bh(v')} - 1) + (2^{bh(v')+1} - 1) + 1$

$\geq 2.2^{bh(v')} - 1$

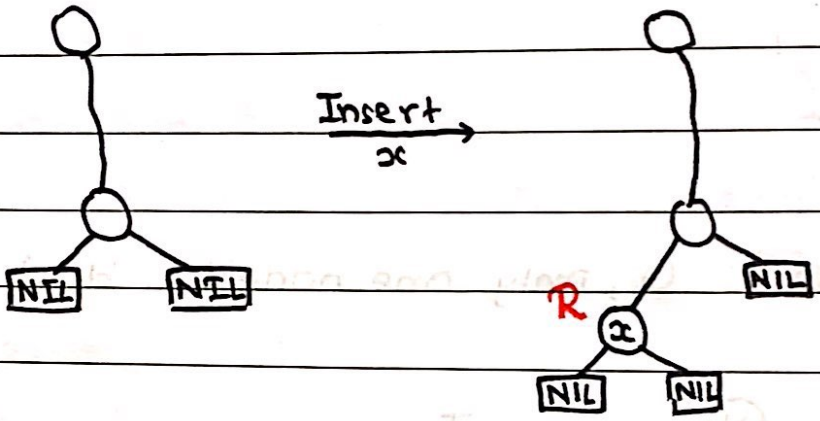
$\geq 2^{bh(v)} - 1$

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

∴ no. of internal nodes in $T \geq 2^{bh(root)} - 1 \geq 2^{\frac{h(T)}{2}} - 1$

∴ $2\log(n+1) \geq h(T)$ ∴ $h(T) = O(\log n)$

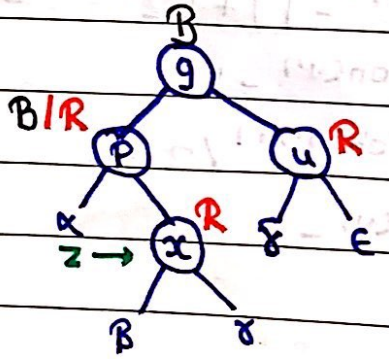
✱ Insertion:



→ Insert like in BST & colour be R.

→ If x is root then change colour of x to black.

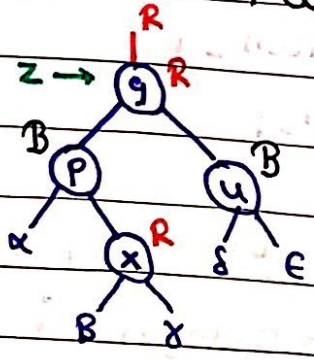
① Case 1: Uncle is Red



g: grandparent
p: parent
u: uncle
x: inserted

If p is B , u → B & g → R

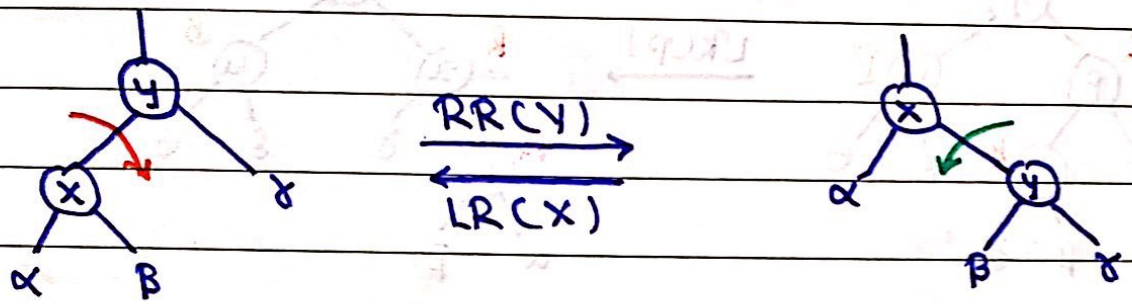
p is R , u → B & g → R and p → B



if parent of g is red our z has shifted up by two levels. else we have successfully recoloured.

