Learn DAA: From B K Sharma

Insertion in Red-Black Tree

Learn DAA: From B K Sharma

RB-INSERT(T, z)

y ← NIL x ← root[T]	Initialize nodes x and y Throughout the algorithm y points to the parent of x

while x ≠ NIL
    y ← x
    if key[z] < key[x]
        then x ← left[x]
    else x ← right[x]
	Go down the tree until reaching a leaf
	At that point y is the parent of the node to be inserted

p[z] ← y	Sets the parent of z to be y
if y = NIL then root[T] ← z	The tree was empty: set the new node to be the root
else if key[z] < key[y] then left[y] ← z else right[y] ← z	Otherwise, set z to be the left or right child of y, depending on whether the inserted node is smaller or larger than y's key
left[z] ← NIL right[z] ← NIL color[z] ← RED	Set the fields of the newly added node
RB-INSERT-FIXUP(T, z)	Fix any inconsistencies that could have been introduced by adding this new red node

RB-INSERT-FIXUP(T, z)

1. **while** color[p[z]] = RED
2. **do if** p[z] = left[p[p[z]]]
3. **then** y ← right[p[p[z]]]
4. **if** color[y] = RED
5. **then Case1**
6. **else if** z = right[p[z]]
7. **then Case2**
8. **Case3**
9. **else** (same as **then** clause with "right" and "left" exchanged)
10. color[root[T]] ← BLACK

The while loop repeats only when case1 is executed: O(lgn) times

Set the value of x's "uncle"

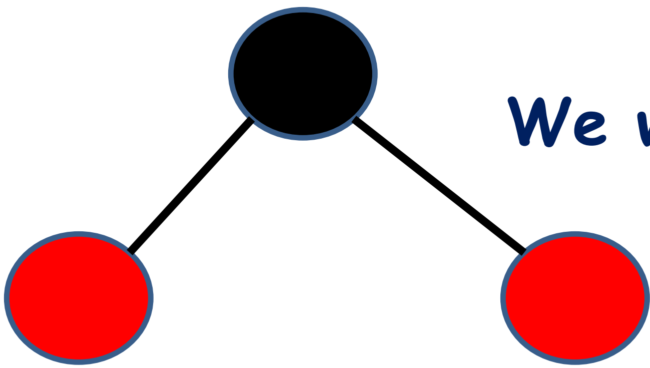
We just inserted the root, or The red violation reached the

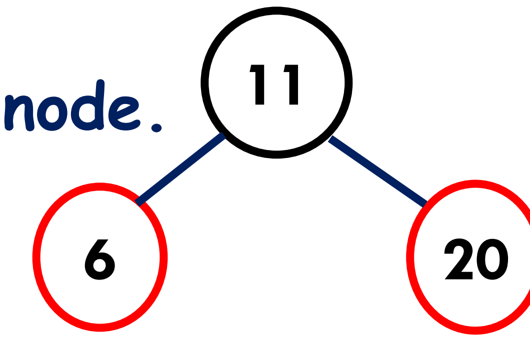
Insertion in Red-Back Tree

Let a Red-Black is given.

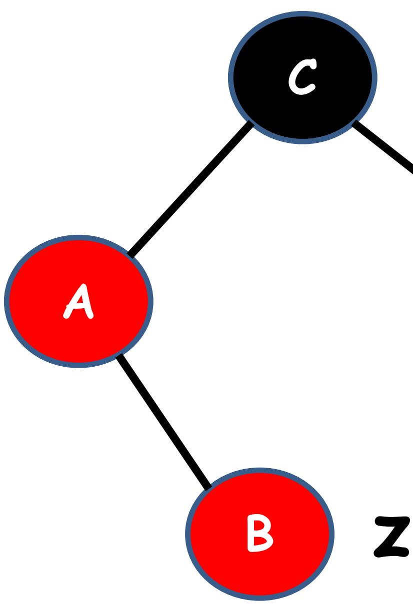
And

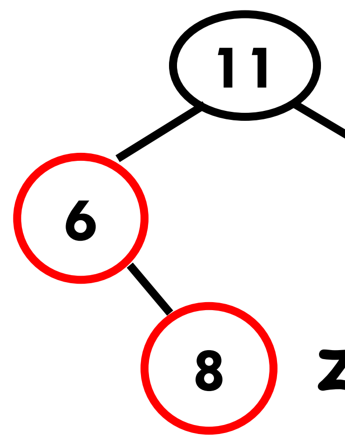
We want to insert a new node.





The color of new node(z) is assumed as red.(Why?)





Learn DAA: From B K Sharma

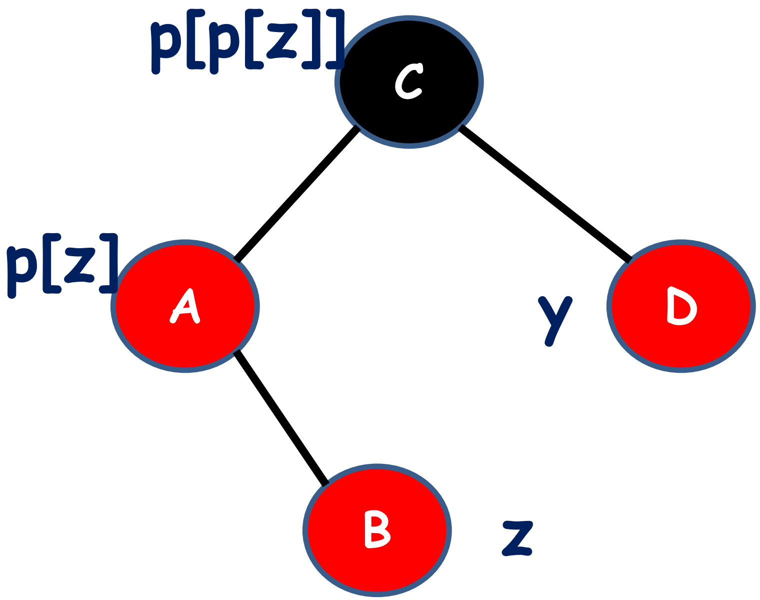
Insertion in Red-Back Tree

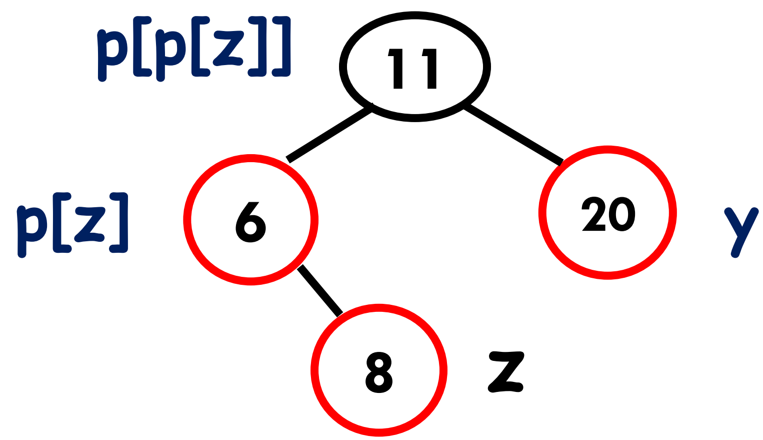
The color of new node(z) is red.
The the color of p[z] =RED

And

z is right child of p[z]
z's uncle y is RED

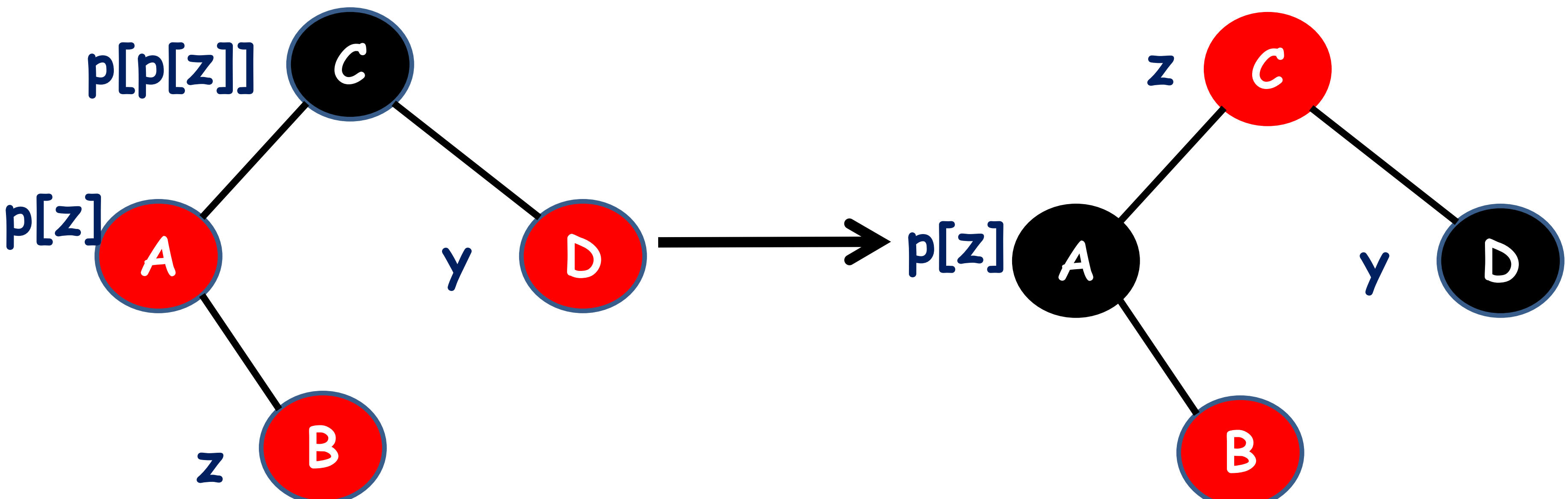
Case 1(A)





Insertion in Red-Back Tree
Case 1(A)

Solution:	Recolor
	color[p[z]]←BLACK
	color[y] ←BLACK
	color[p[p[z]] ←RED
	z=p[p[z]]

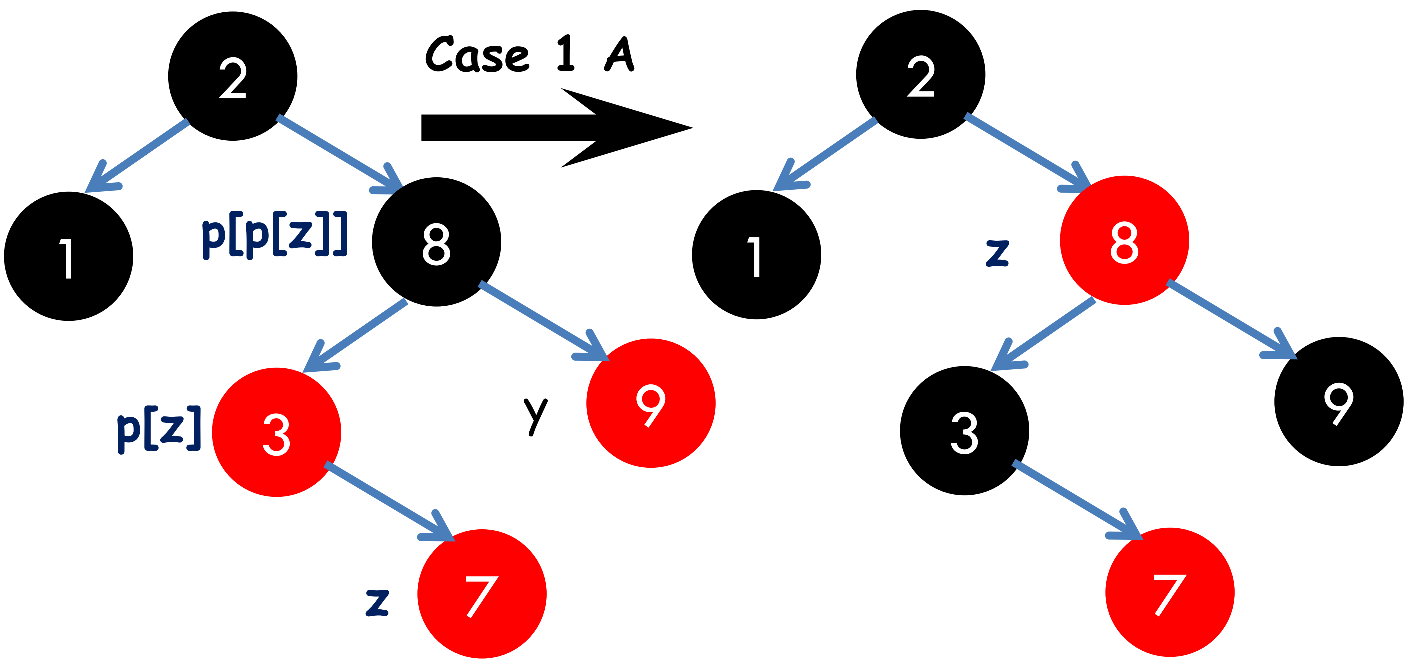


The case 1 pushes the RED-RED violation up the tree

Insertion in Red-Back Tree

Case 1(A)

