Red Black Tree

Deletion

Properties of red black tree

Property #1: Red - Black Tree must be a Binary Search Tree.

Property #2: The ROOT node must be colored BLACK.

Property #3: The children of Red colored node must be colored BLACK. (There should not be two consecutive RED nodes).

Property #4: In all the paths of the tree, there should be same number of BLACK colored nodes.

Property #5: Every new node must be inserted with RED color.

Property #6: Every leaf (e.i. NULL node) must be colored BLACK.

Deletion in Red Black Tree

Step 1: Remove as in BST, replace it (the data in the node) then delete the leaf one

a. Leaf (no children) -> just delete it
b. 1 child -> replace it with its child, then delete its child
c. 2 children -> replace it with its predecessor or its successor, then delete this child

Deletion in Red Black Tree

Step 2: Before the deletion of the leaf node, Check its color:

- If Red: just delete it
- If Black: GO to STEP 3

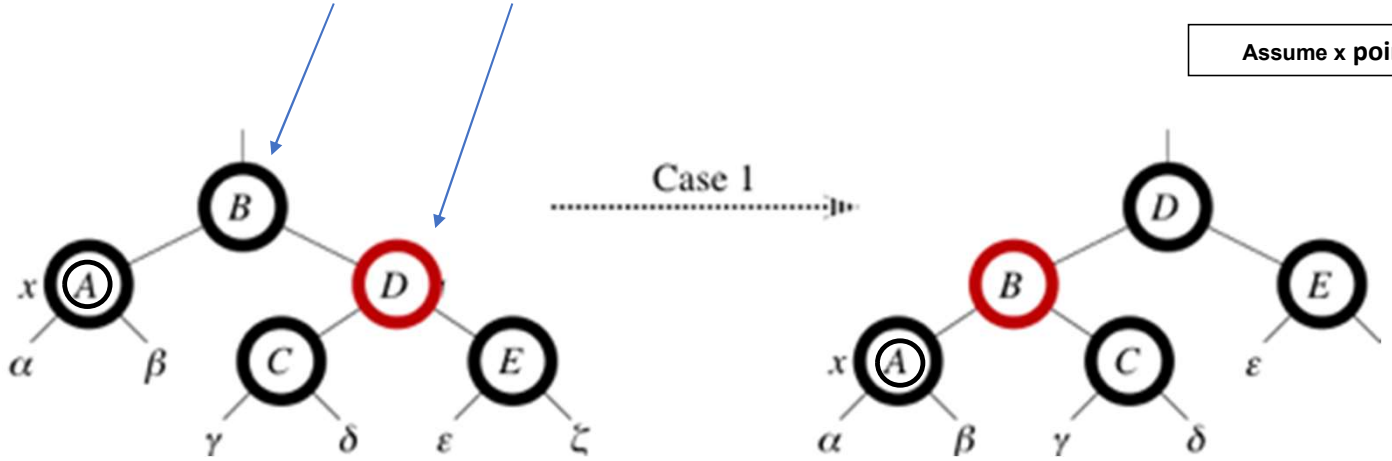
Step 3: Replace this node with its child (nil node) and its color became double black (DB)

Deletion in Red Black Tree

Step 4: Check its brother's color

- If Red: -> Case 1

Rotation and recoloring (parent and brother)



Deletion in Red Black Tree

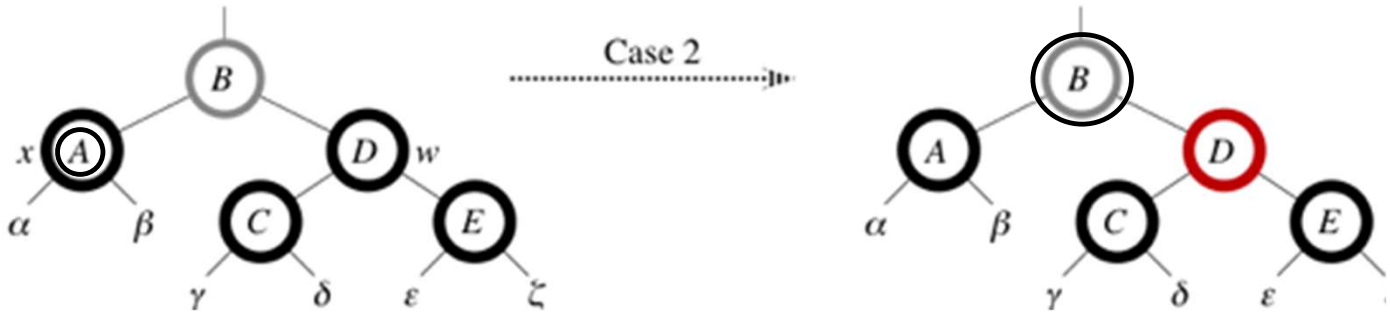
Step 4: Check its brother's color

- If Black: -> GO TO STEP 5

Step 5: Check your brother's children [colors]

- If the 2 children are black [in the example: C & E]-> Case 2
  - Parent will take the DB (if red > black, if black > DB)
  - Then recolor my brother [D]

Assume x points to a doubly black node

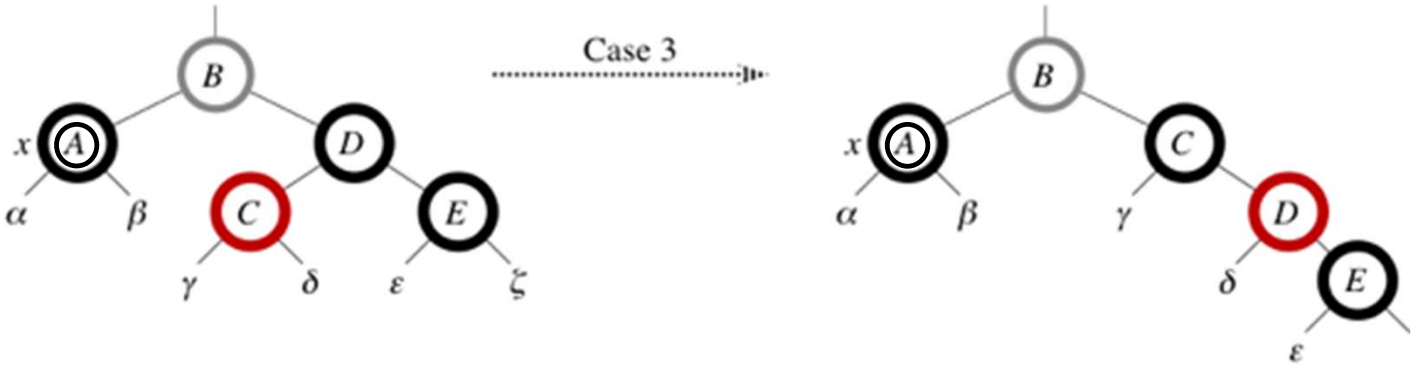


Deletion in Red Black Tree

Step 5: Check your brother's children [colors]

- If the near child is Red [in the example: C], the other child black -> Case 3
  - Rotation and recoloring [brother (D) and its near child (C)], then go to *Case 4*

Assume x points to a doubly black node

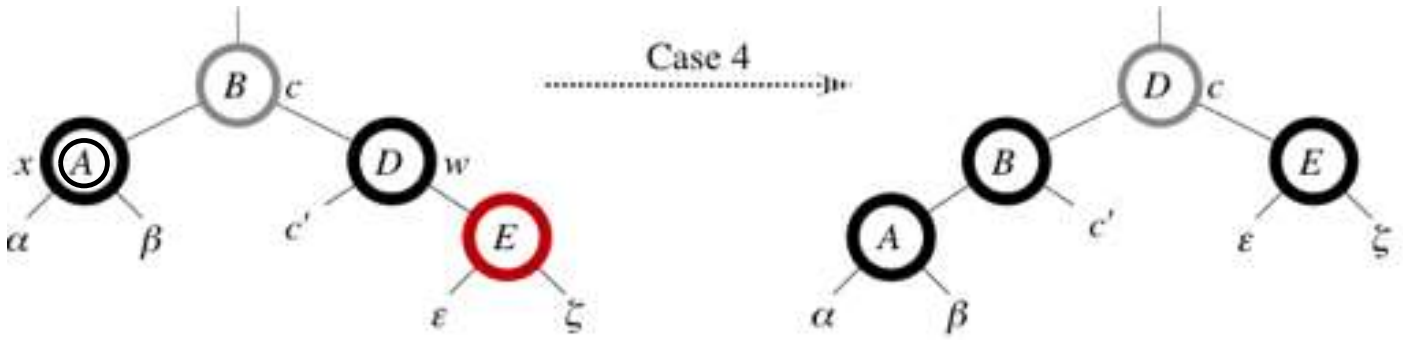


Deletion in Red Black Tree

Step 5: Check your brother's children [colors]

- If the far child is Red [in the example: C] (or both red) -> Case 4
  - Rotation [parent (B) and brother (D)], then **DB will be removed.**
  - Then Recoloring:
    - Parent (B) >> black & brother (D) >> color of the parent (as if they exchange their colors)
    - brother's child (E) >> black