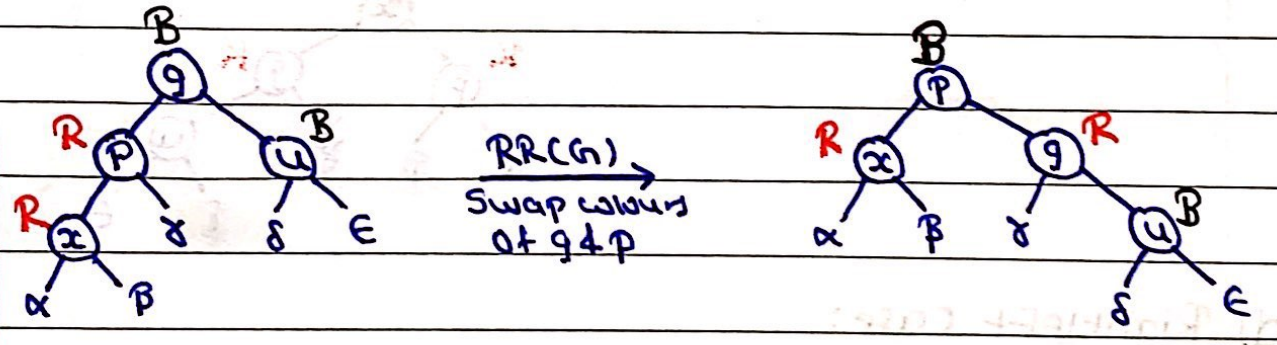
(2) Case 2: Uncle is Black

→ Rotation: OCII: Adjust pointers

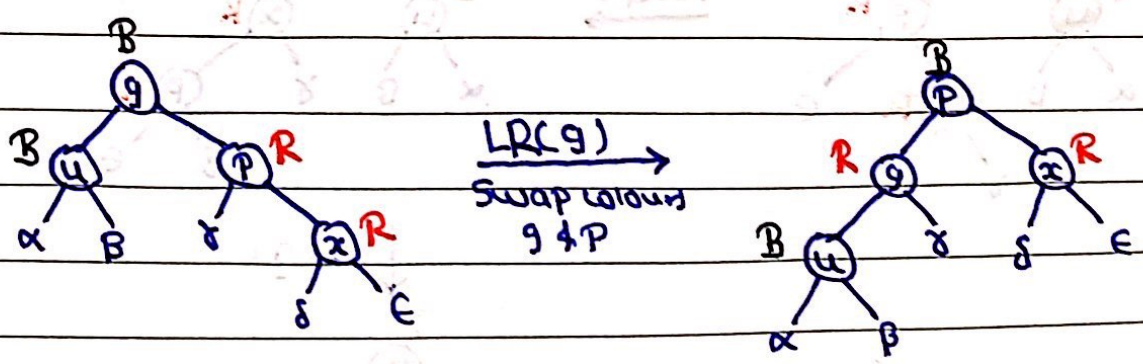


* Problem when p → R ⇒ g → B (# initially R-B tree)