Solution: Recolor
color[p[z]]←BLACK
color[y] ←BLACK
color[p[p[z]] ←RED
z=p[p[z]]



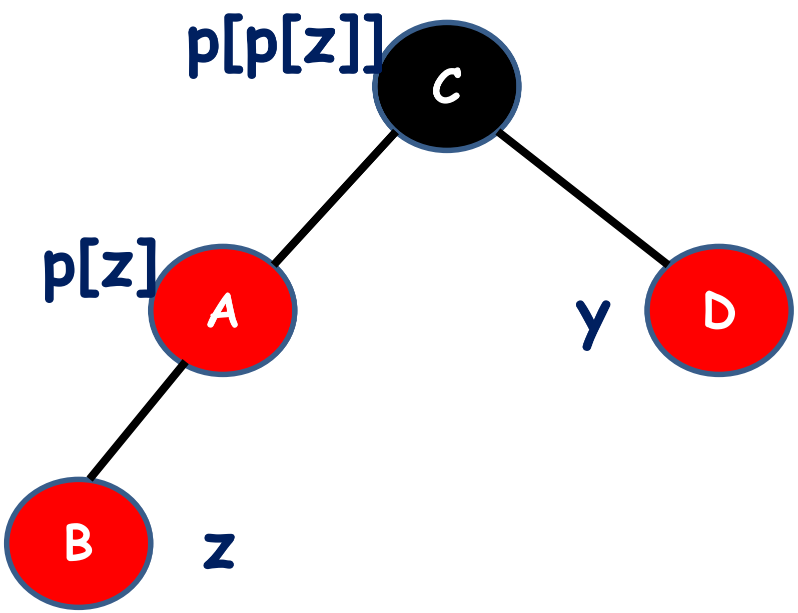
Insertion in Red-Back Tree

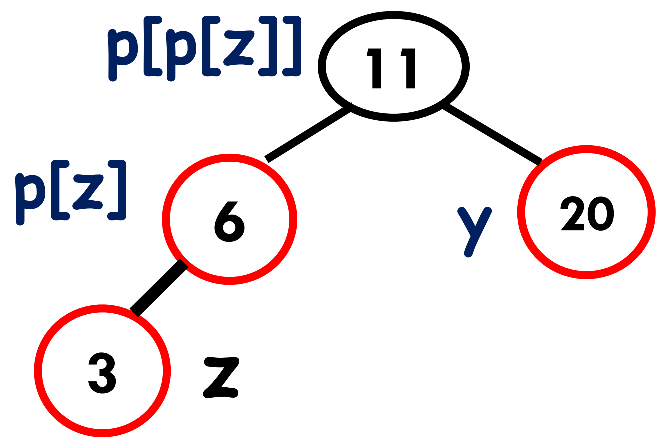
The color of new node(z) is red.
The the color of p[z] =RED

And

z is left child of p[z]
z's uncle y is RED

Case 1(B)

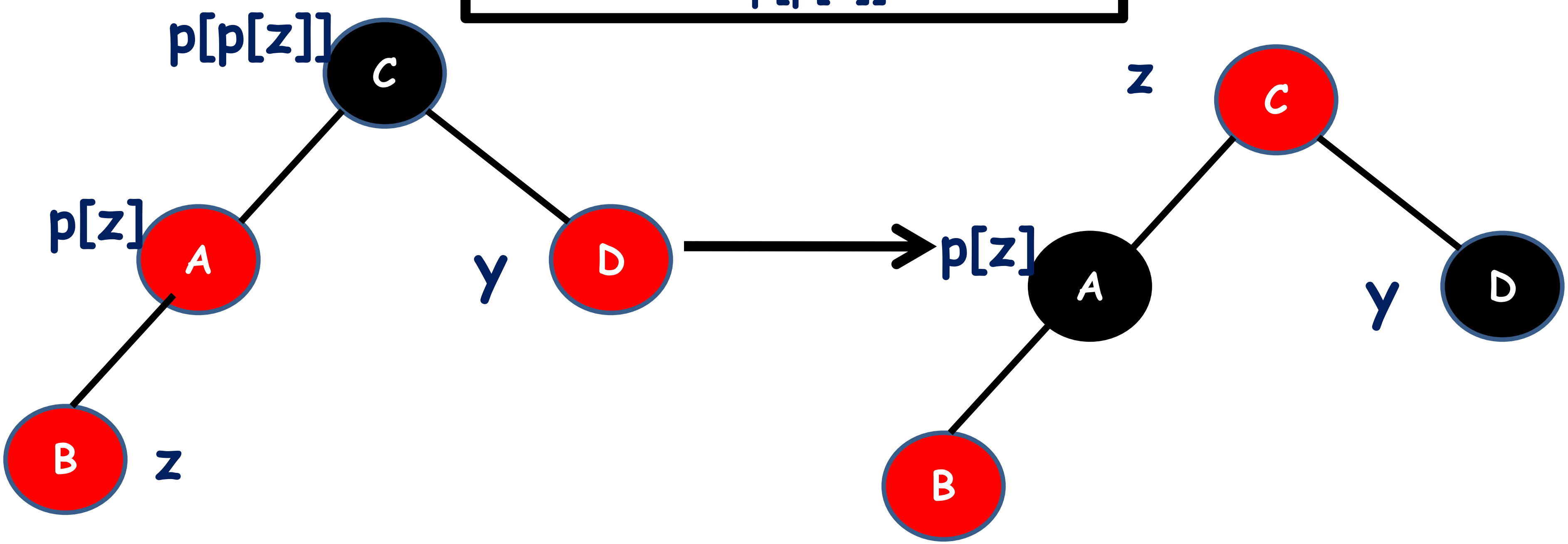




Insertion in Red-Back Tree

Case 1(B)

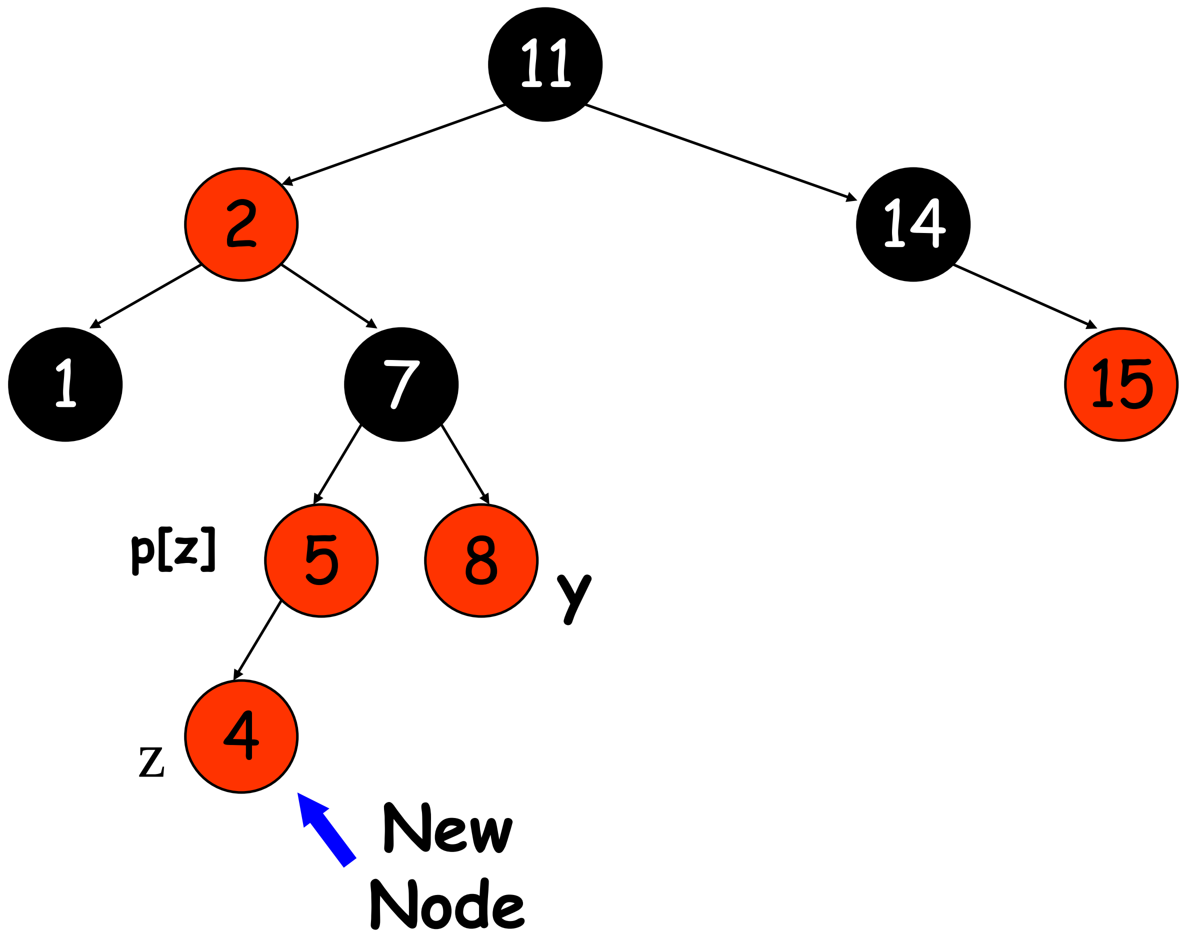
Solution: Recolor
color[p[z]] ← BLACK
color[y] ← BLACK
color[p[p[z]] ← RED
z=p[p[z]]



The case 1 pushes the RED-RED violation up the tree

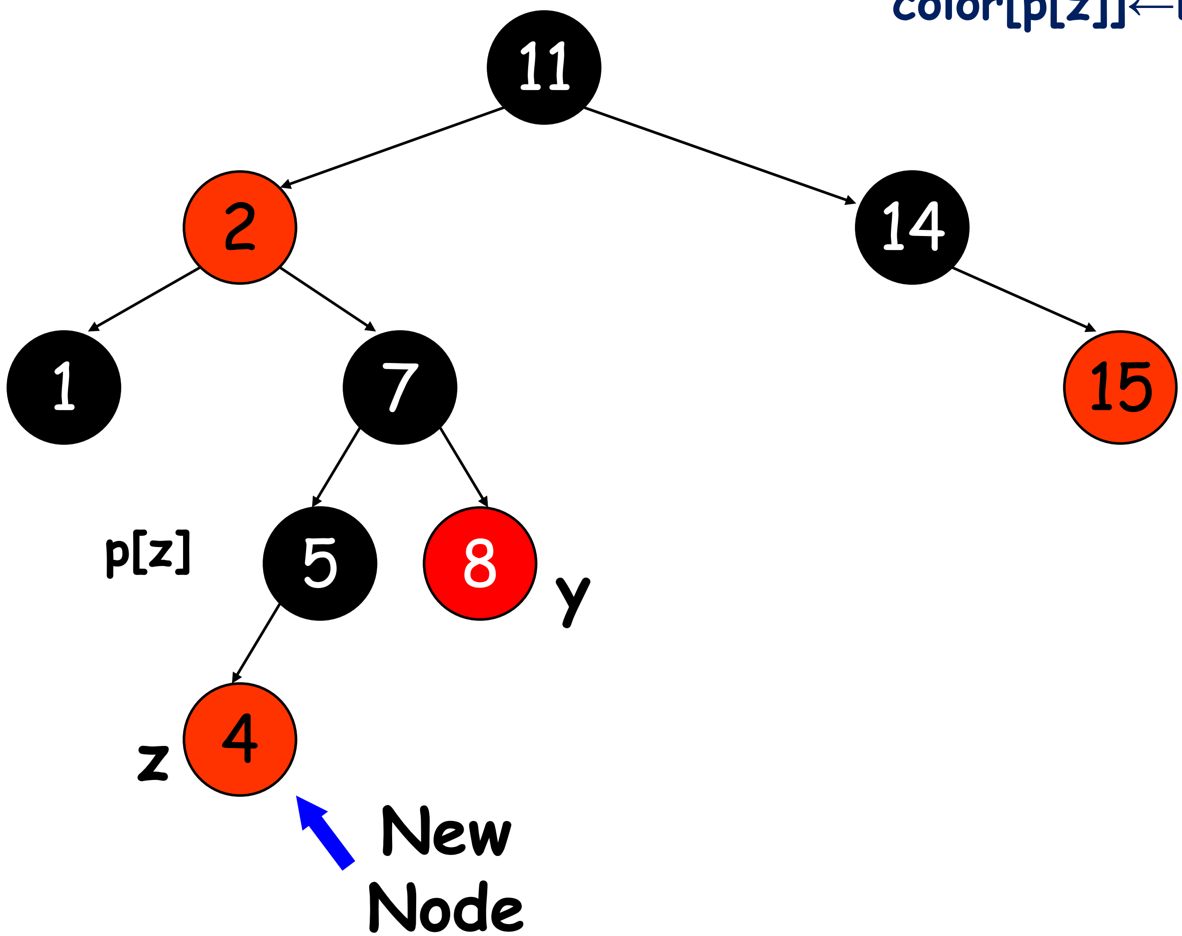
Learn DAA: From B K Sharma

Insertion in Red-Back Tree
Case 1(B)