Assume x points to a doubly black node

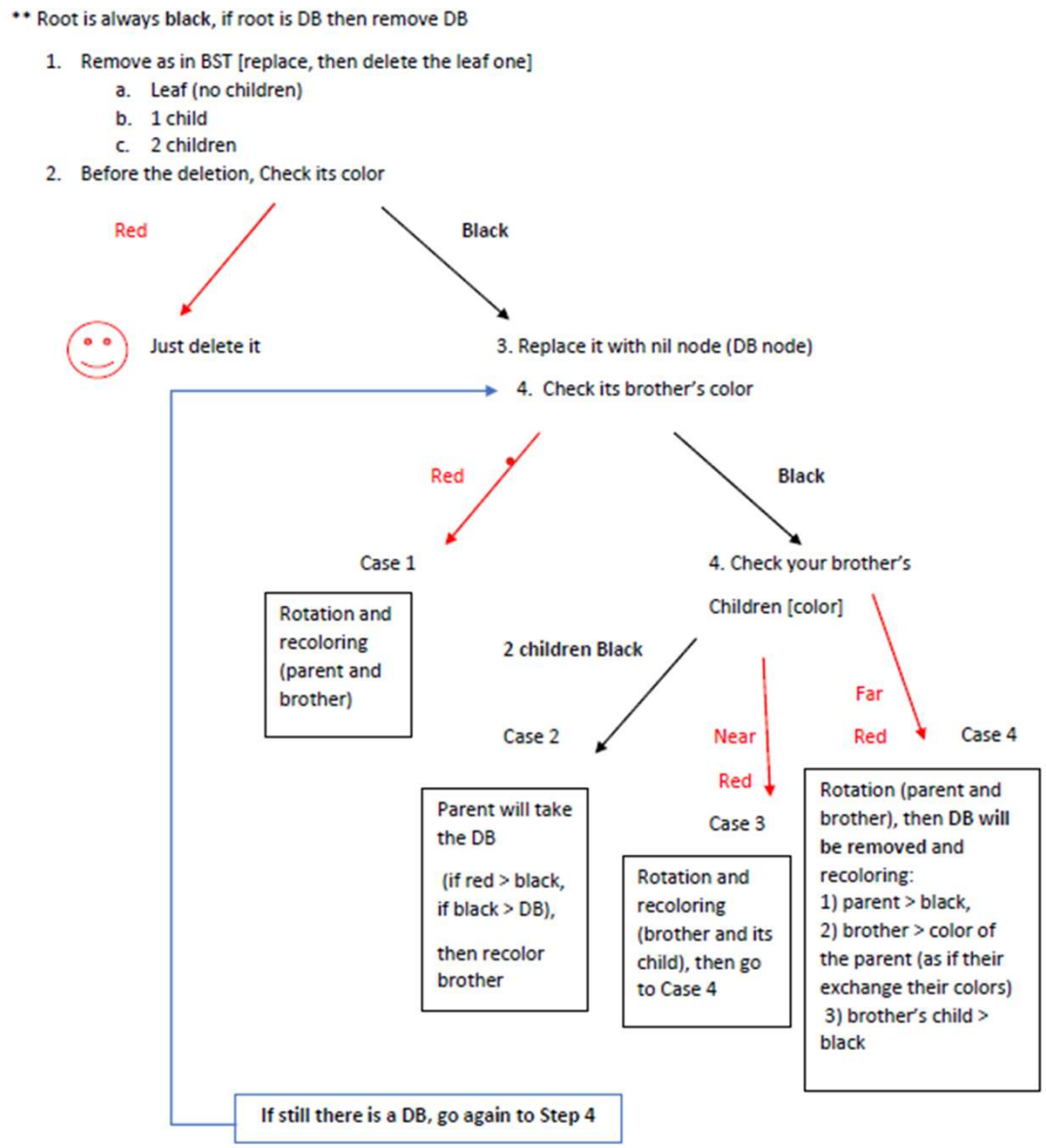


Deletion in Red Black Tree

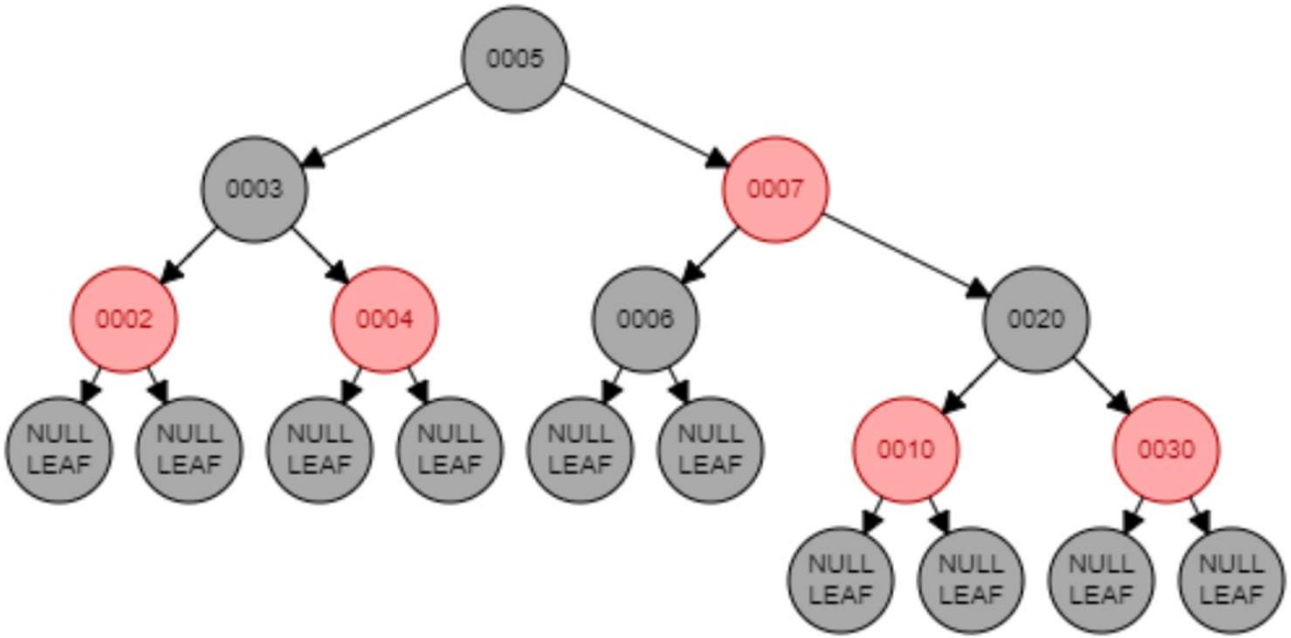
If still there is a DB, go again to Step 4

Root is always black, if root is DB then just remove DB

Summary


** Root is always black, if root is DB then remove DB
1. Remove as in BST [replace, then delete the leaf one]
a. Leaf (no children)
b. 1 child
c. 2 children
2. Before the deletion, Check its color
Red
Black
Just delete it
3. Replace it with nil node (DB node)
4. Check its brother's color
Red
Black
Case 1
Rotation and recoloring (parent and brother)
4. Check your brother's
Children [color]
2 children Black
Case 2
Parent will take the DB
(if red > black, if black > DB),
then recolor brother
Near
Red
Case 3
Rotation and recoloring (brother and its child), then go to Case 4
Far
Red
Case 4
Rotation (parent and brother), then DB will be removed and recoloring:
1) parent > black,
2) brother > color of the parent (as if their exchange their colors)
3) brother's child > black
If still there is a DB, go again to Step 4

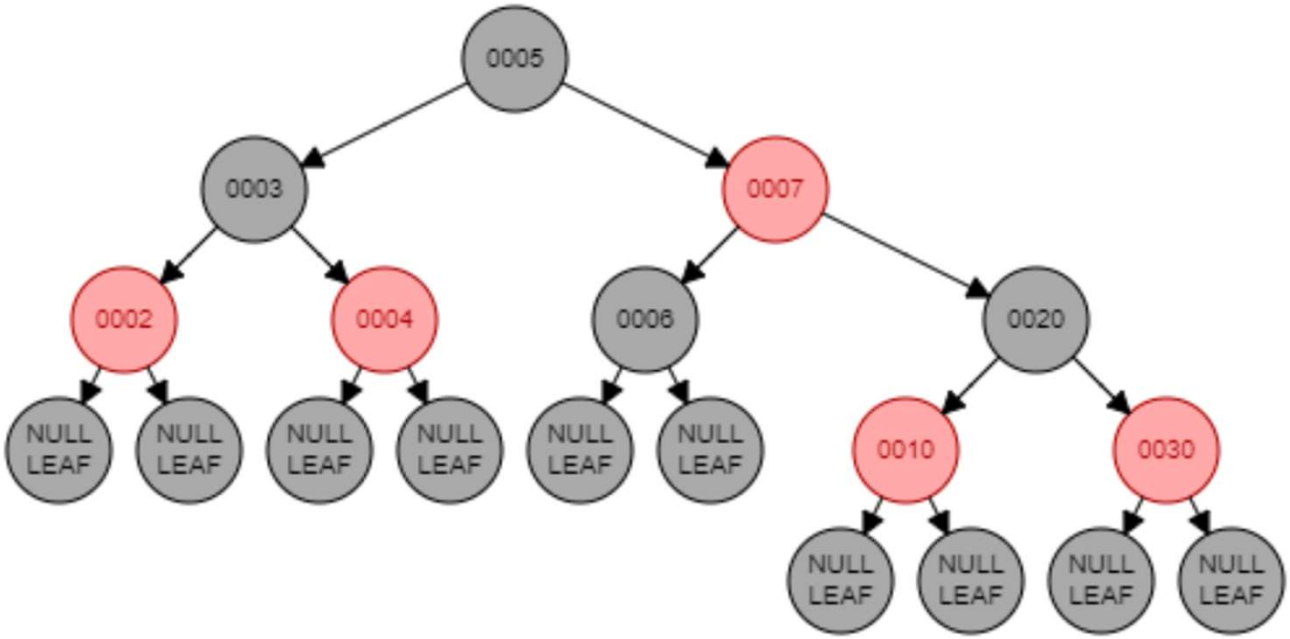
Example:


0005
0003
0007
0002
0004
0006
0020
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
0010
0030
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF