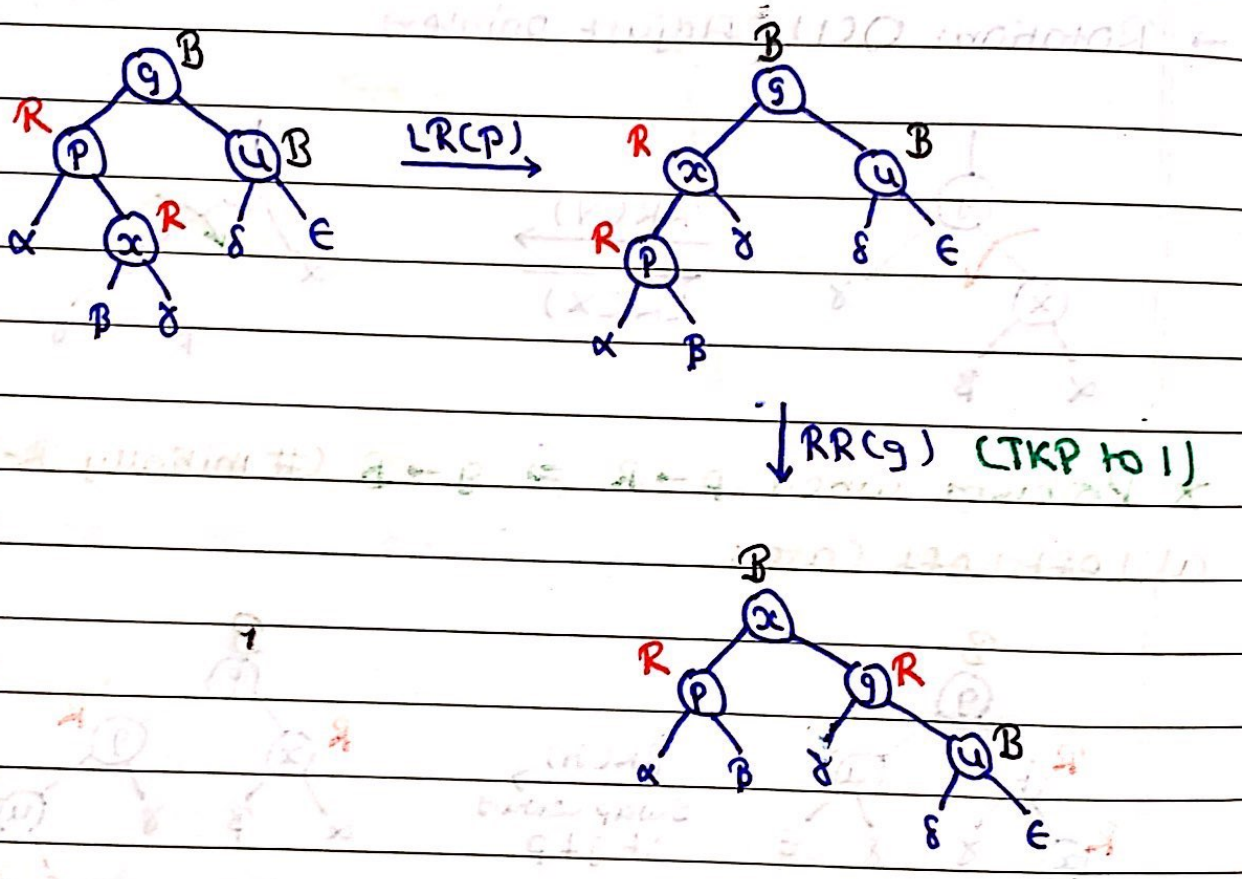
a) Left Left Case:



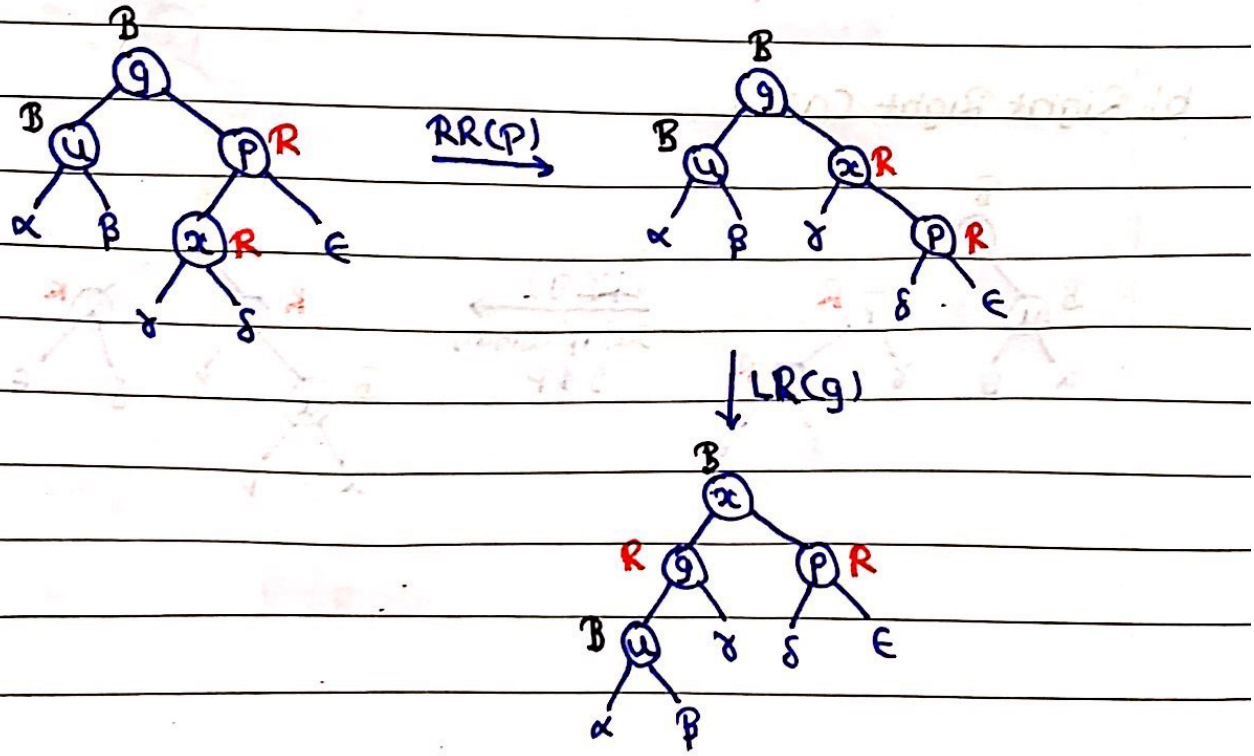
b) Right Right Case:



c) Left Right Case:

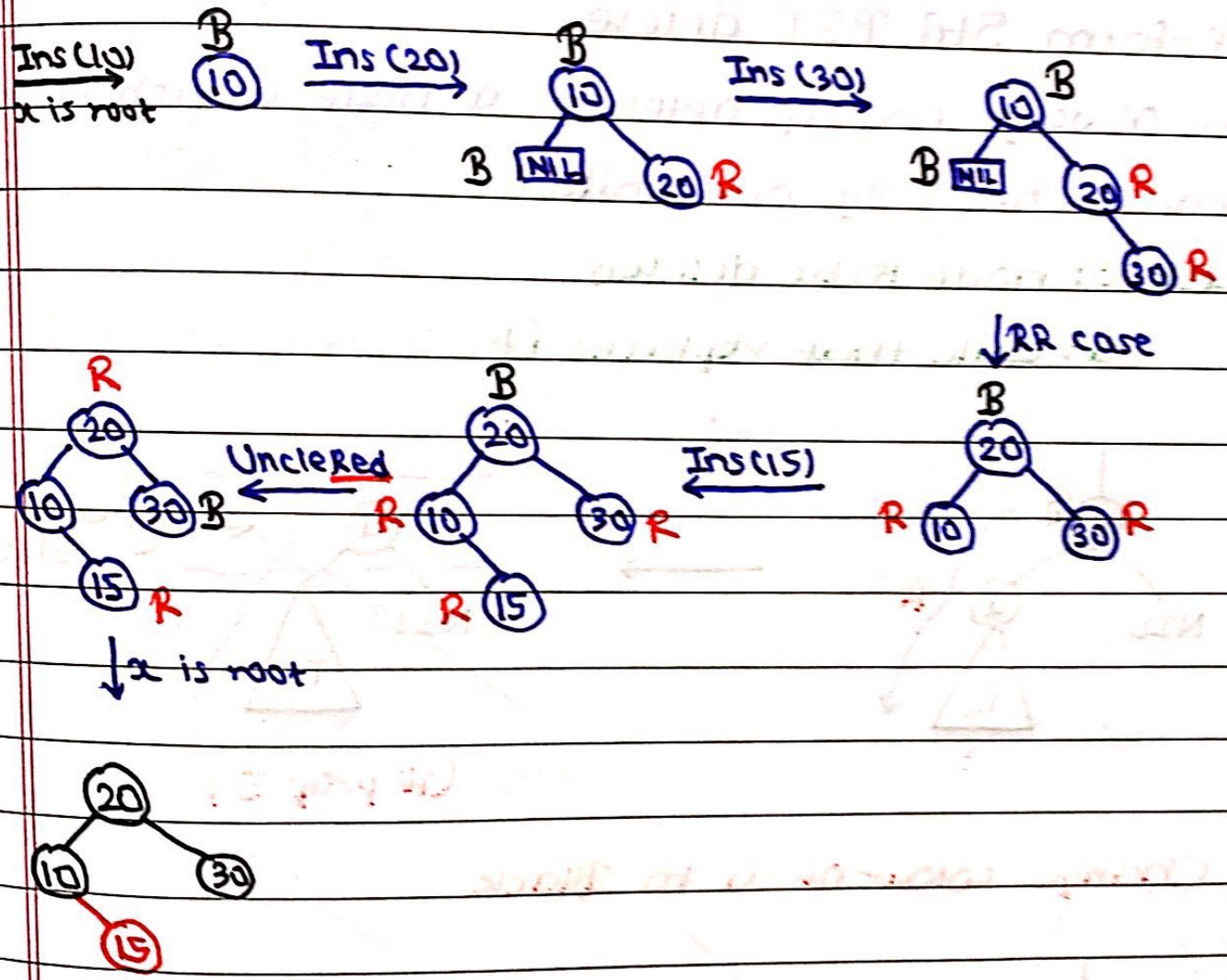


d) Rightleft Case:



Date
Page

★ Insert 10, 20, 30, 15 in empty tree



Date
Page

* Delete:

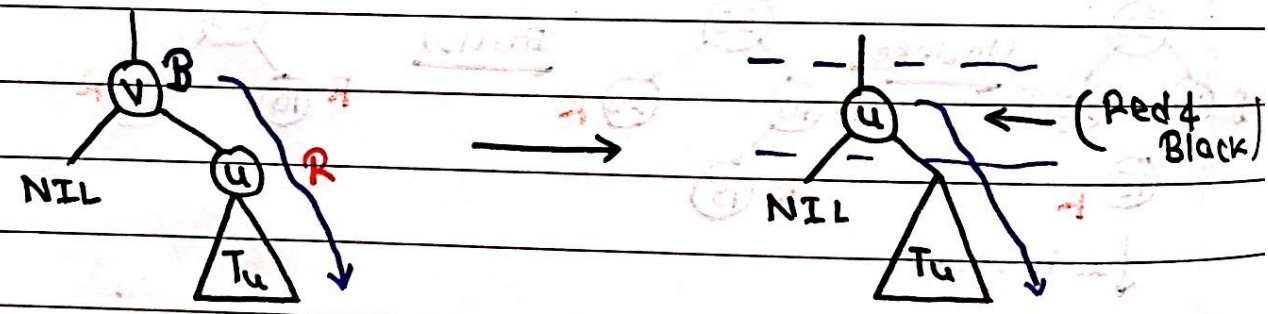
→ Perform Std BST delete

we always end up deleting a node which is a leaf or has only one child

Let v: node to be deleted

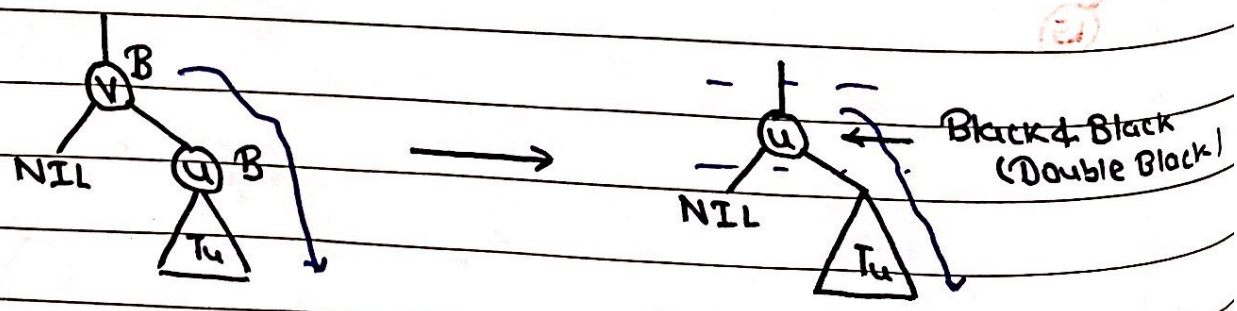
u: child that replaces it

① v: Black



(# Prop 5)

→ Change colour of u to Black



(# Prop 5)

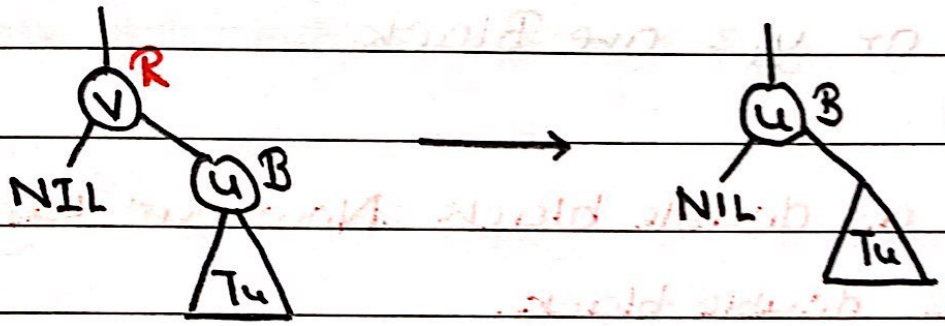
Between blue lines

initially no. of blacks = 2

that should be preserved after deletion.