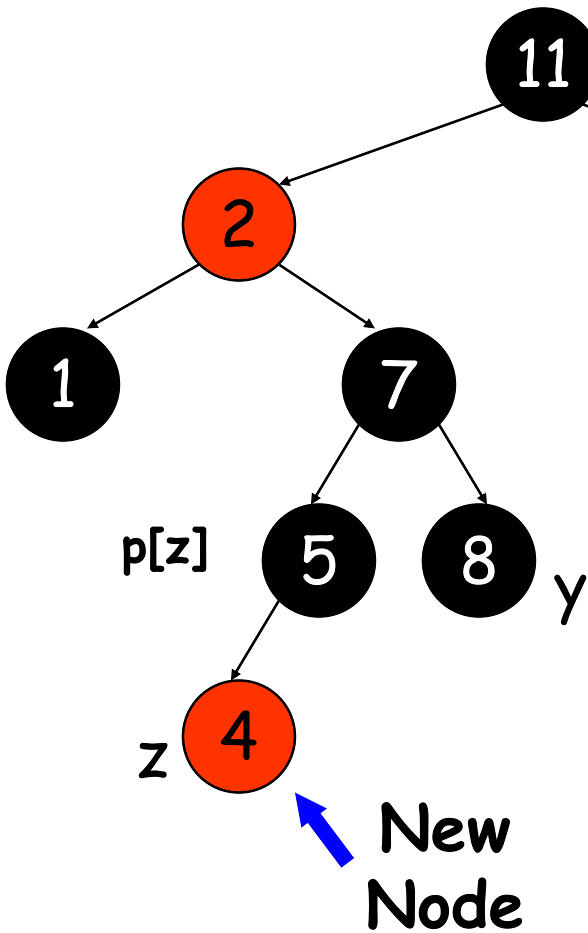
Insertion in Red-Back Tree
Case 1(B)

color[p[z]]←BLACK



Insertion in Red-Back Tree
Case 1(B)

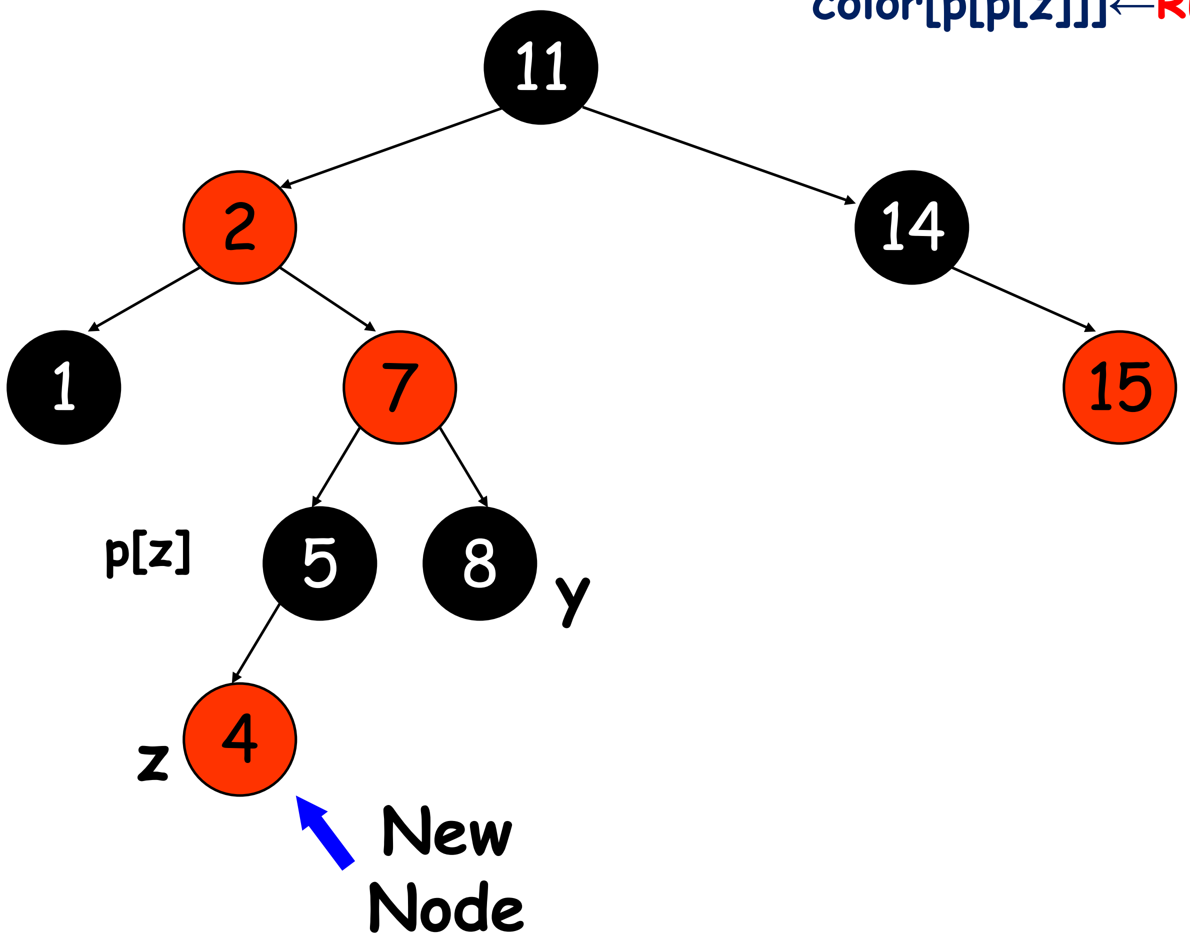
color[y] ←BLACK



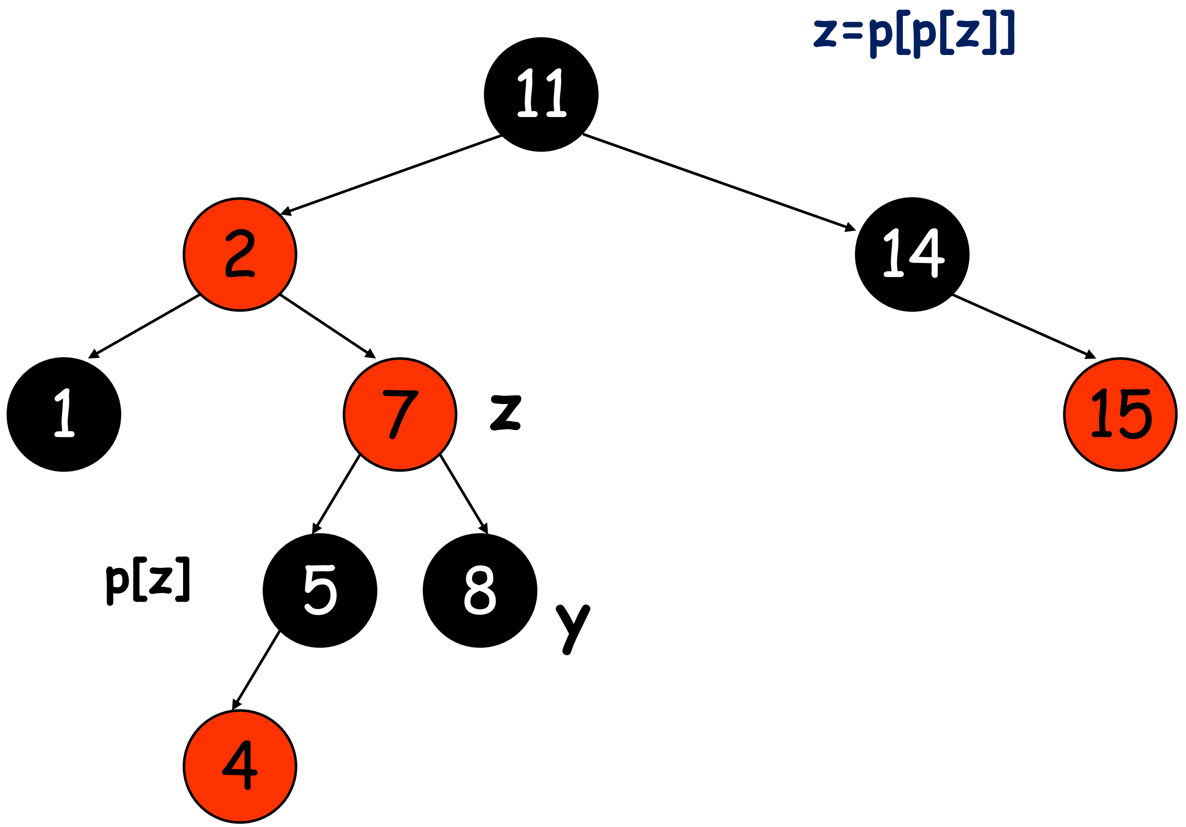
Learn DAA: From B K Sharma

Insertion in Red-Back Tree
Case 1(B)

color[p[p[z]]]←RED



Insertion in Red-Back Tree
Case 1(B)



The case 1 pushes the RED-RED violation up the tree

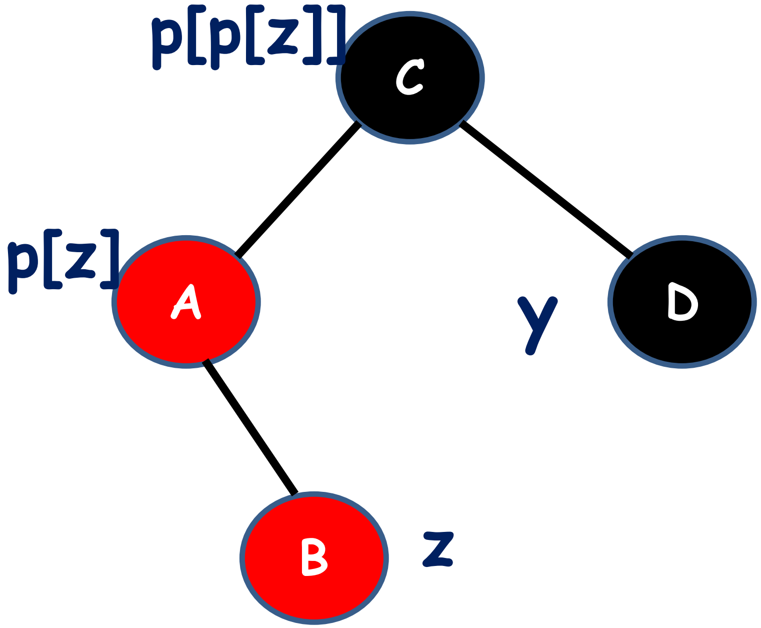
Insertion in Red-Back Tree

The color of new node(z) is red.
The the color of p[z] =RED

And

z is right child of p[z]
z's uncle y is **BLACK**

Case 2(A)