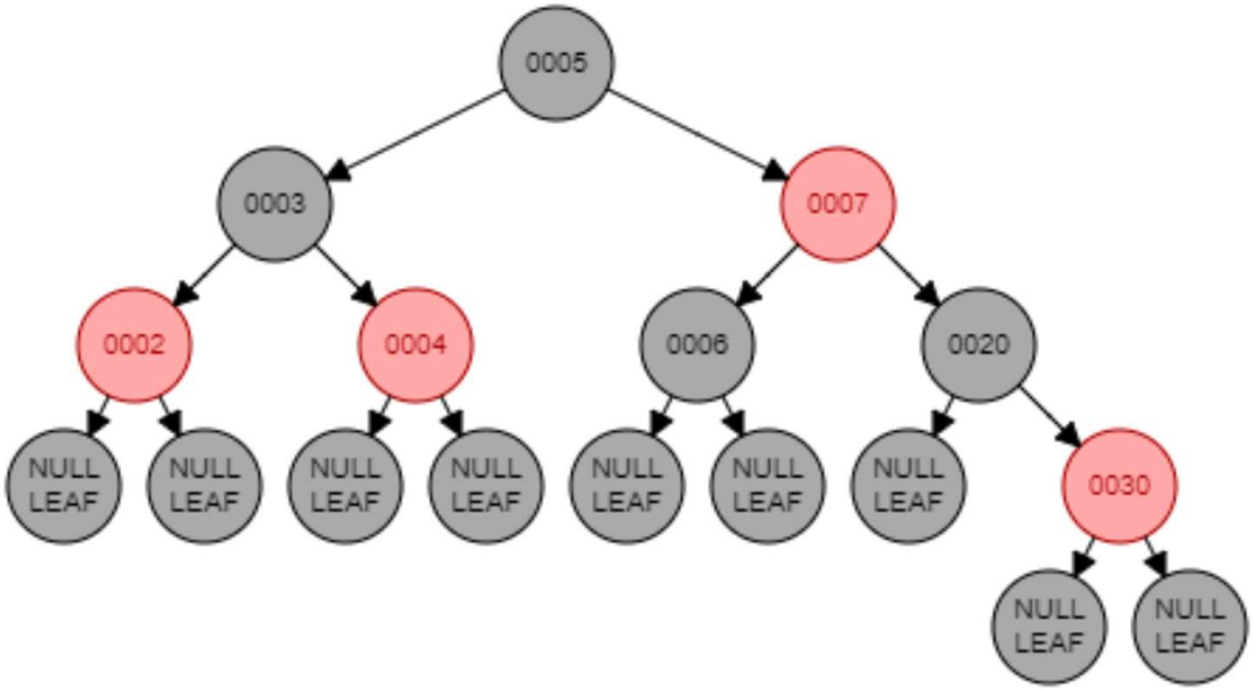
Delete 10

Step 1: Here 10 is a leaf node,

Step 2: check its color -> Red, just delete it

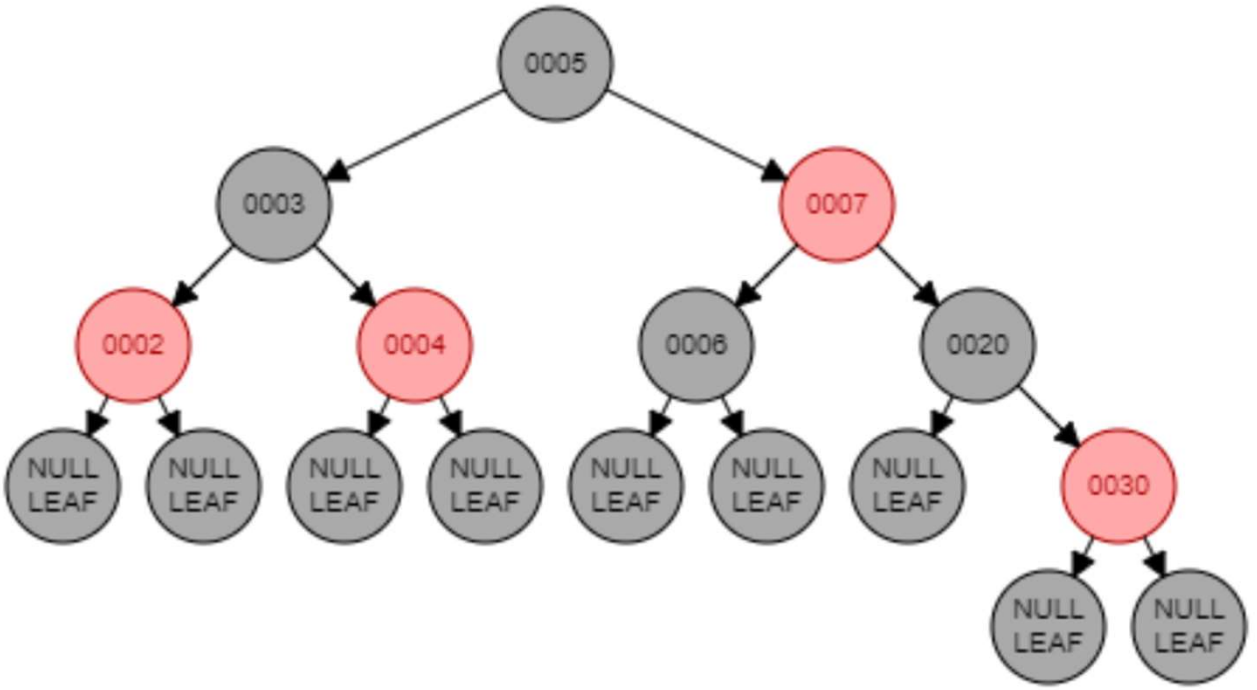


Delete 10


0005
0003
0007
0002
0004
0006
0020
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF
0030
NULL LEAF
NULL LEAF

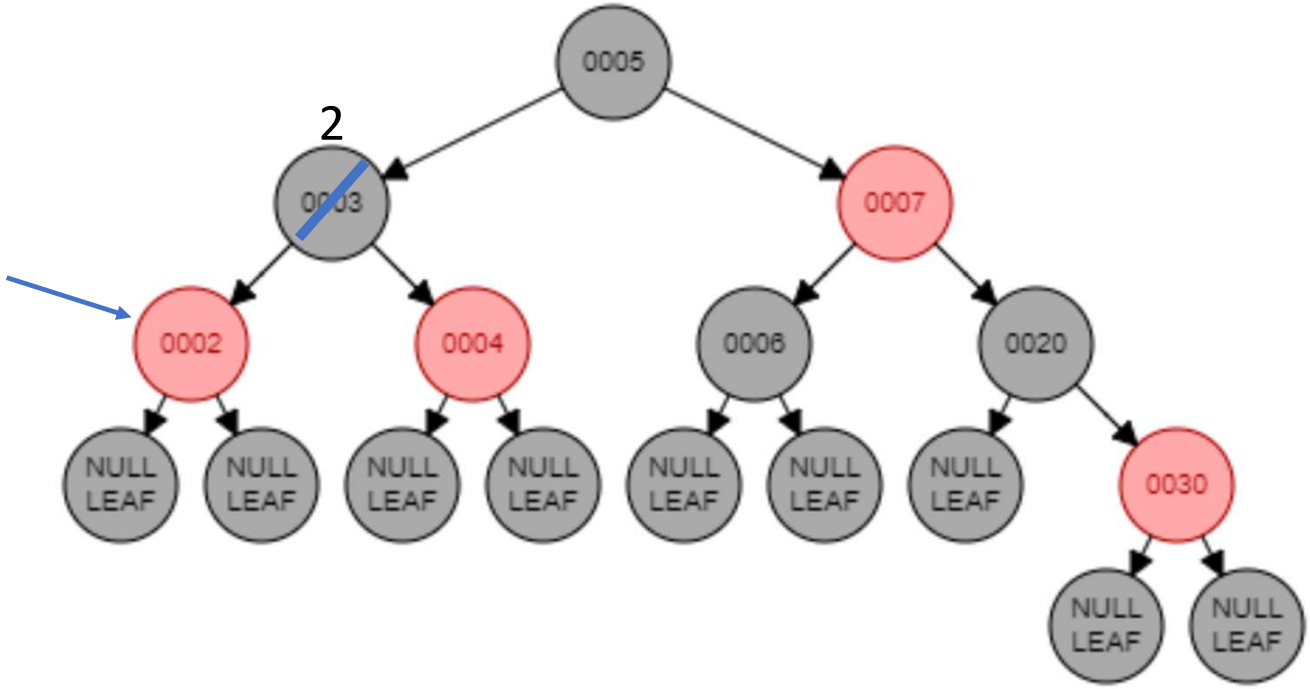
Delete 3

Step 1: Here 3 has 2 children, so first replace it with its predecessor or its successor, then delete this child



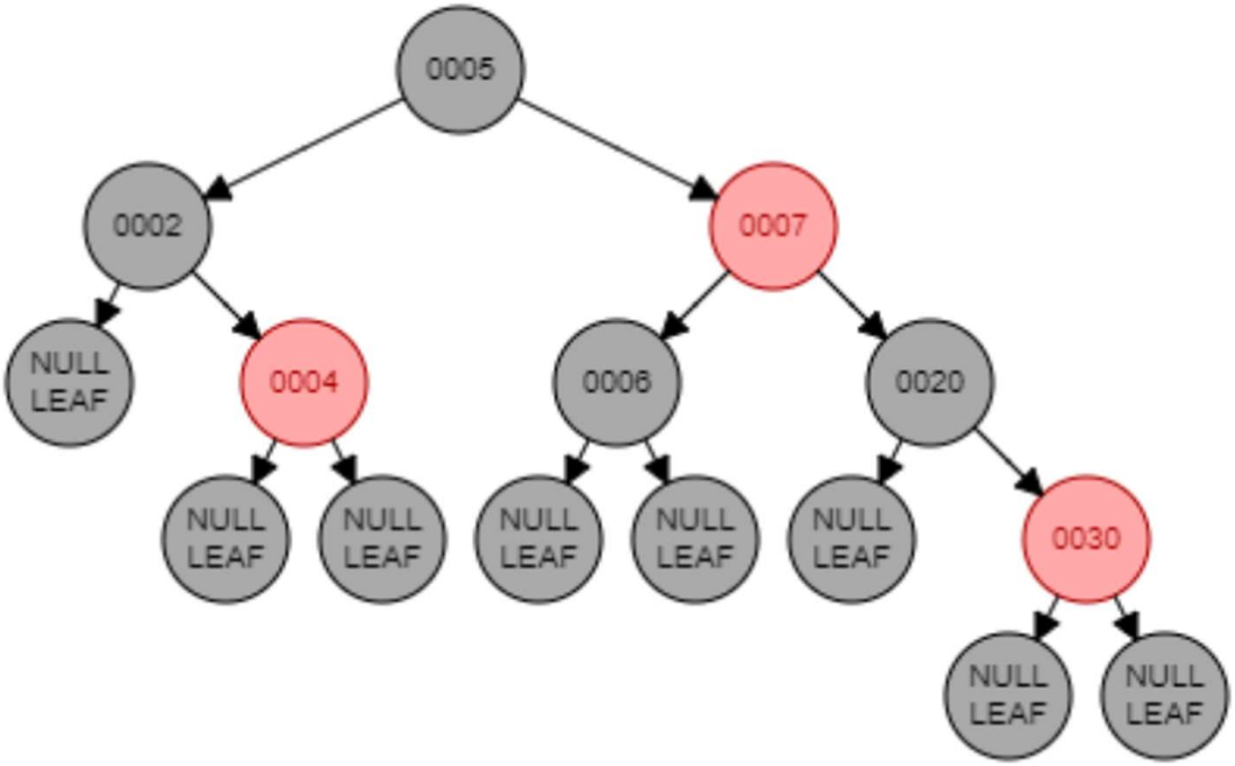
Delete 3

Step 2: Check its color first -> Red, just delete it



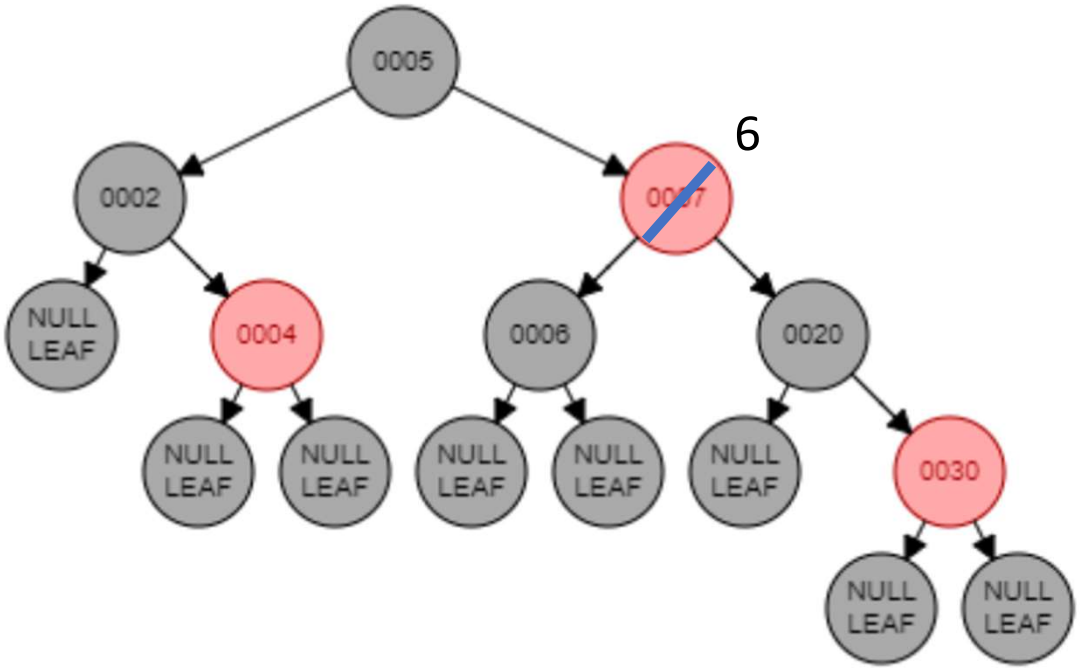
Delete 7

Step 1: Here 7 has 2 children, so first replace it with its predecessor or its successor, then delete this child