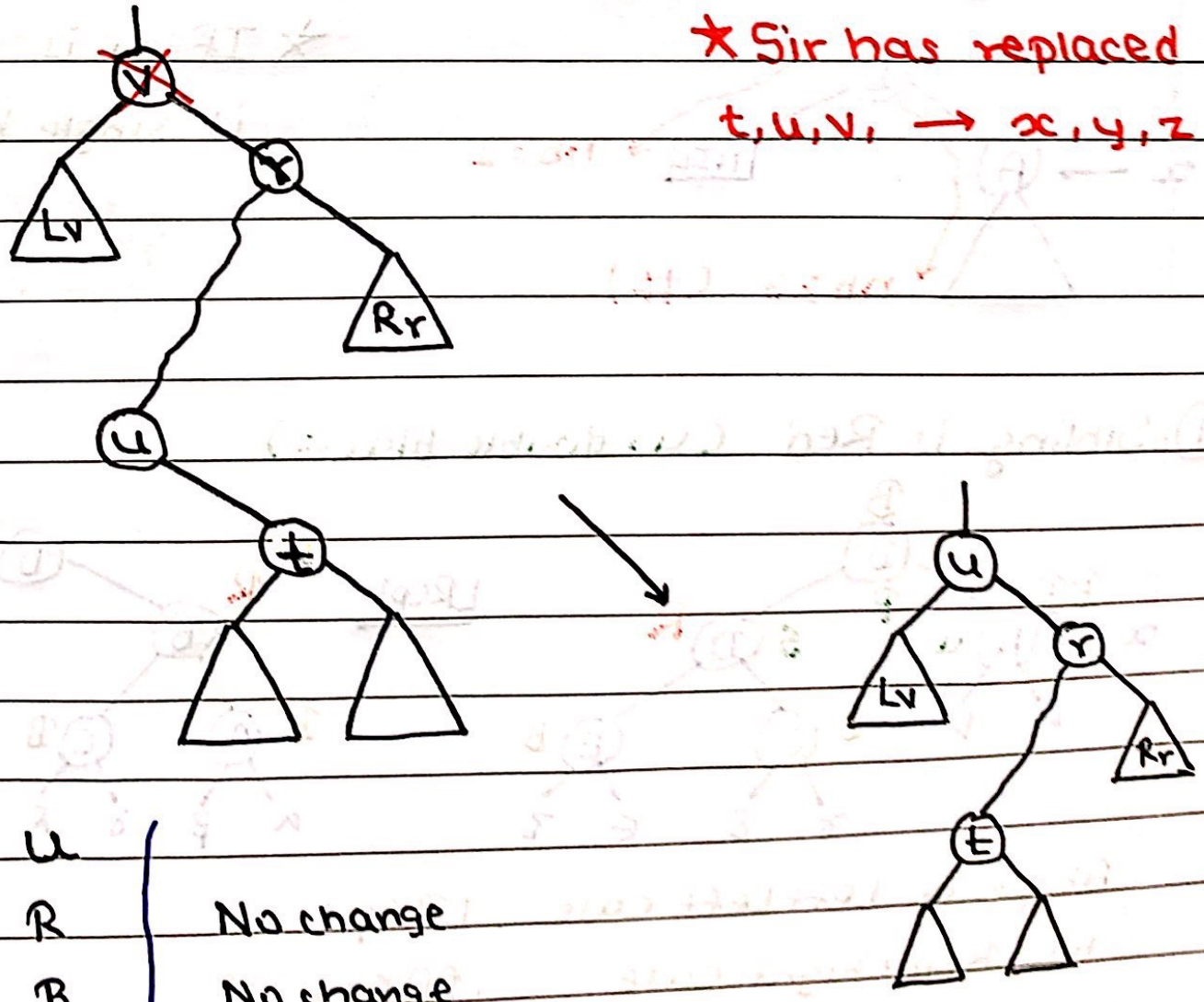
→ Above case discussed later.