Learn DAA: From B K Sharma

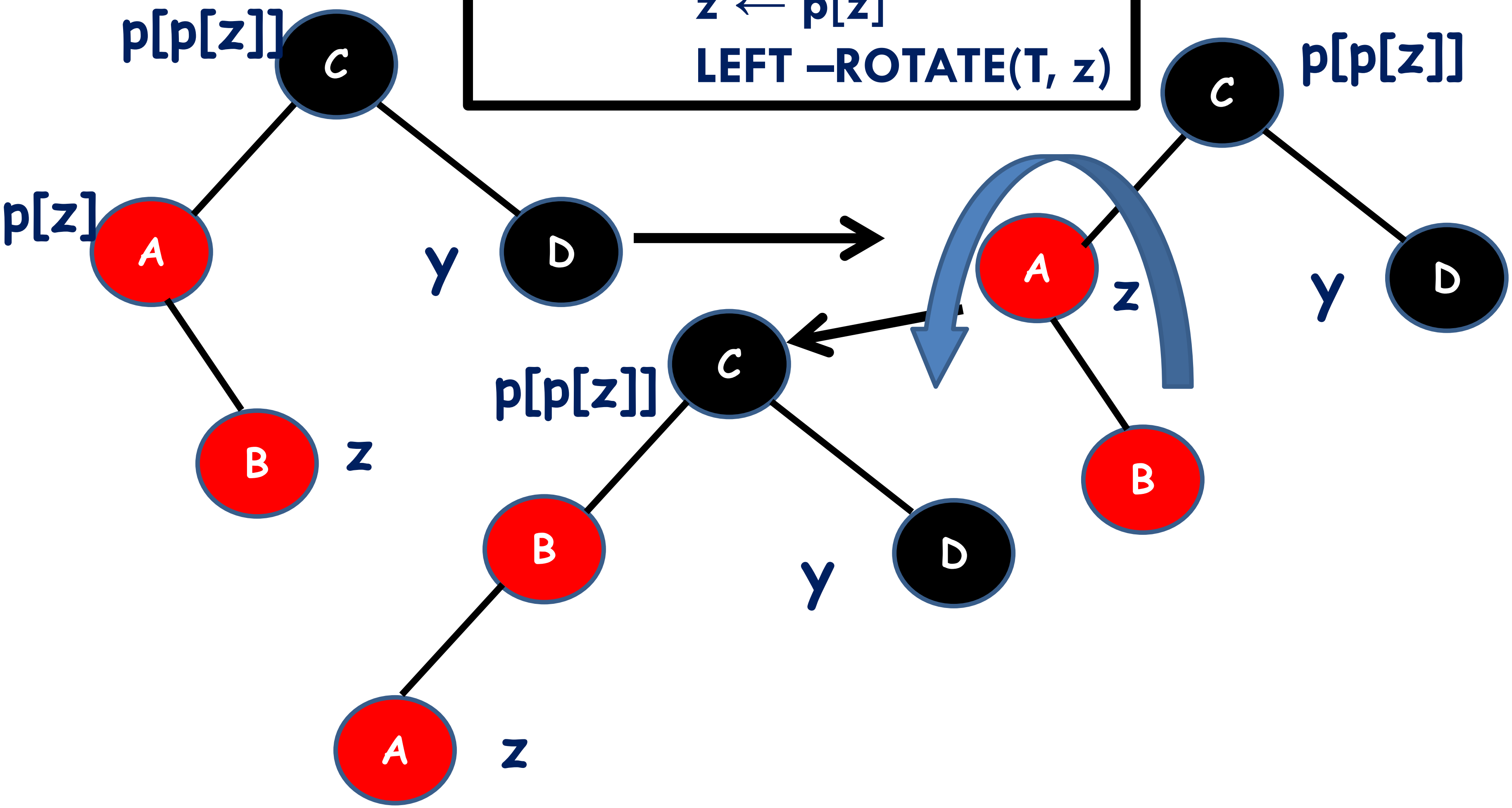
Insertion in Red-Back Tree

Case 2(A)

Solution: Left-Rotation

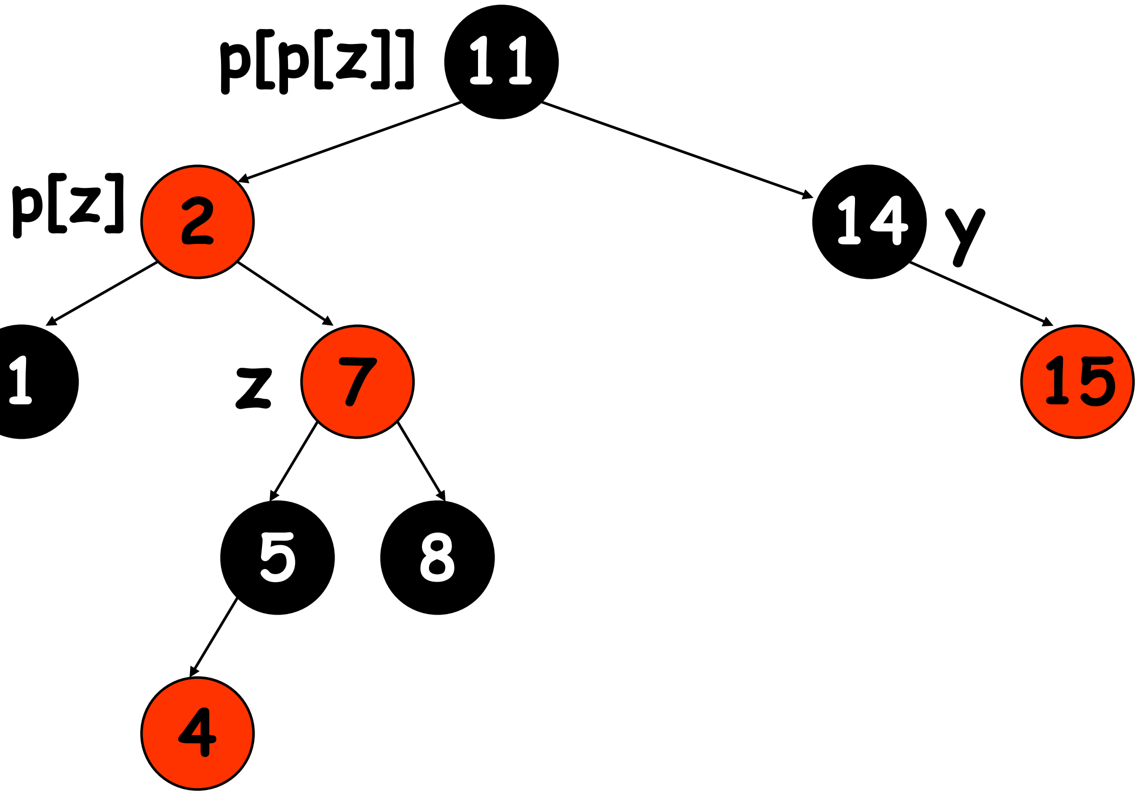
$z \leftarrow p[z]$

LEFT –ROTATE(T, z)



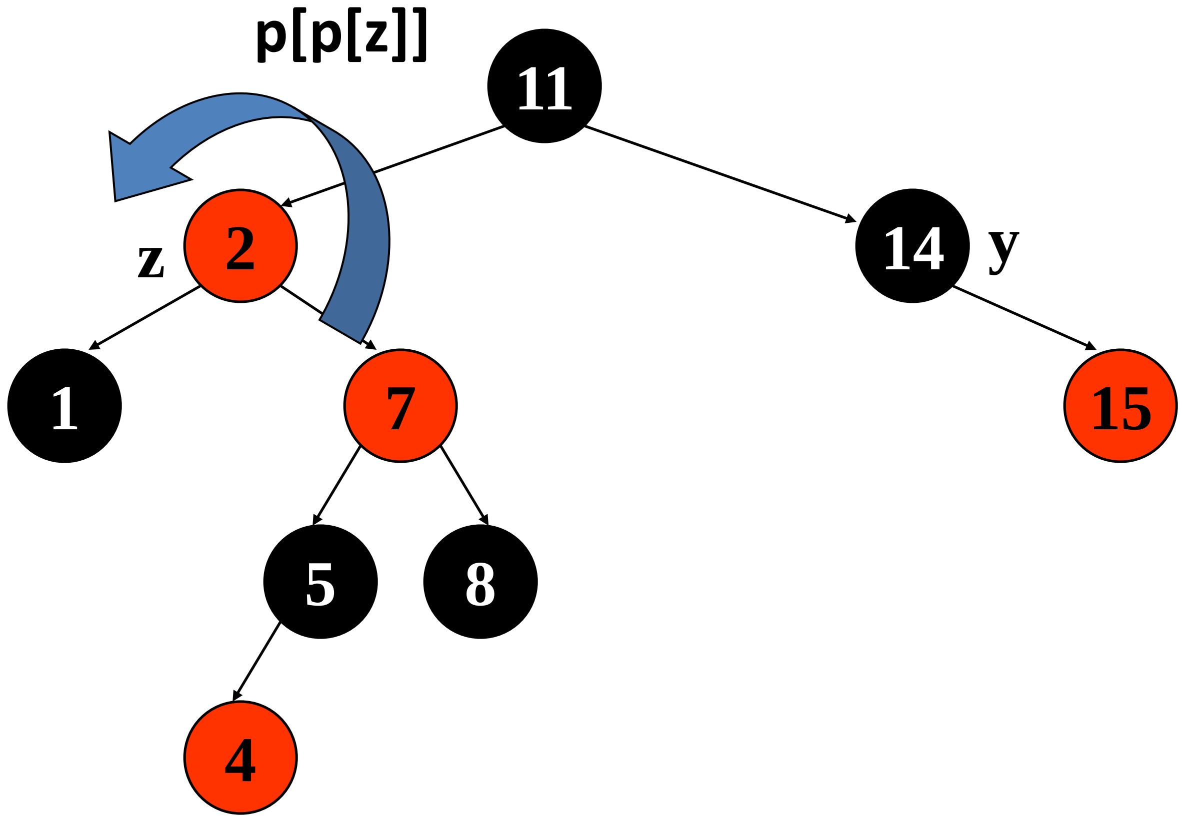
Learn DAA: From B K Sharma

Insertion in Red-Back Tree
Case 2(A)

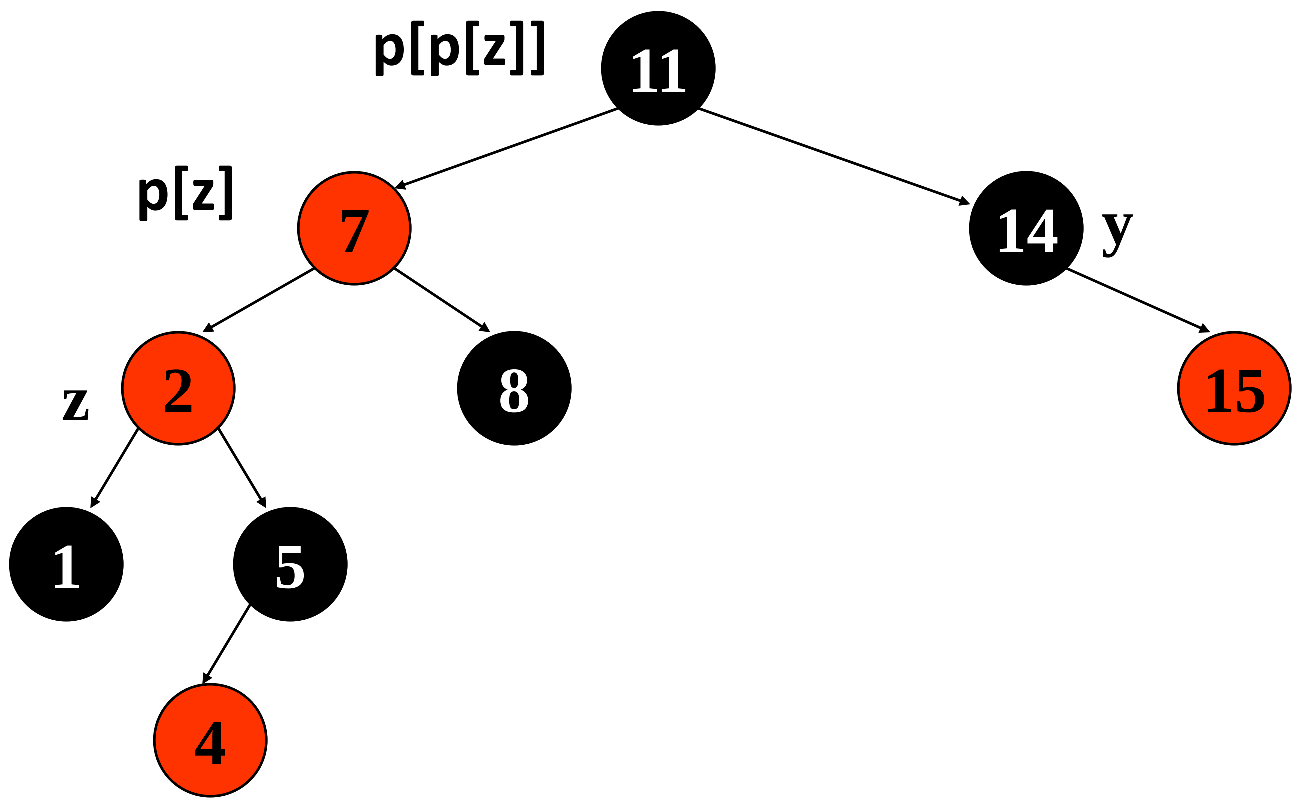


Learn DAA: From B K Sharma

Insertion in Red-Back Tree
Case 2(A)



Insertion in Red-Back Tree
Case 2(A)