Delete 7

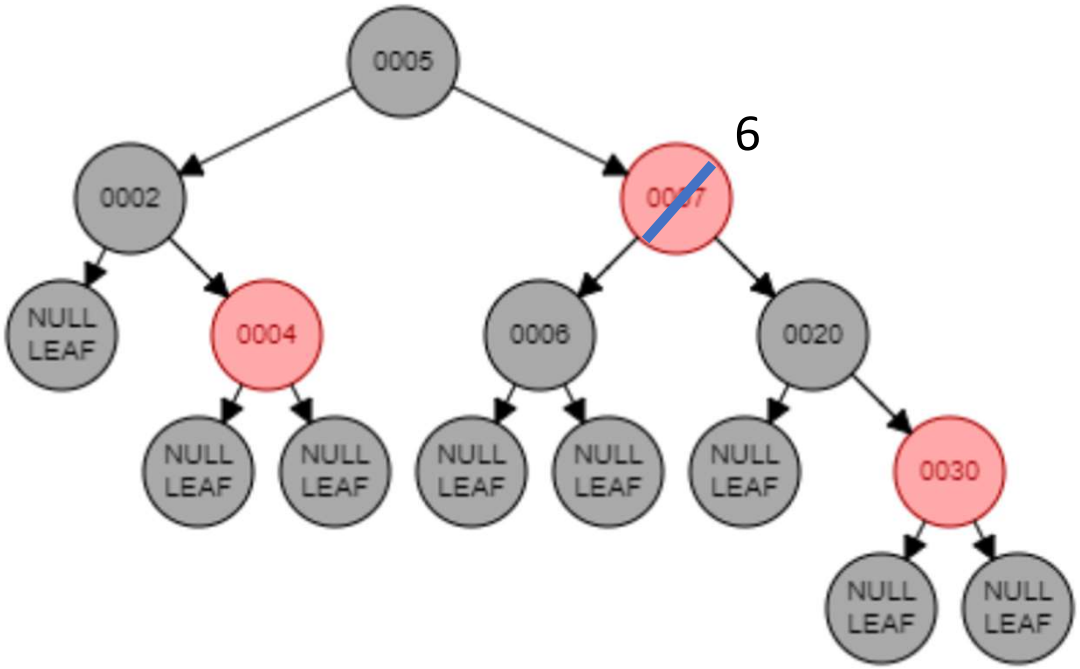
Step 1: Here 7 has 2 children, so first replace it with its predecessor or its successor, then delete this child [6]



Delete 7

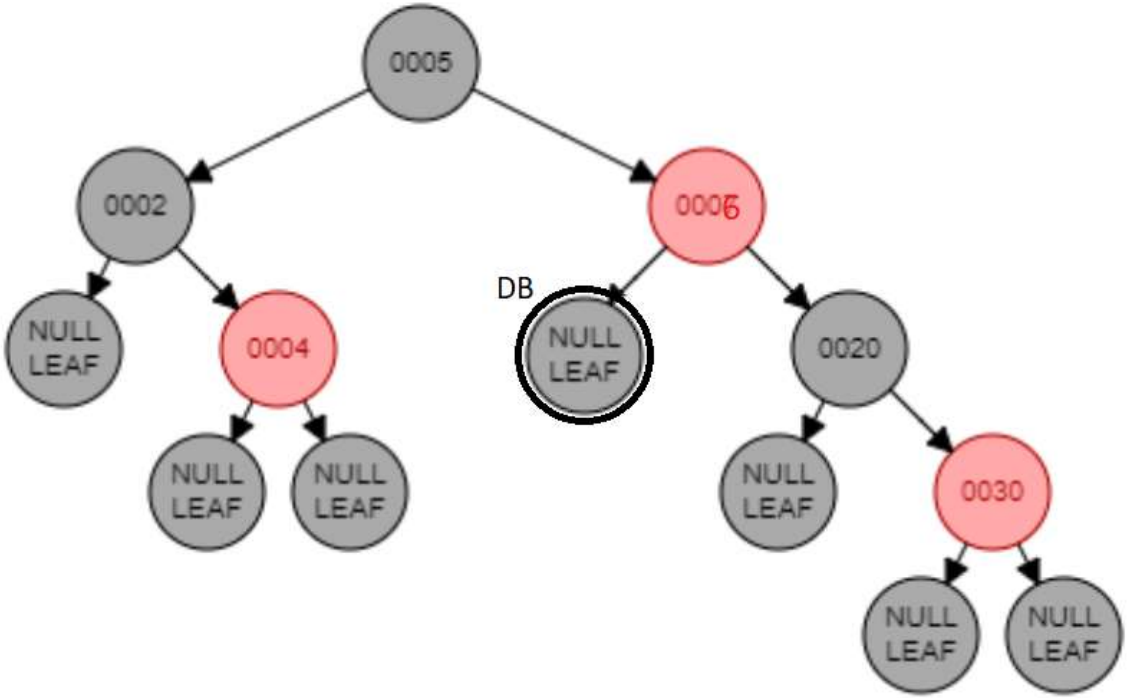
Step 2: Check its color first -> black

Go to step 3



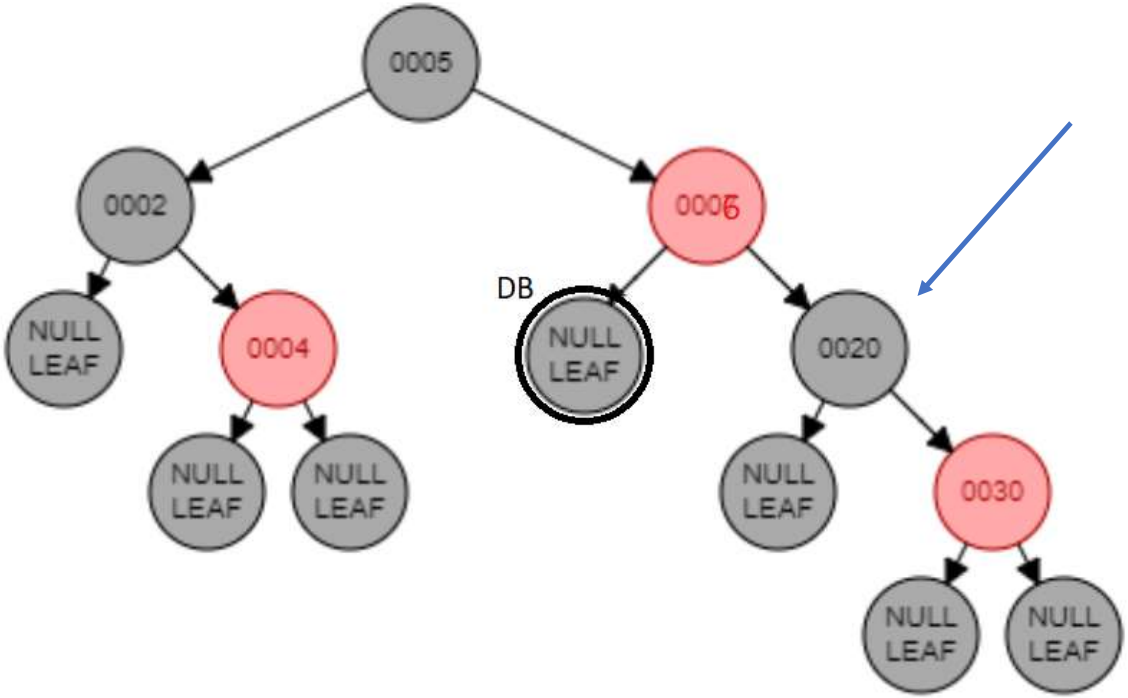
Delete 7

Step 3: Replace this node with its child (nil node) and its color became double black (DB)



Delete 7

Step 4: check its brother's color [20] -> black, go to step 5



Delete 7

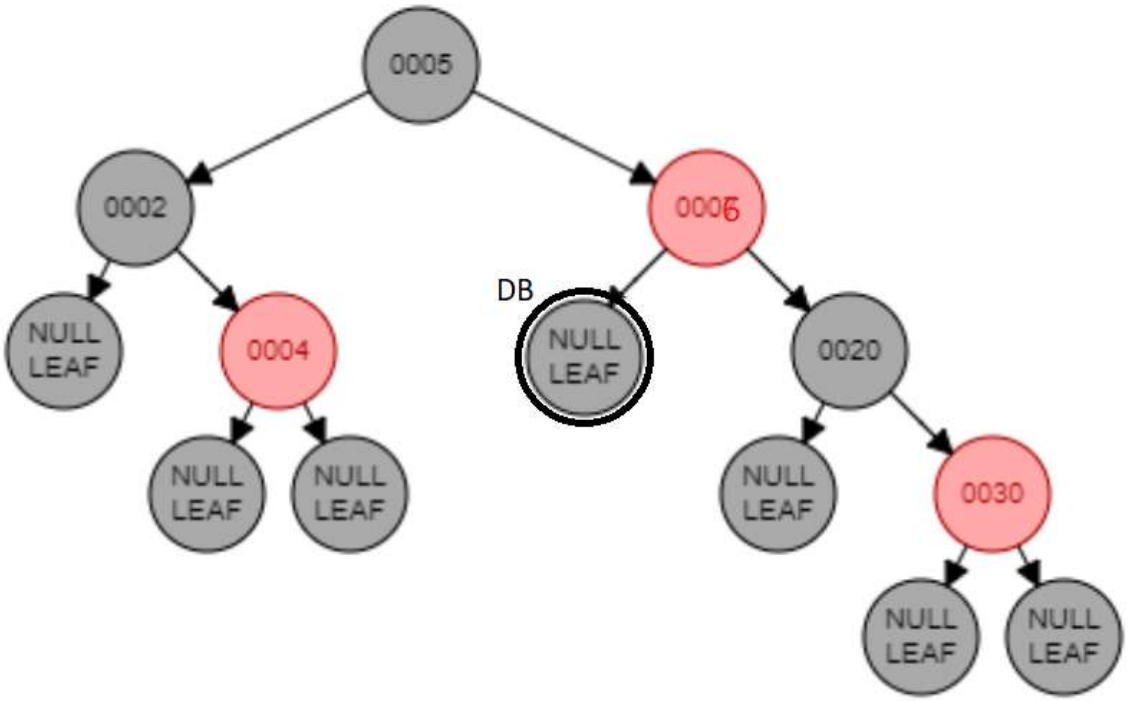
Step 5: Check your brother's children [colors]