② V: Red



(No need to do anything)

* Sir has replaced
t, u, v, → x, y, z



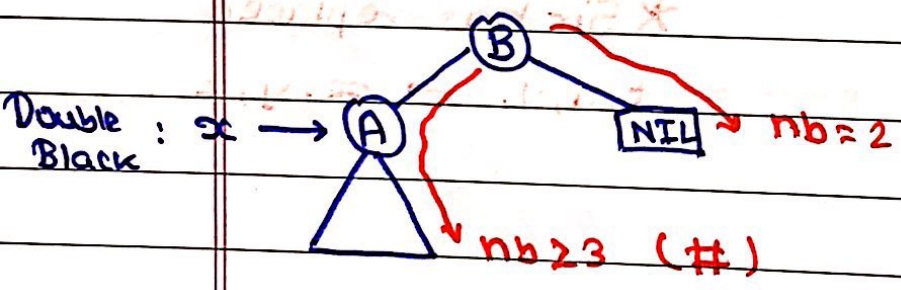
V	u	
R	R	No change
R	B	No change
B	R	Change u to Black
B	B	Double Black

★ Black & Black (Double Black Case)

Both u, v or y, z are Black.

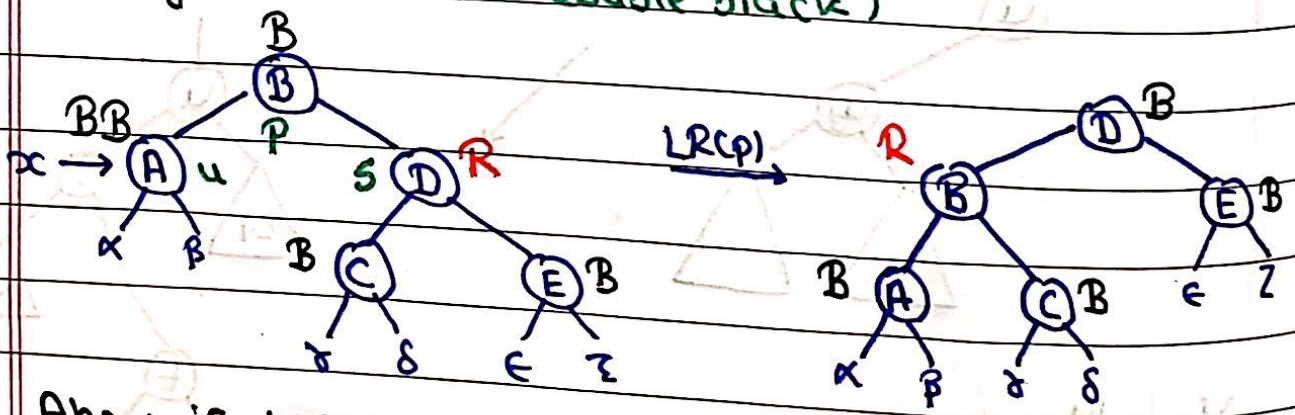
→ Colour u as double black Now our task is to reduce this double black.

→ Siblings cant be NIL



★ If u is root make it single black

① Sibling is Red (u double black)

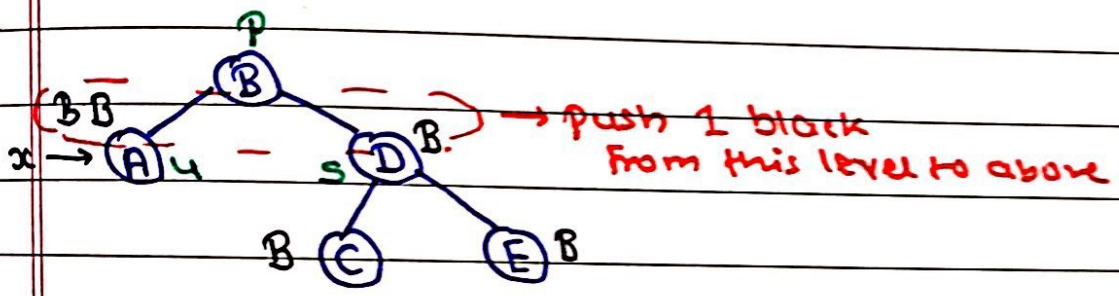


Above is Left Left case , LR(P)

For Right Right case , RR(P)

② Sibling is black

① Both the children are black



IF P is red it becomes Red & Black colour it to Black else P is black & black & recour.

IF P was black