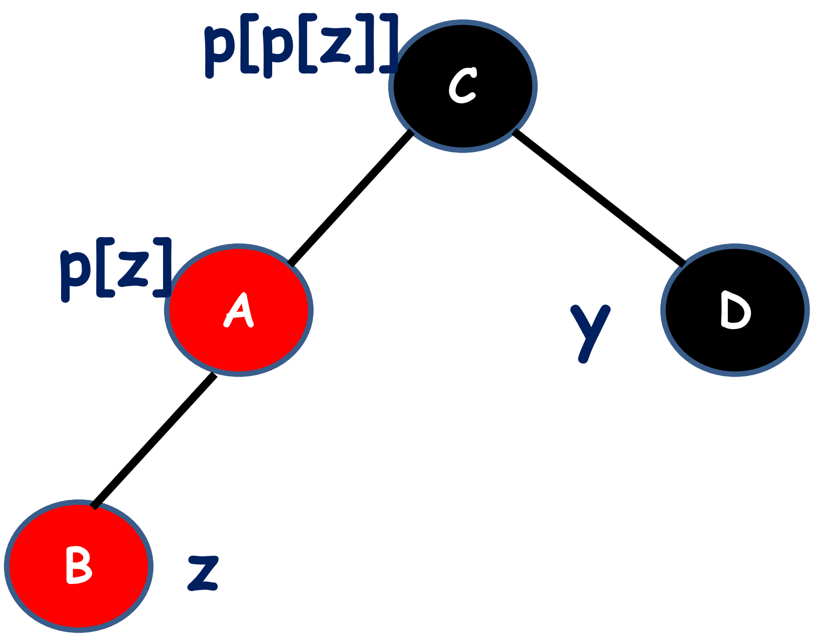
Insertion in Red-Back Tree

The color of new node(z) is red.
The the color of p[z] =RED

And

z is left child of p[z]
z's uncle y is BLACK

Case 2(B)



Learn DAA: From B K Sharma

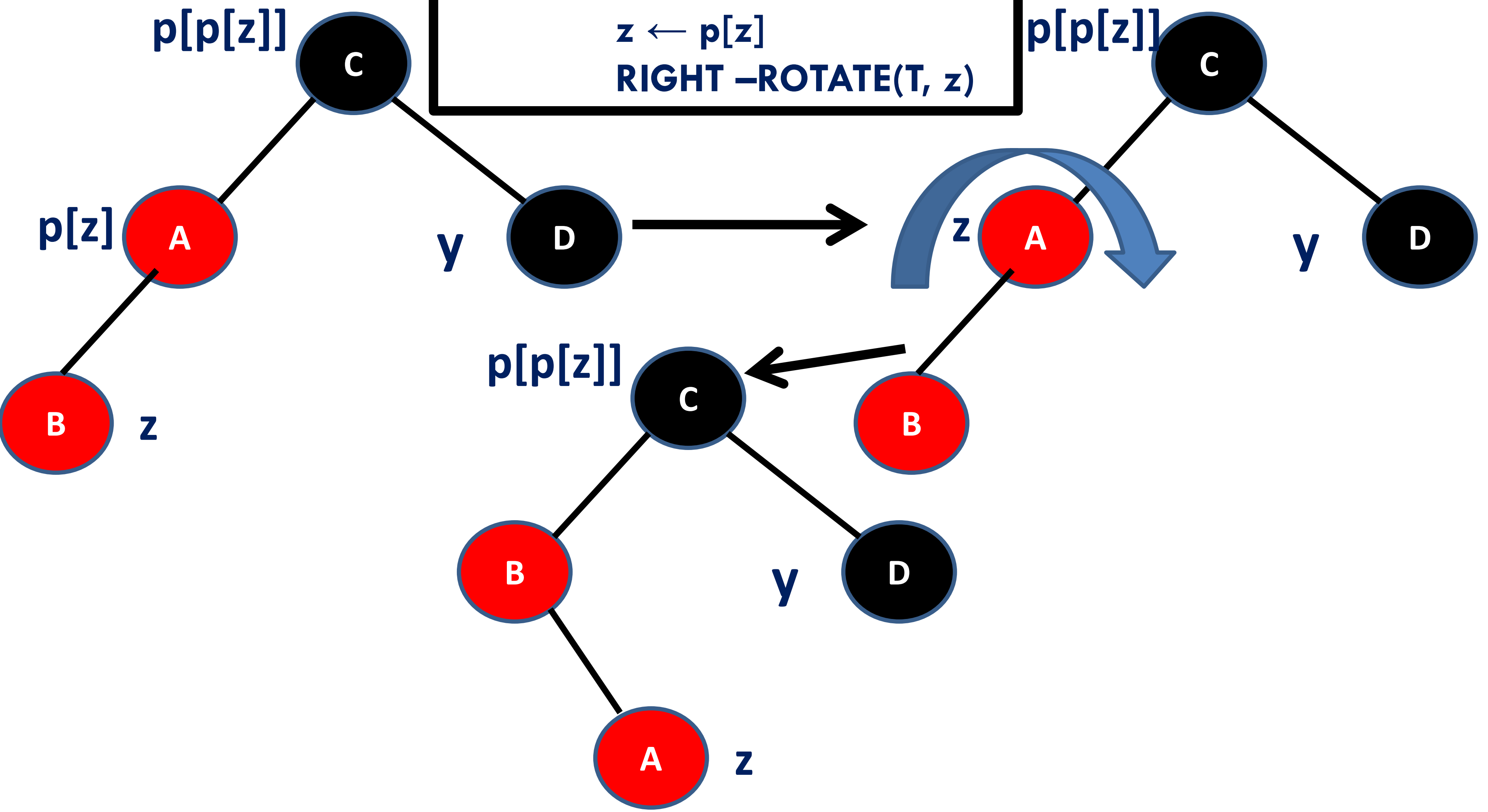
Insertion in Red-Back Tree

Case 2(B)

Solution: Right Rotation

$z \leftarrow p[z]$

RIGHT –ROTATE(T, z)



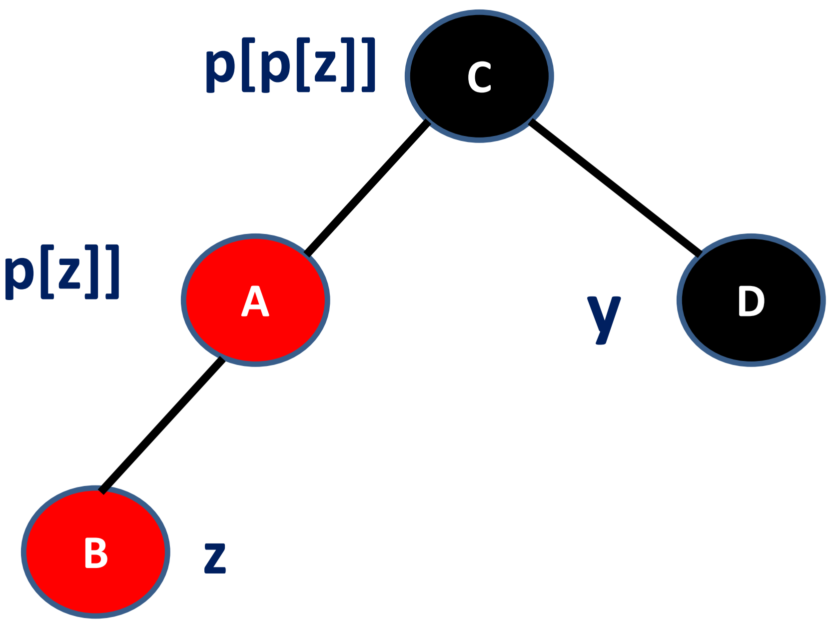
Insertion in Red-Back Tree

The color of new node(z) is red.
The the color of p[z] =RED

And

z is left child of p[z]
z's uncle y is BLACK

Case 3(A)



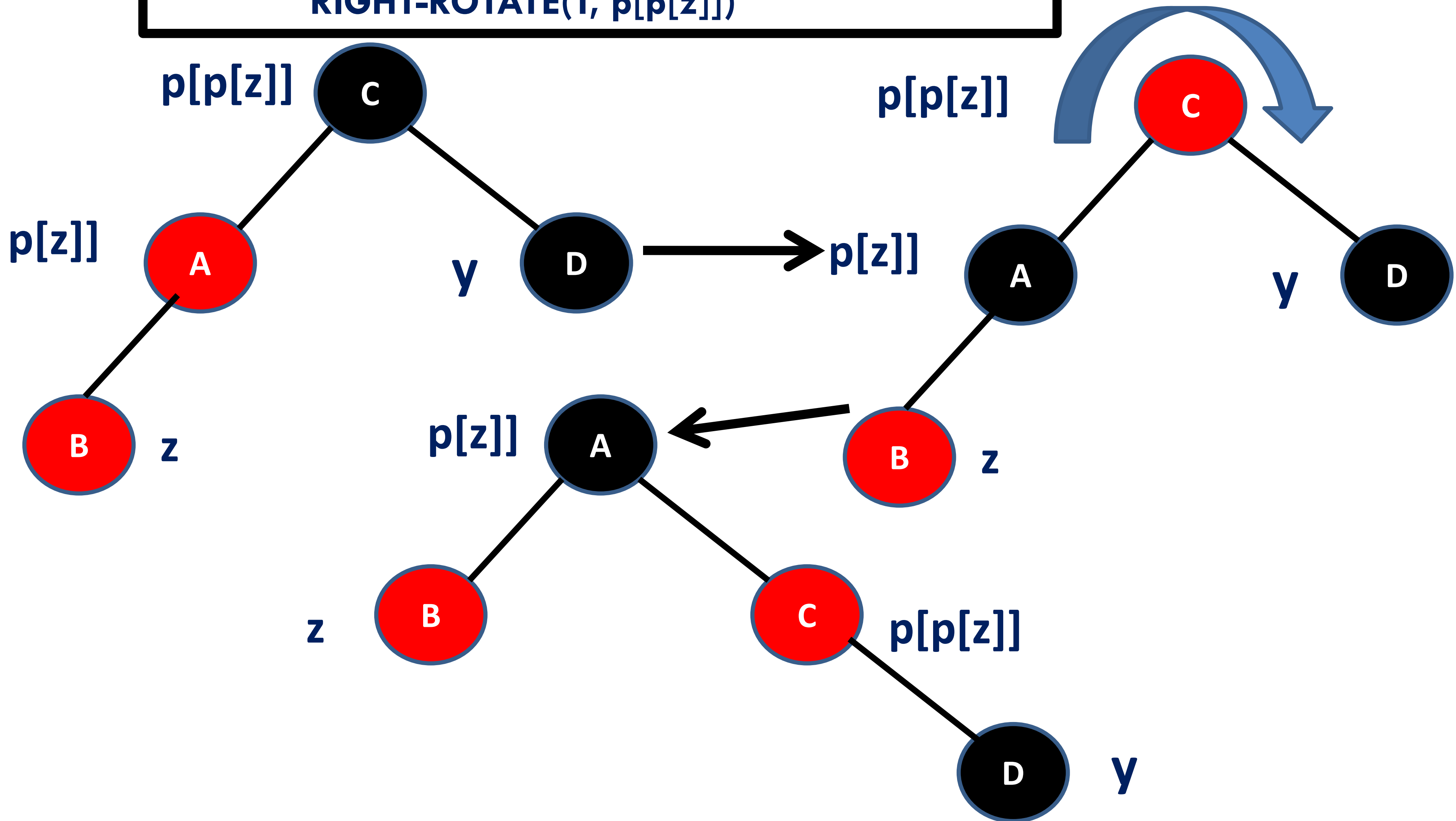
Solution: Recolor + RIGHT-ROTATE AROUND p[p[z]]

color[p[z]]$\leftarrow$BLACK

color[p[p[z]] $\leftarrow$RED

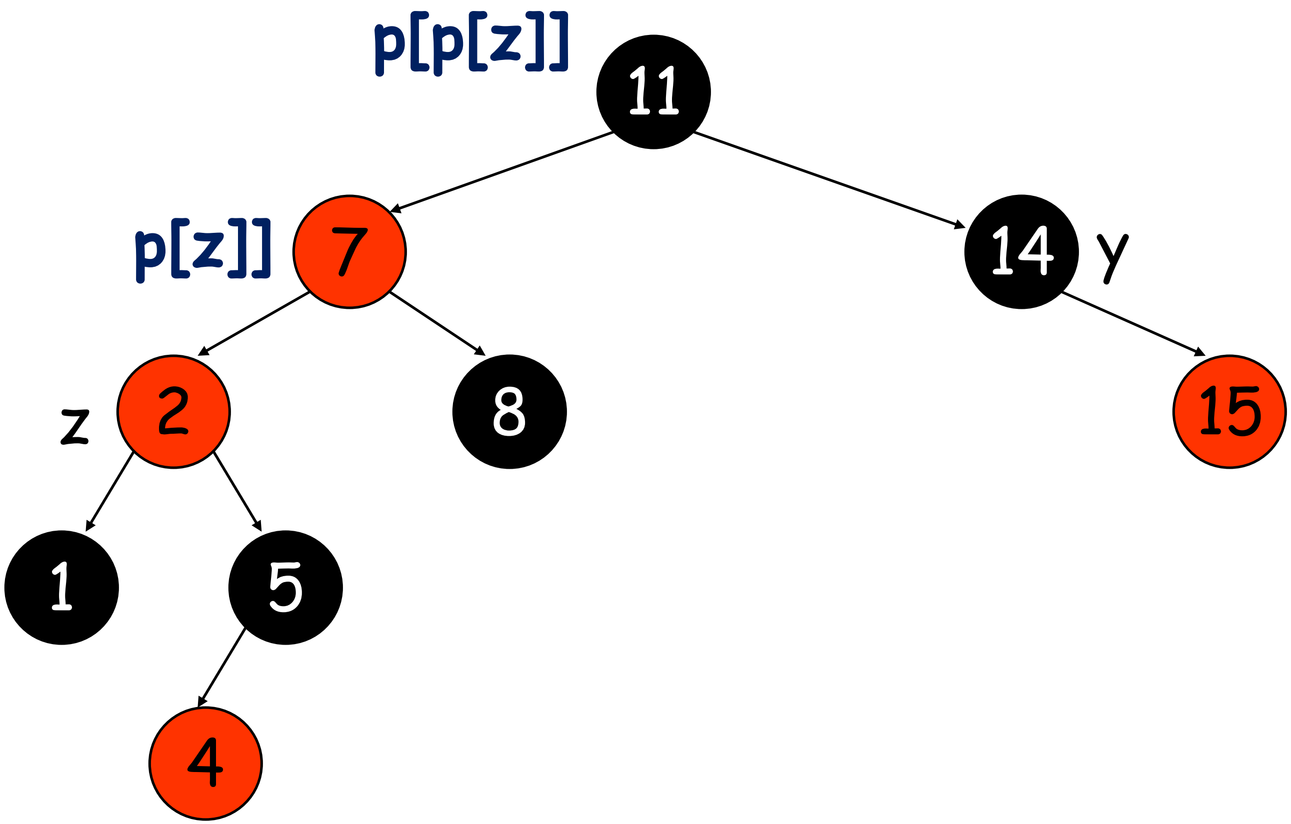
RIGHT-ROTATE(T, p[p[z]])