- far child is RED -> Case 4

1)Rotation [parent (6) and brother (20)], then **DB will be removed.**

2)Then Recoloring:

- Parent (6) >> black & brother (20) >> color of the parent (RED) (as if they exchange their colors)
- brother's child (30) >> black



Delete 7

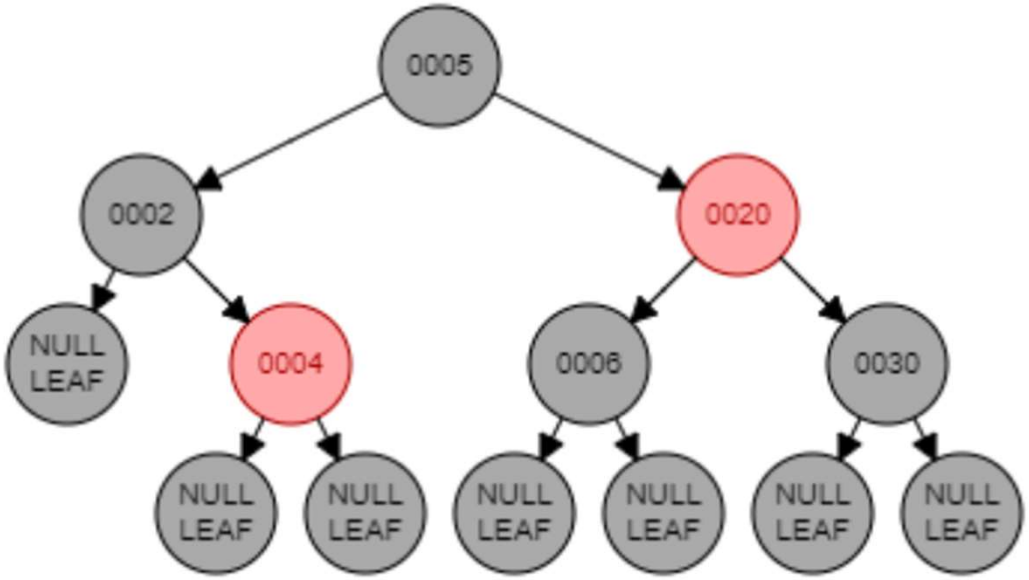
Step 5: Check your brother's children [colors]

- far child is RED -> Case 4

1)Rotation [parent (6) and brother (20)], then **DB will be removed.**

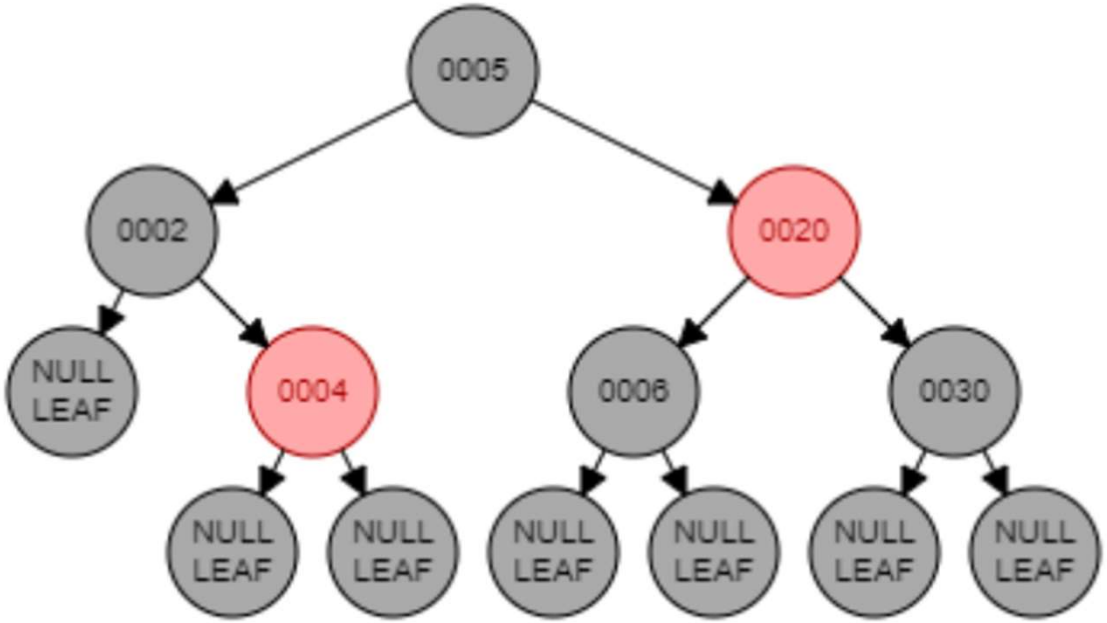
2)Then Recoloring:

- Parent (6) >> black & brother (20) >> color of the parent (RED) (as if they exchange their colors)
- brother's child (30) >> black



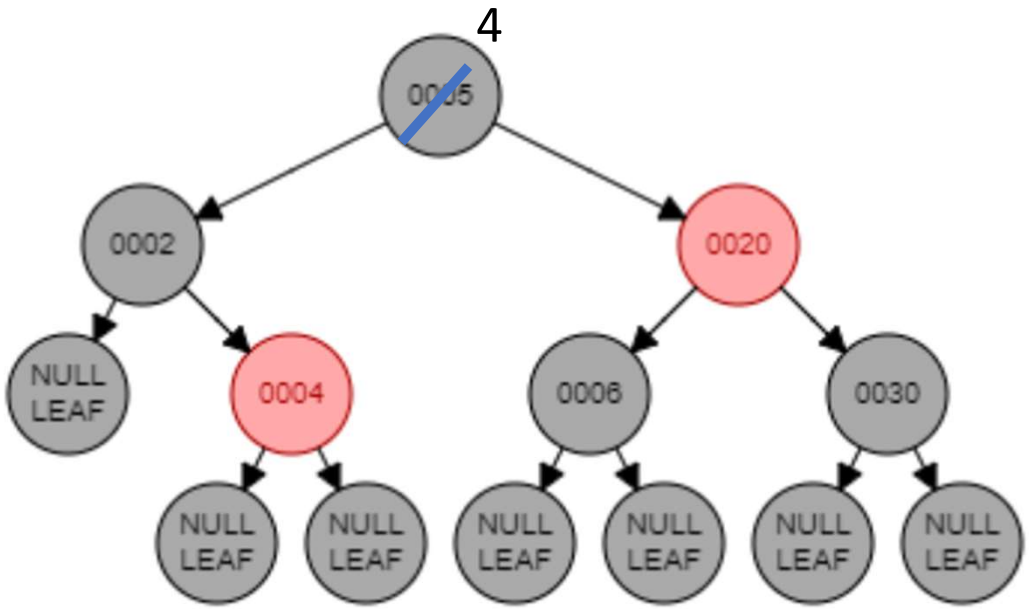
Delete 5

Step 1: Here 5 has 2 children, so first replace it with its predecessor or its successor, then delete this child

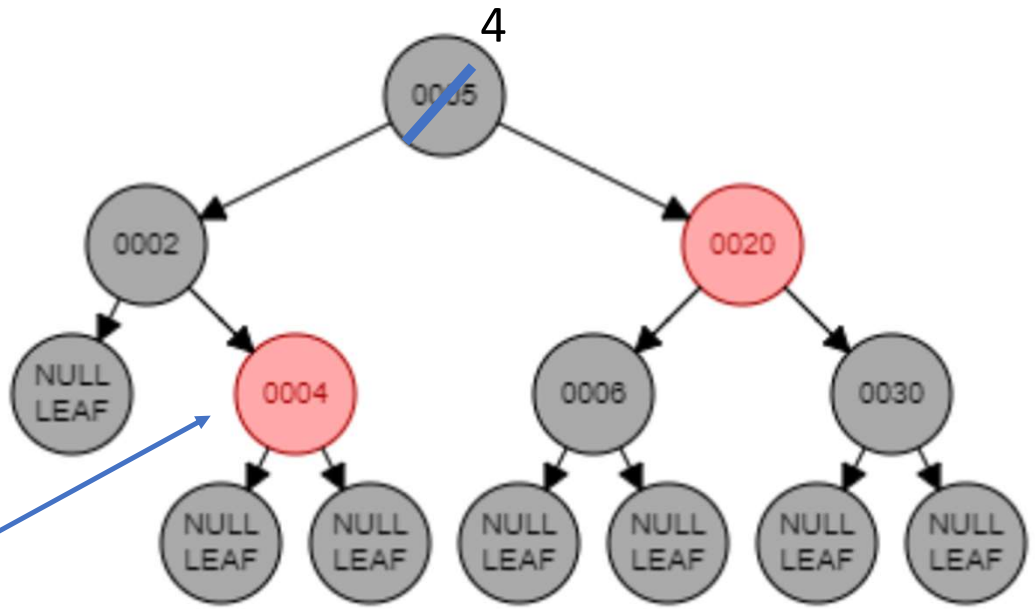


Delete 5

Step 1: Here 5 has 2 children, so first replace it with its predecessor or its successor, then delete this child [4]

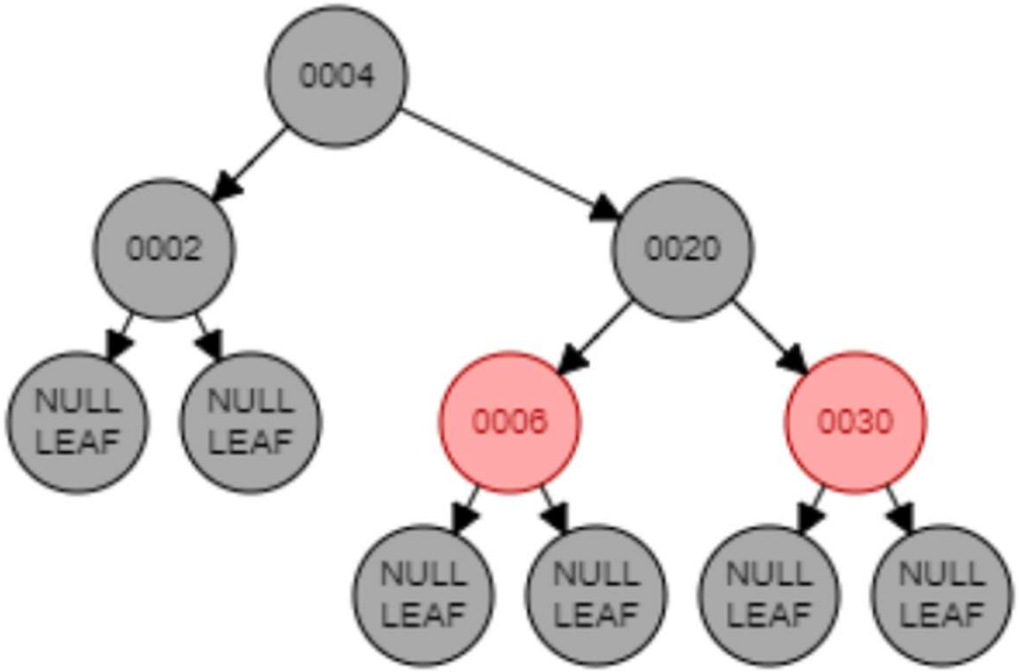


Delete 5



Step 2: Check its color first -> Red, just delete it

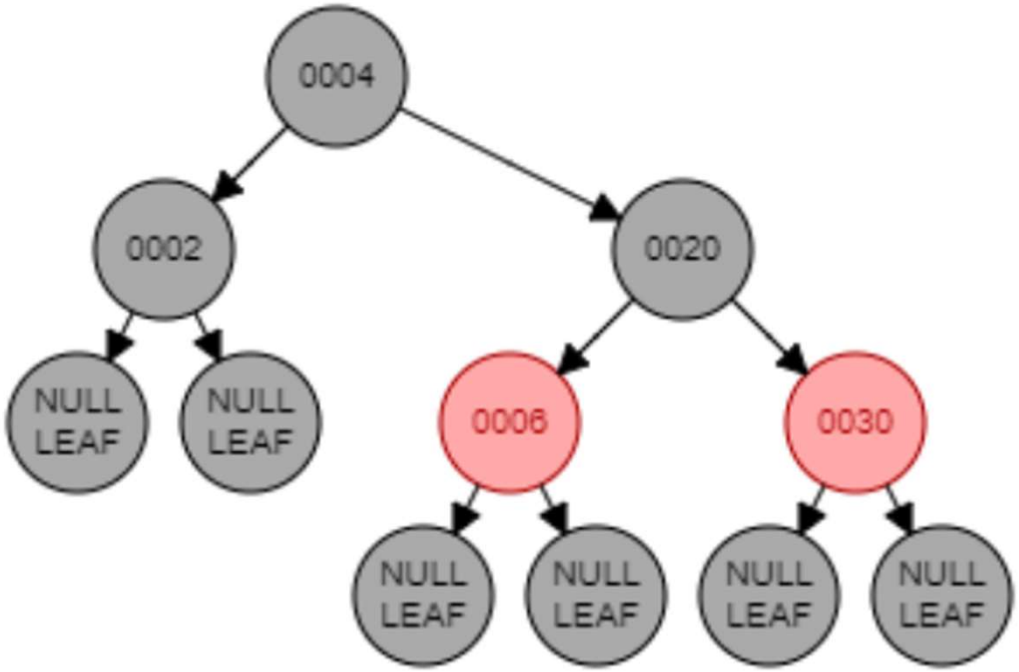
Delete 5


0004
0002
0020
NULL LEAF
NULL LEAF
0006
0030
NULL LEAF
NULL LEAF
NULL LEAF
NULL LEAF

Delete 2

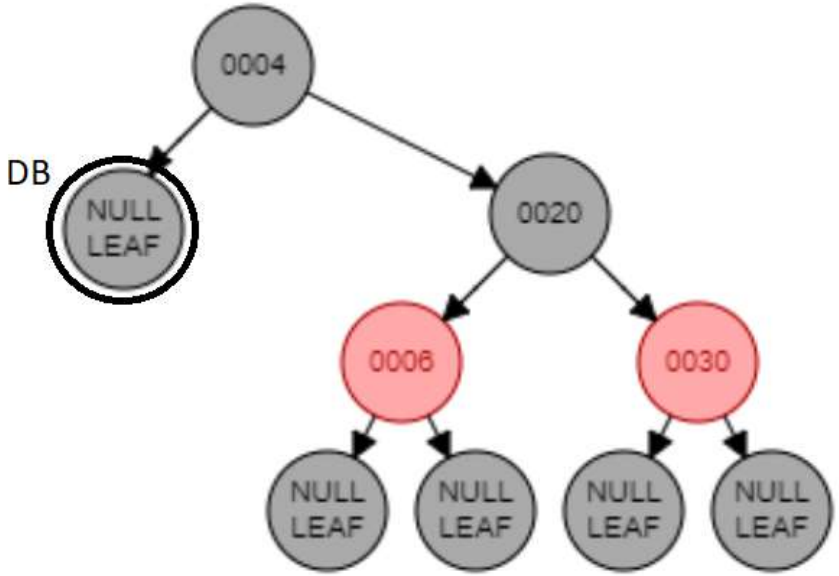
Step 1: Here 2 is a leaf node

Step 2: check its color first -> black then go to step 3



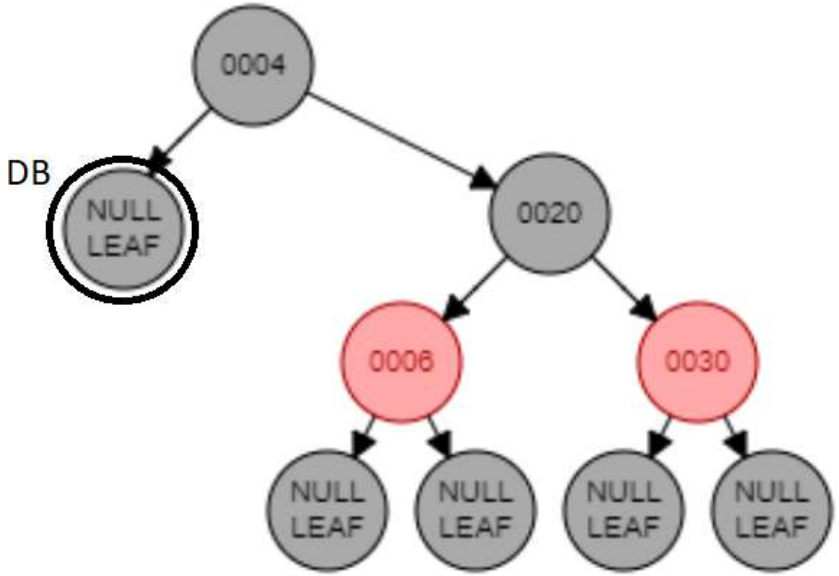
Delete 2

Step 3: Replace this node with its child (nil node) and its color became double black (DB)



Delete 2

Step 4: check its brother's color [20] -> black, go to step 5



Delete 2

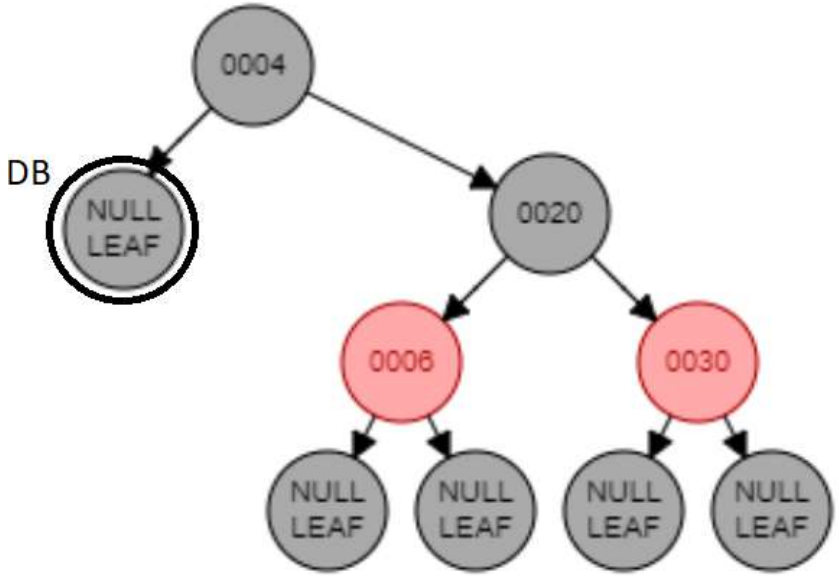
Step 5: Check your brother's children [colors]

- far child is RED (or both red) -> Case 4

1)Rotation [parent (4) and brother (20)], then **DB will be removed.**

2)Then Recoloring:

- Parent (4) >> black & brother (20) >> color of the parent (black)
- brother's child (30) >> black



Delete 2

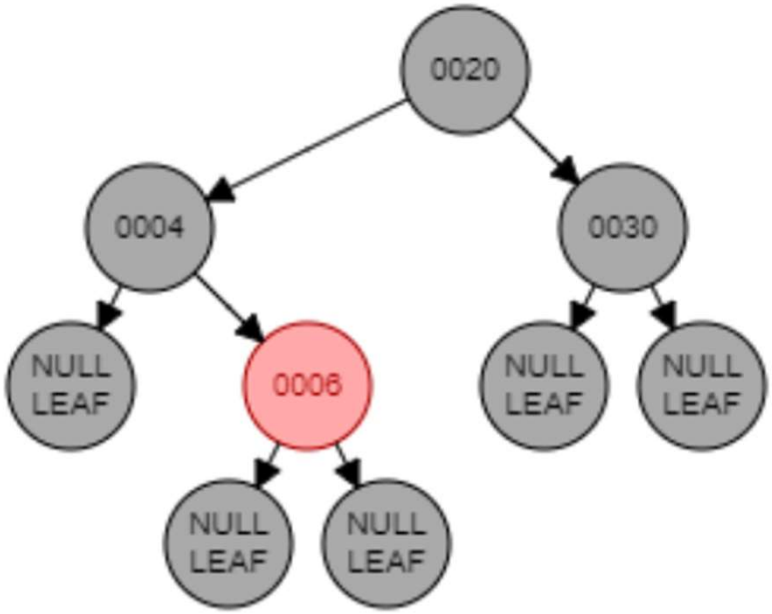
Step 5: Check your brother's children [colors]

- far child is RED (or both red) -> Case 4

1)Rotation [parent (4) and brother (20)], then **DB will be removed.**

2)Then Recoloring:

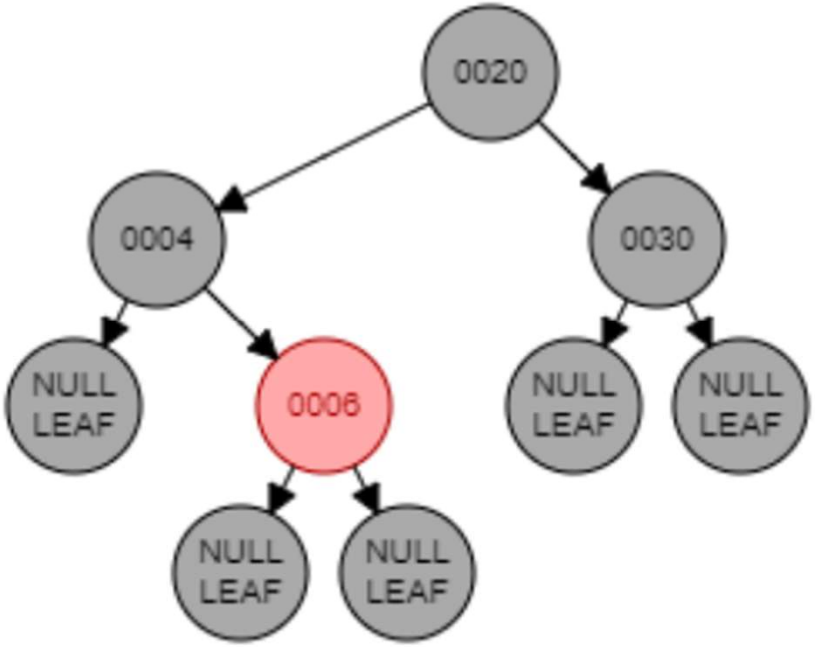
- Parent (4) >> black & brother (20) >> color of the parent (black)
- brother's child (30) >> black



Delete 30

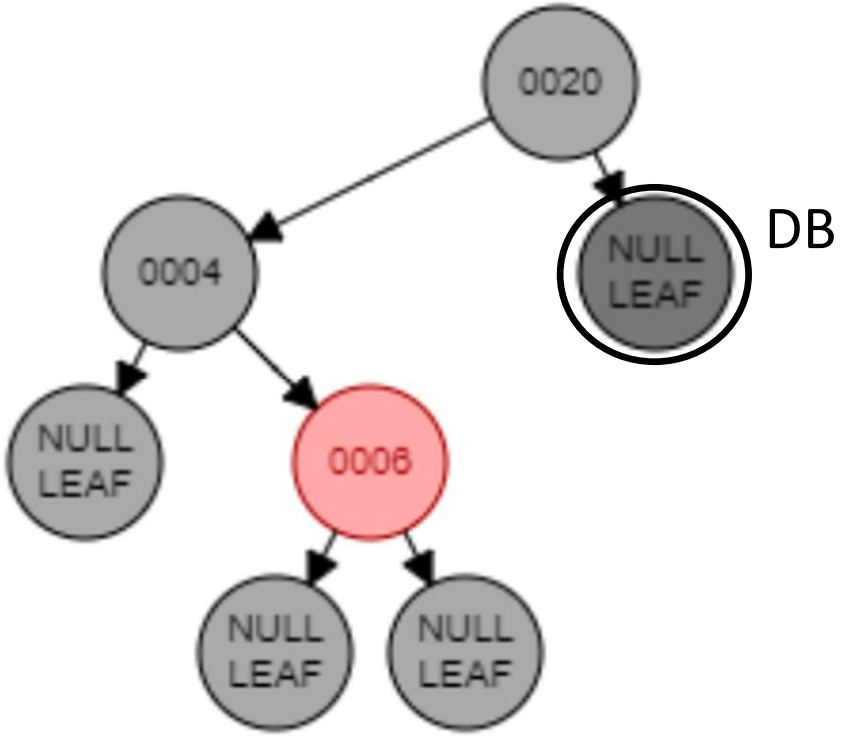
Step 1: Here 30 is a leaf node

Step 2: check its color first -> black then go to step 3



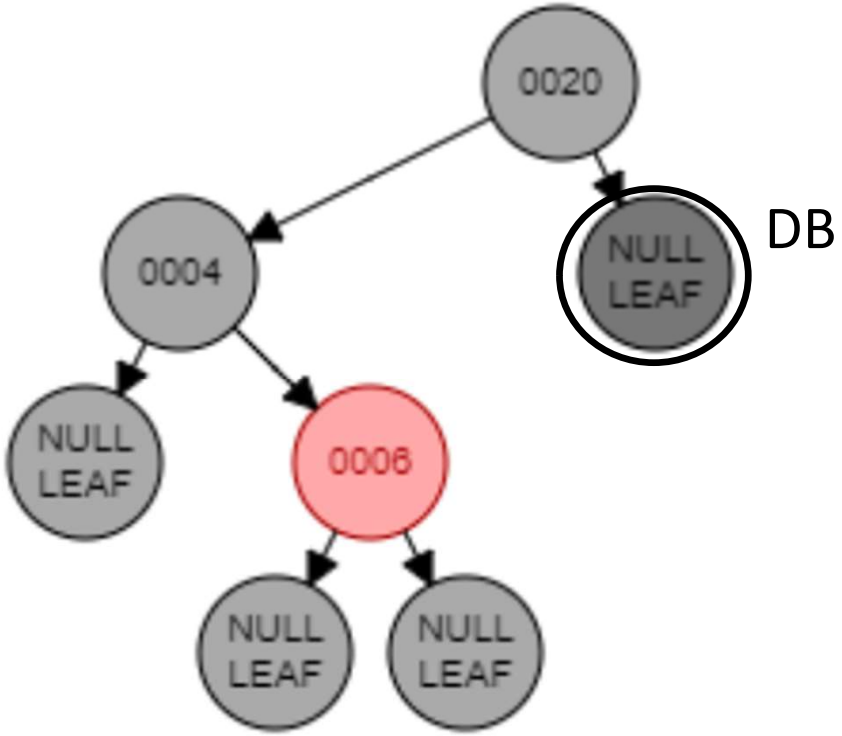
Delete 30

Step 3: Replace this node with its child (nil node) and its color became double black (DB)



Delete 30

Step 4: check its brother's color [4] -> black, go to step 5

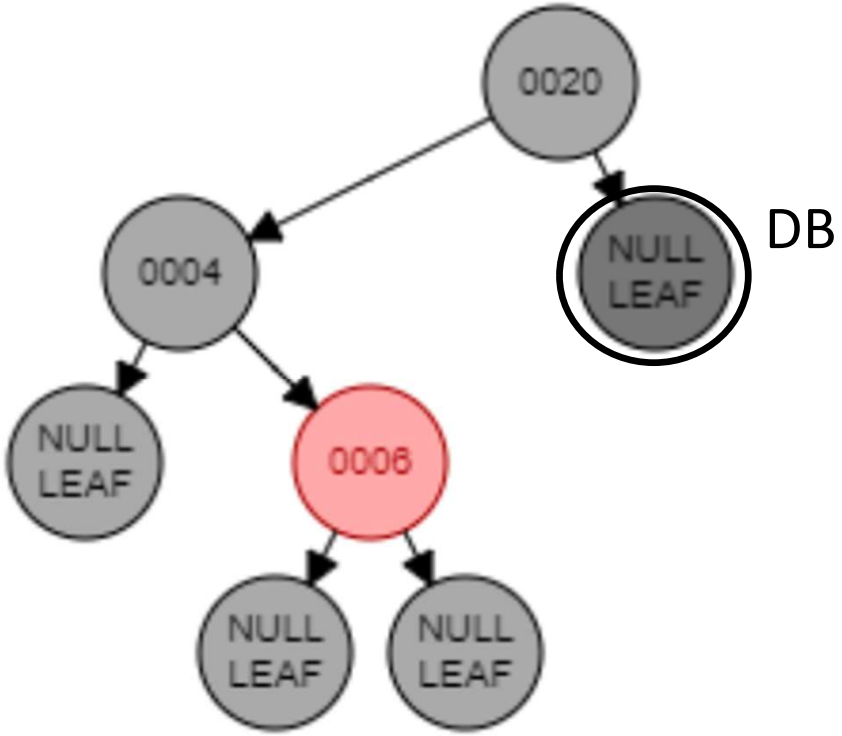


Delete 30

Step 5: Check your brother's children [colors]:

If the near child is Red [6], the other child black (nil) -> Case 3

Rotation and recoloring [brother (4) and its near child (6)], **then go to *Case 4***

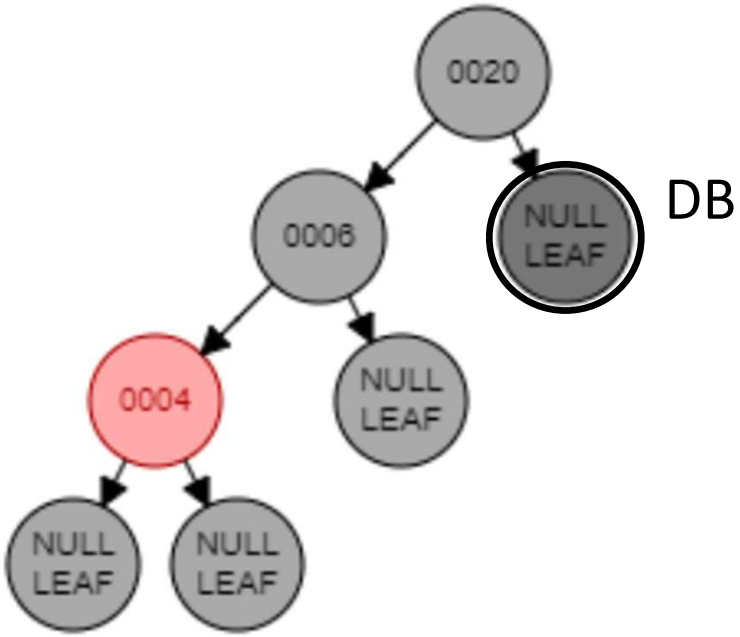


Delete 30

Step 5: Check your brother's children [colors]:

If the near child is Red [6], the other child black (nil) -> Case 3

Rotation and recoloring [brother (4) and its near child (6)], **then go to *Case 4***



Delete 30

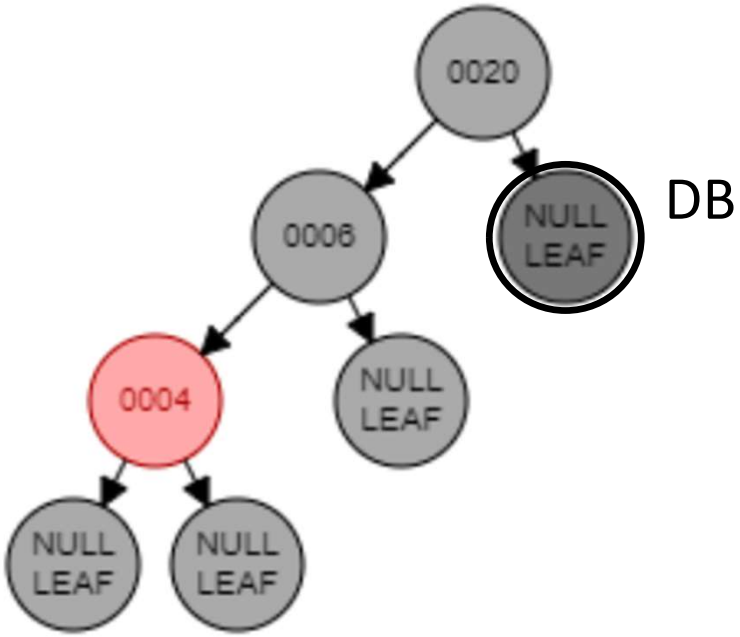
Step 5: Check your brother's children [colors]

- far child is RED (or both red) -> Case 4

1) Rotation [parent (20) and brother (6)], then
   DB will be removed.