Case 3(A)



Insertion in Red-Back Tree

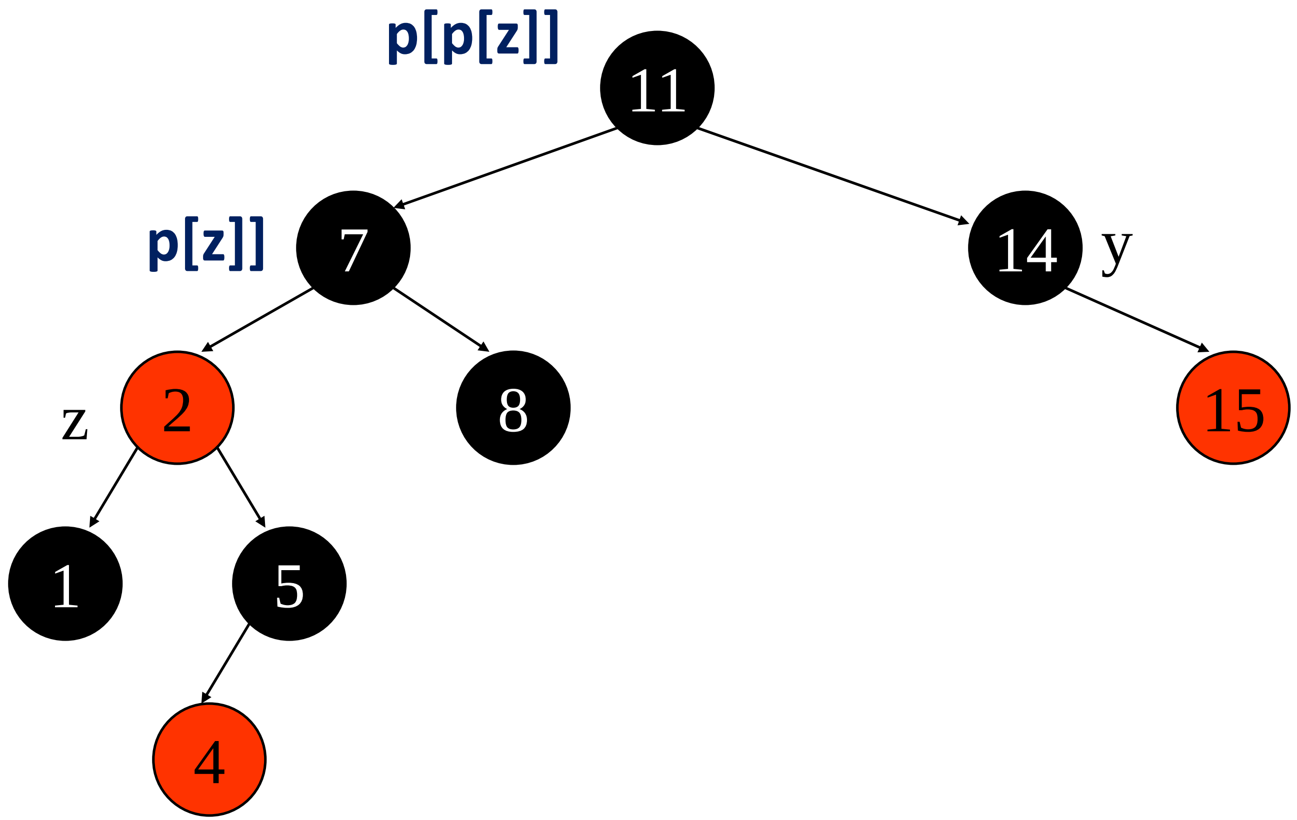
Case 3(A)



Insertion in Red-Back Tree

Case 3(A)

color[p[z]]←BLACK

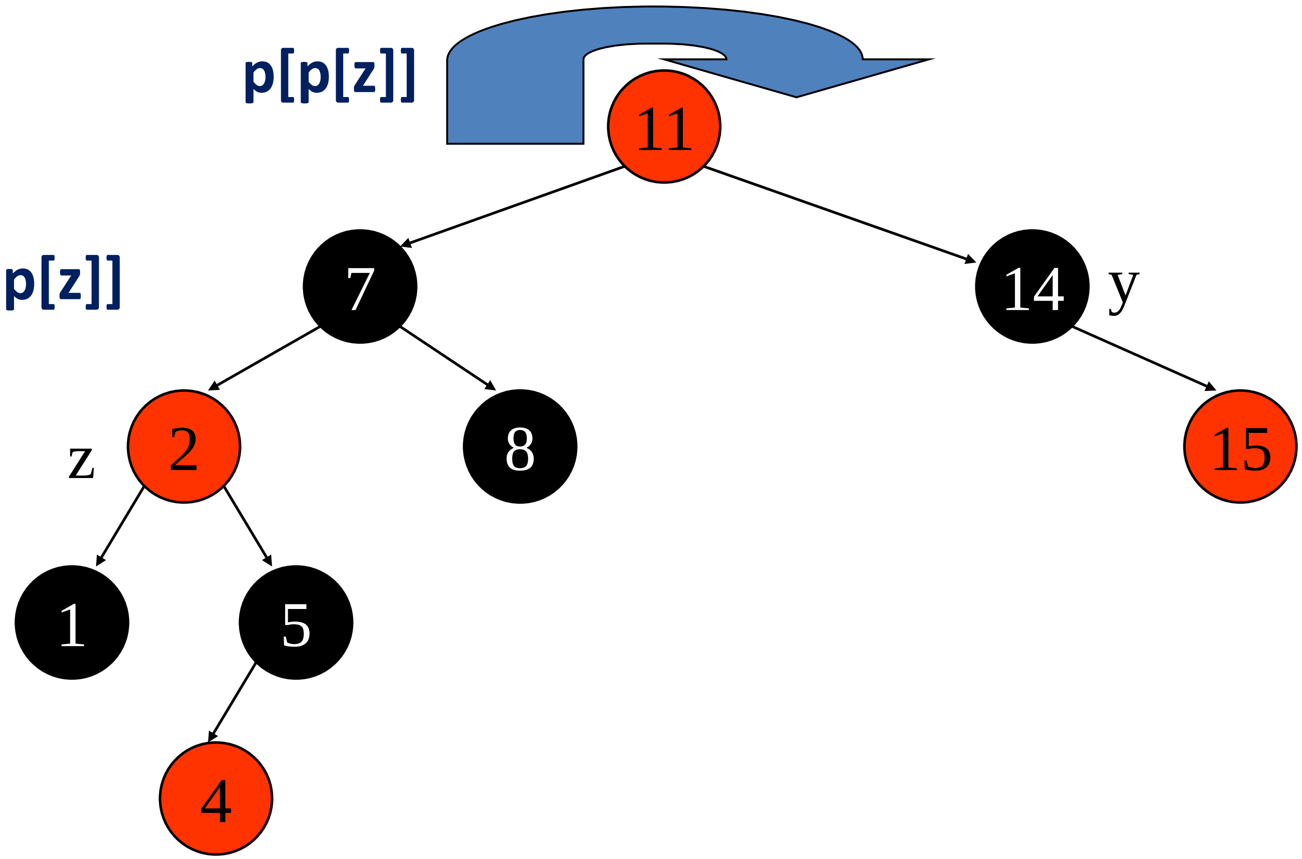


Insertion in Red-Back Tree

Case 3(A)

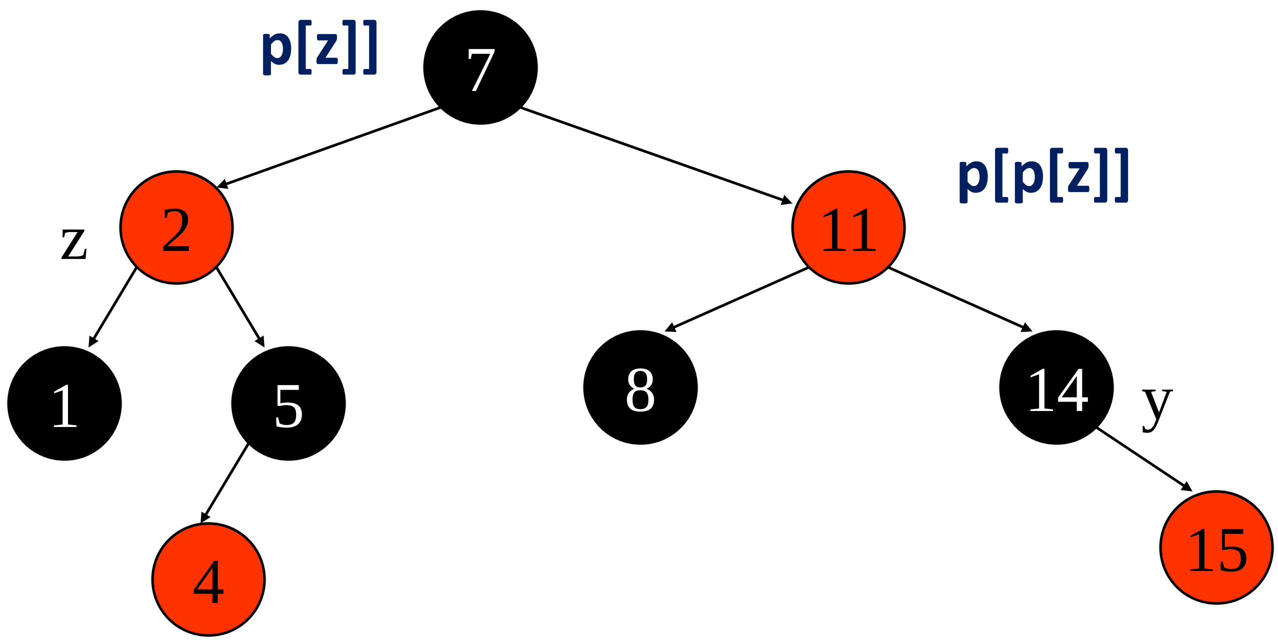
color[p[p[z]] ←RED

RIGHT-ROTATE(T, p[p[z]])



Insertion in Red-Back Tree

Case 3(A)



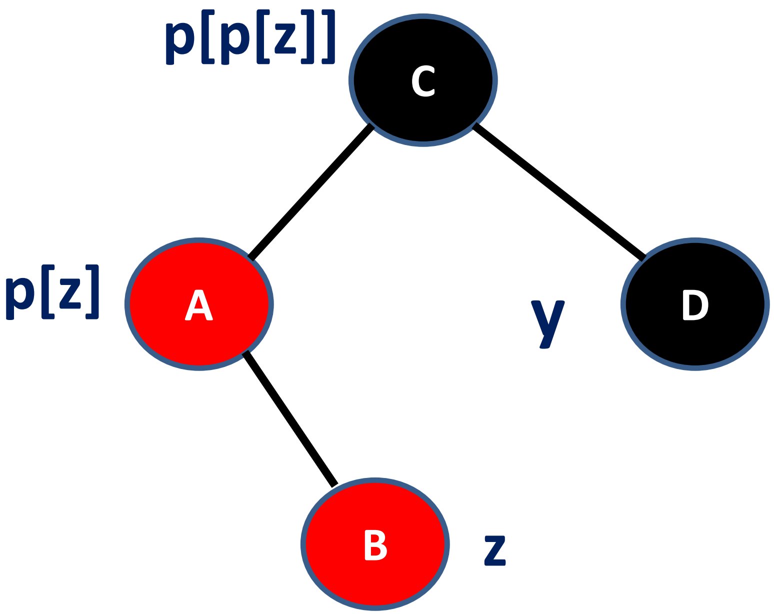
Insertion in Red-Back Tree

The color of new node(z) is red.
The the color of p[z] =RED

And

z is right child of p[z]
z's uncle y is BLACK

Case 3(B)