2) Then Recoloring:

- Parent (20) >> black & brother (6) >> color of the parent (black)
- brother's child (4) >> black



Delete 30

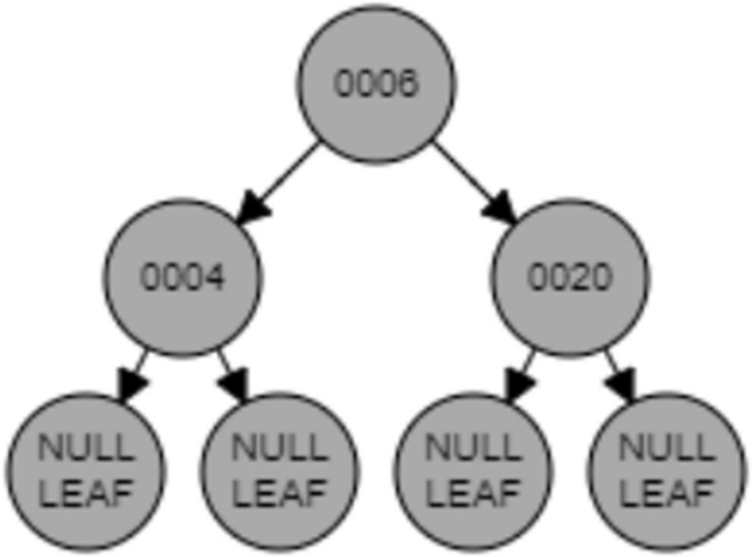
Step 5: Check your brother's children [colors]

- far child is RED (or both red) -> Case 4

1)Rotation [parent (20) and brother (6)], then **DB will be removed.**

2)Then Recoloring:

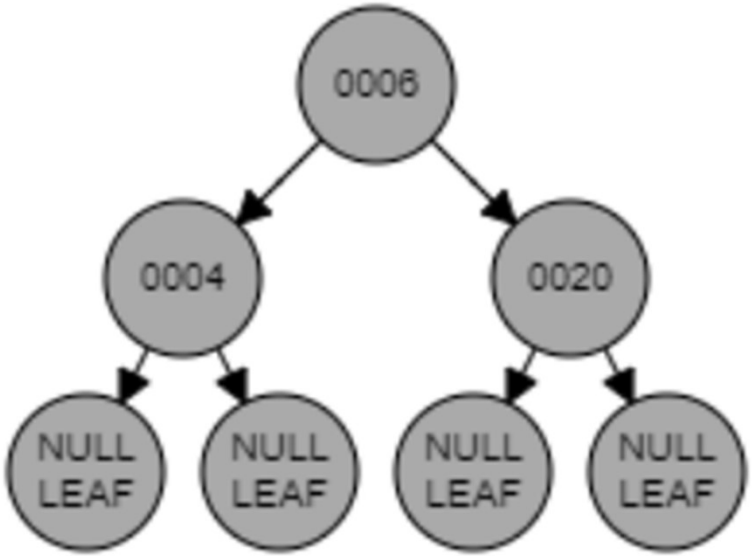
- Parent (20) >> black & brother (6) >> color of the parent (black)
- brother's child (4) >> black



Delete 4

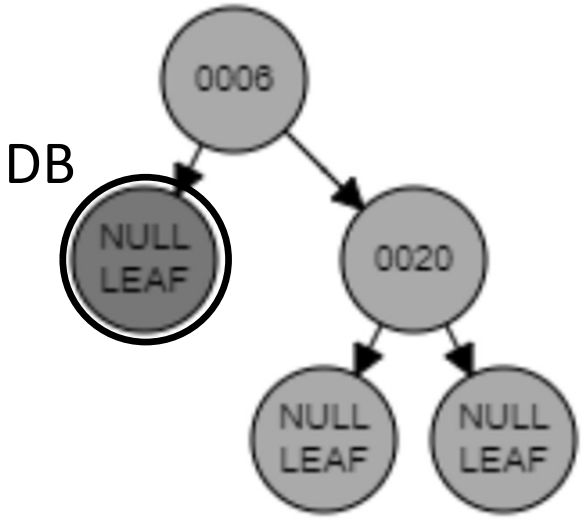
Step 1: Here 4 is a leaf node

Step 2: check its color first -> black then go to step 3



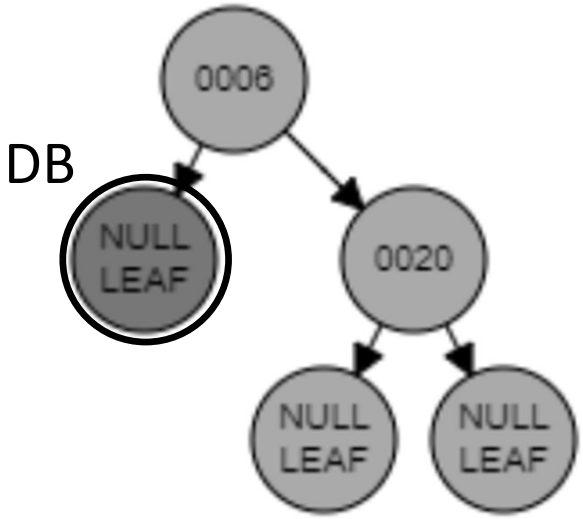
Delete 4

Step 3: Replace this node with its child (nil node) and its color became double black (DB)



Delete 4

Step 4: check its brother's color [20] -> black, go to step 5

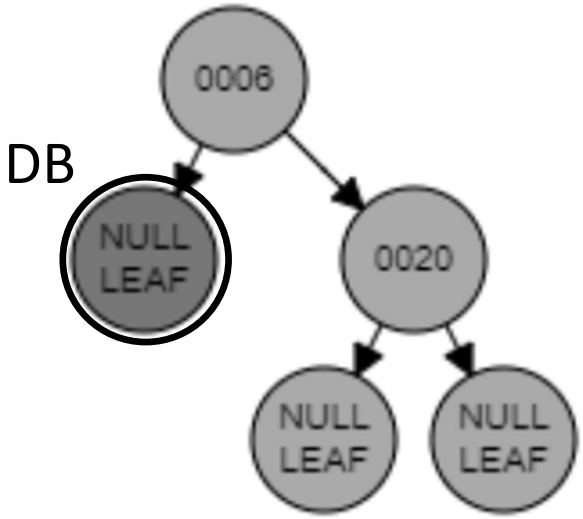


Delete 4

Step 5: Check your brother's children [colors]

- If the 2 children are black [nil & nil] -> Case 2
  1) Parent [6] will take the DB
  if red > black
  if black > DB
  2) Then recolor my brother [20] -> RED

  *Here [6] is the root -> so just remove the DB



Delete 4

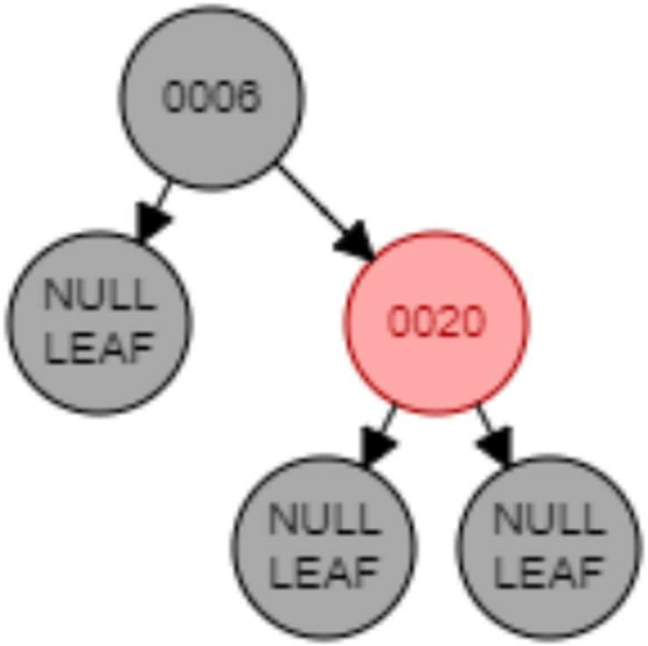
Step 5: Check your brother's children [colors]

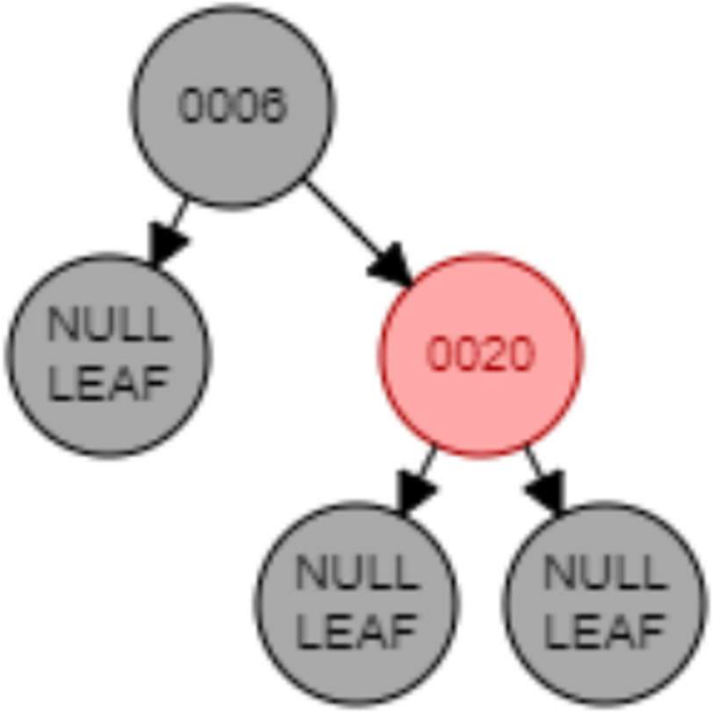
- If the 2 children are black [in the example: nil & nil]-> Case 2

1) Parent [6] will take the DB (if red > black, if black > DB)

2) Then recolor my brother [20] -> RED

- Here [6] is the root -> so just remove the DB




0006
NULL
LEAF
0020
NULL
LEAF
NULL
LEAF