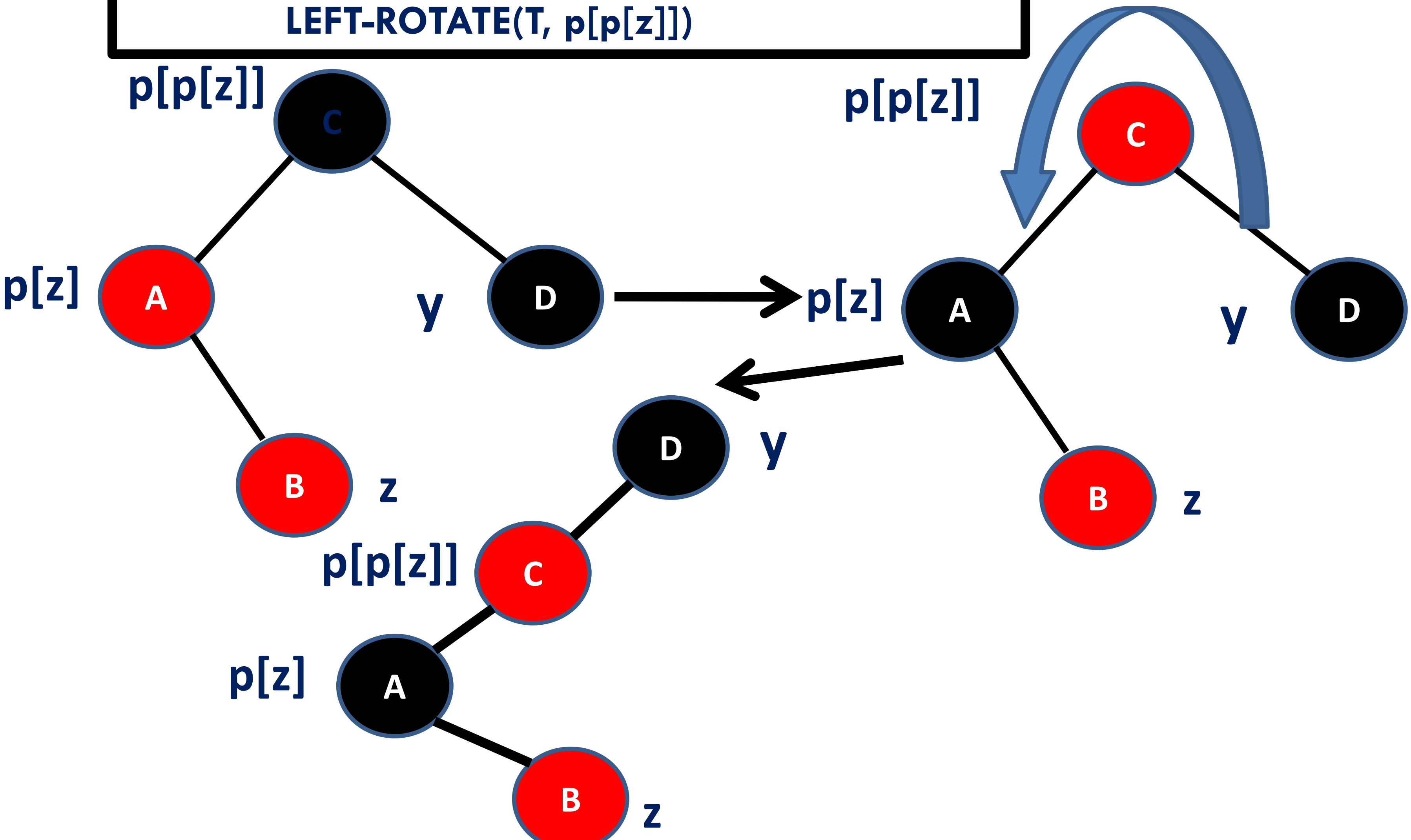
Solution: Recolor + LEFT-ROTATE AROUND p[p[z]]

color[p[z]]←BLACK

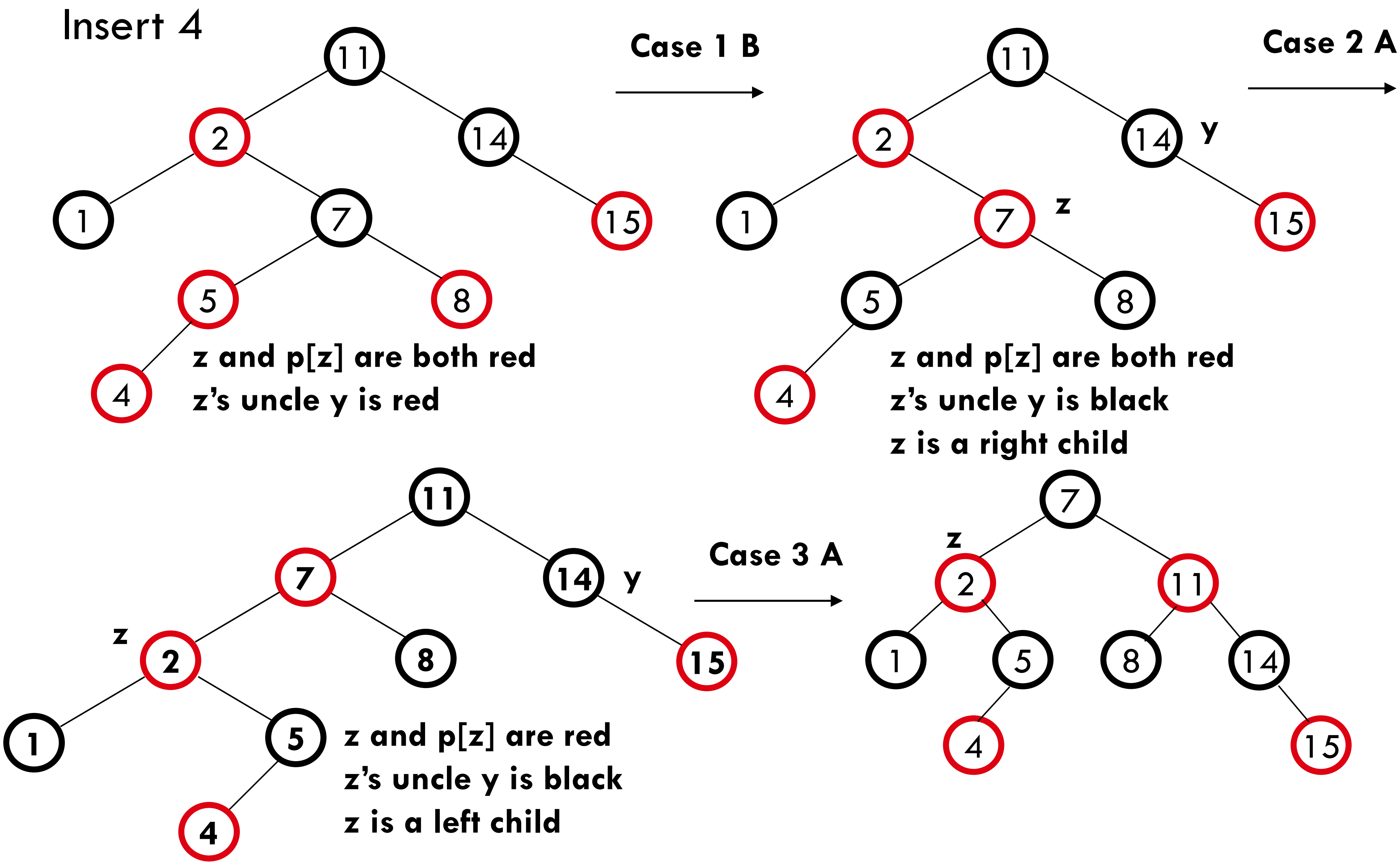
color[p[p[z]] ←RED

LEFT-ROTATE(T, p[p[z]])

Case 3(B)



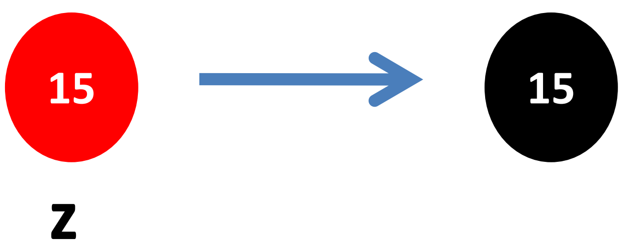
Insertion in Red-Back Tree



Example

Insert the following nodes in sequence in an empty Red-Black Tree:

15, 12, 35, 3, 21

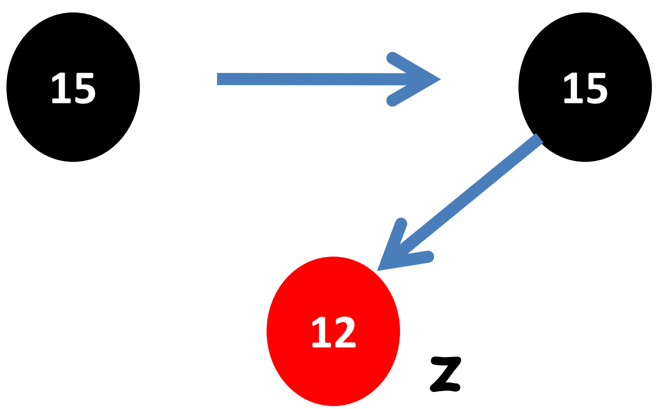


It is Root. Color it BLACK

Example

Insert the following nodes in sequence in an empty Red-Black Tree:

15, 12, 35, 3, 21

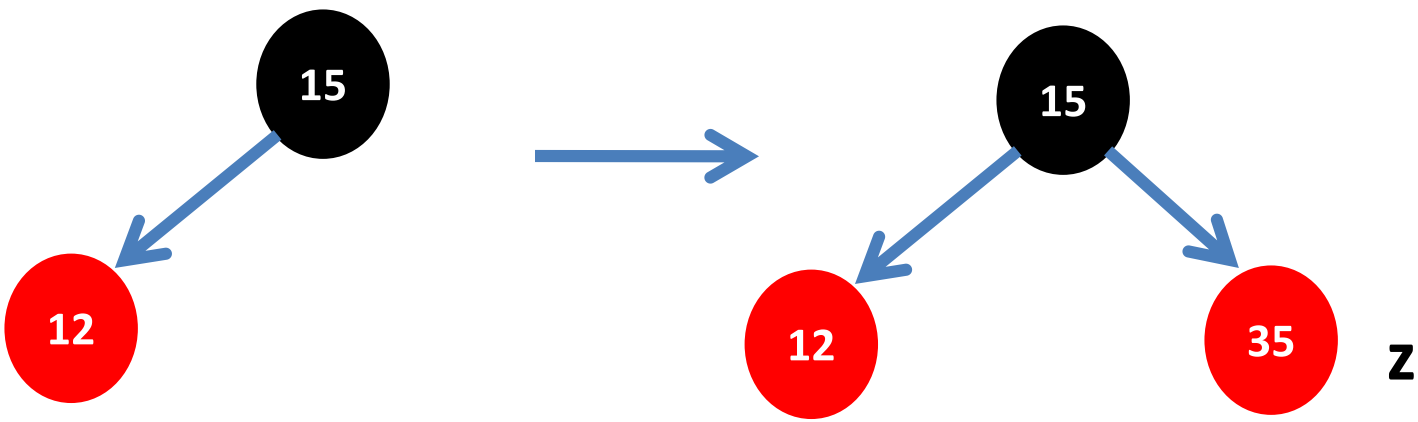


No RED-BLACK Tree properties violation

Example

Insert the following nodes in sequence in an empty Red-Black Tree:

15, 12, 35, 3, 21

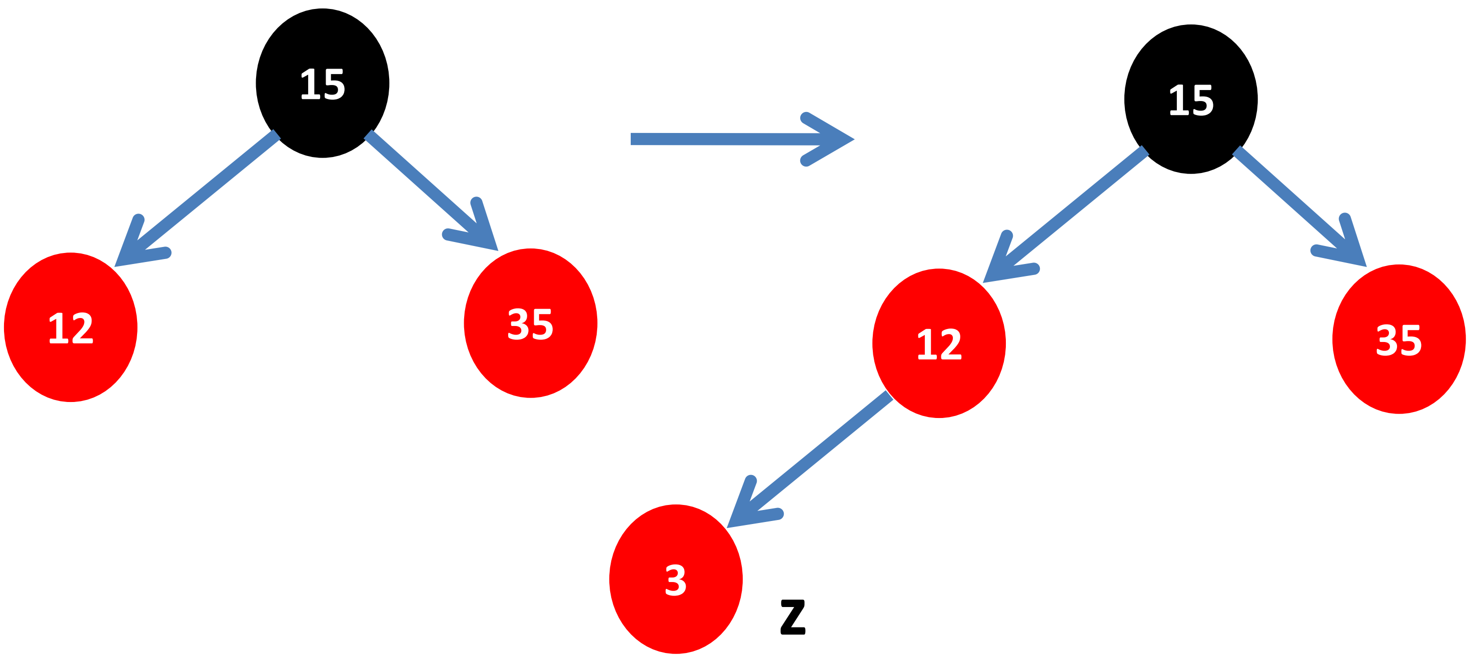


No RED-BLACK Tree properties violation

Example

Insert the following nodes in sequence in an empty Red-Black Tree:

15, 12, 35, 3, 21



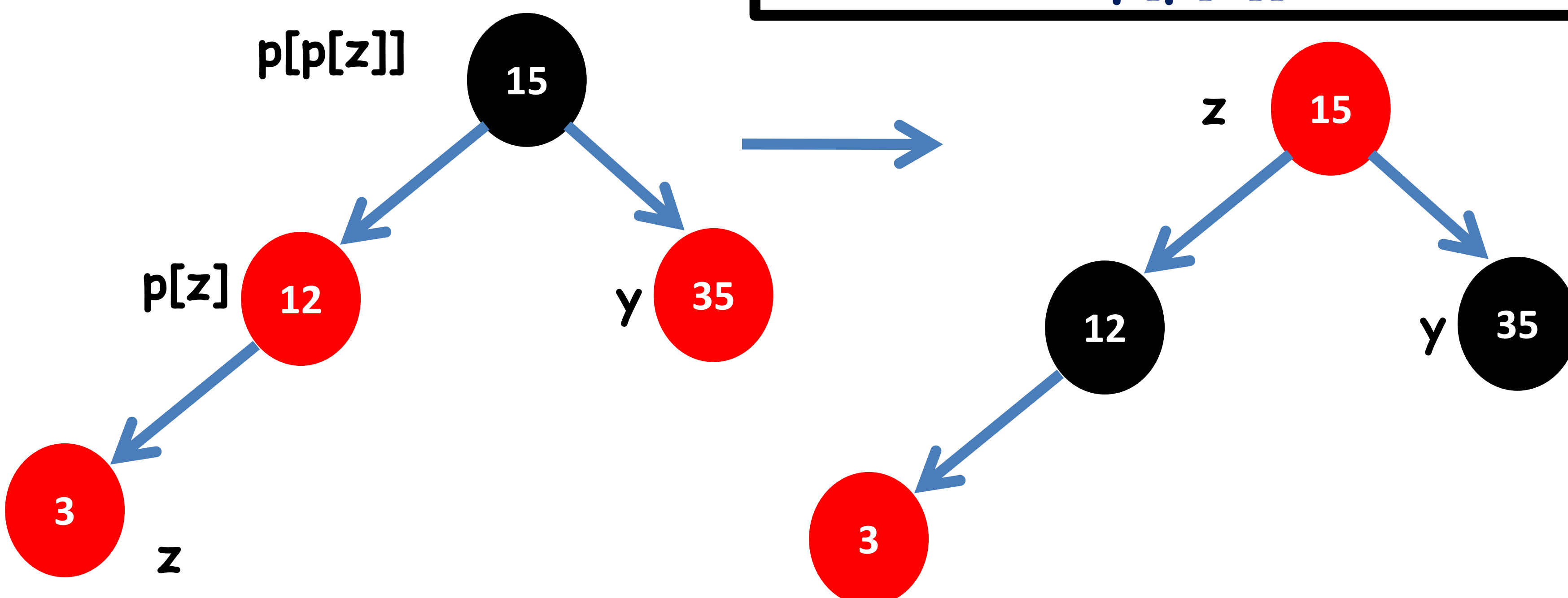
Property (4) Violated: There should not be two consecutive reds

Example

Case 1(B)

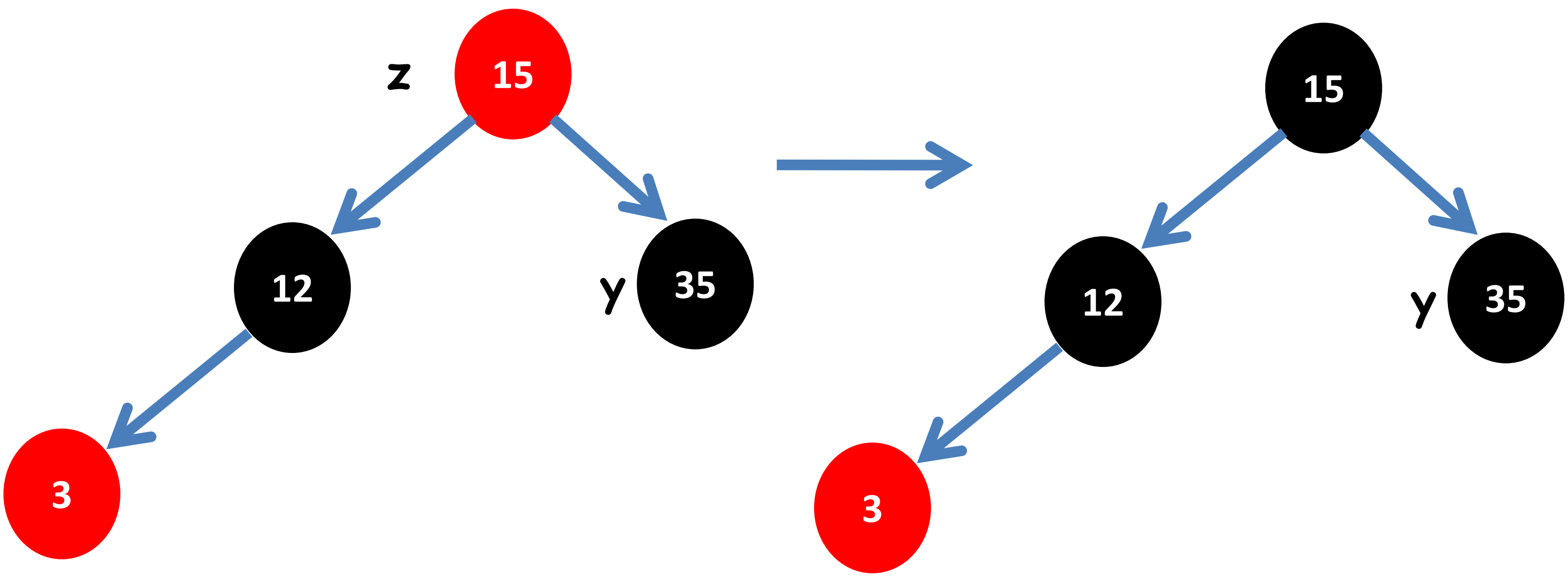
z is Left child of p[z]
z's uncle y is RED

Solution: Recolor
color[p[z]] ← BLACK
color[y] ← BLACK
color[p[p[z]] ← RED
z=p[p[z]]



Since z is Root , Make its color BLACK

Example

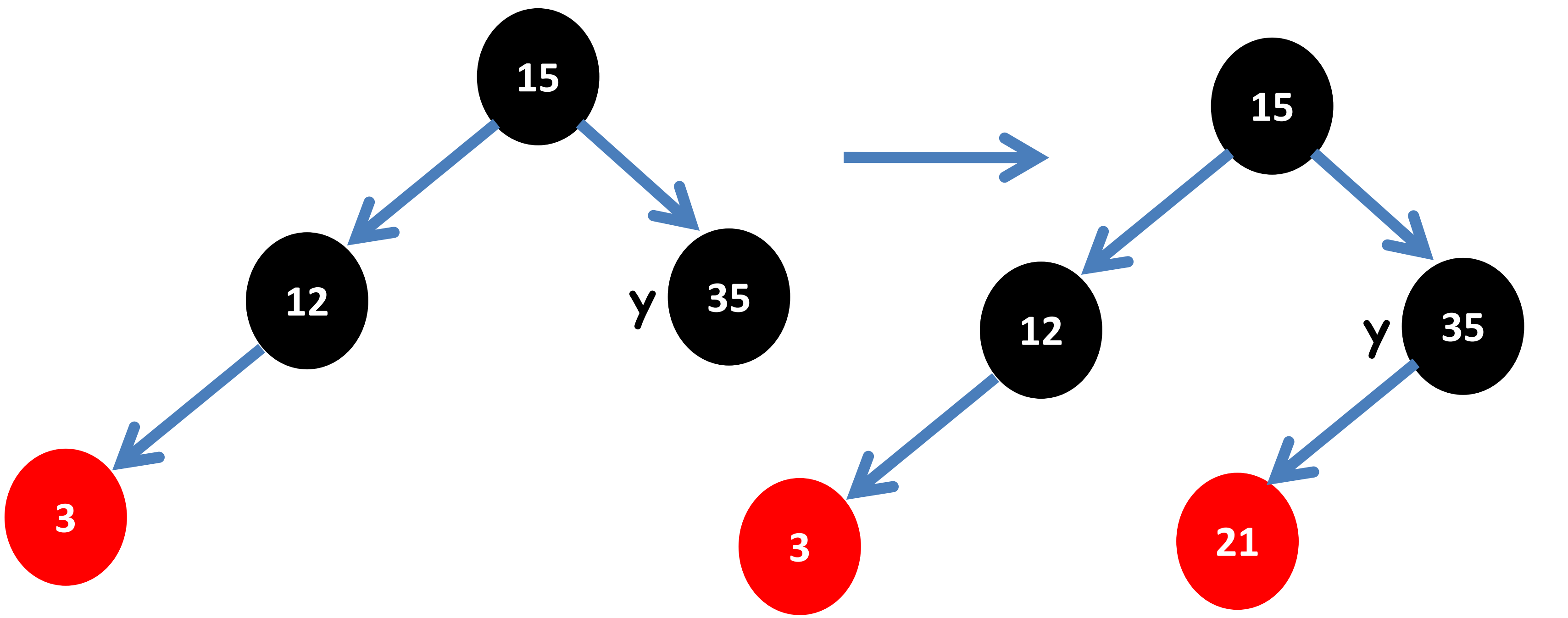


Since z is Root , Make its color BLACK

Example

Insert the following nodes in sequence in an empty Red-Black Tree:

15, 12, 35, 3, 21

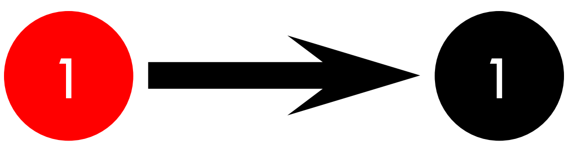


No RED-BLACK Tree properties violation

Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, 4, 6

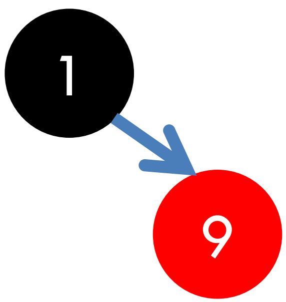


It is Root. Color it BLACK

Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, 4, 6

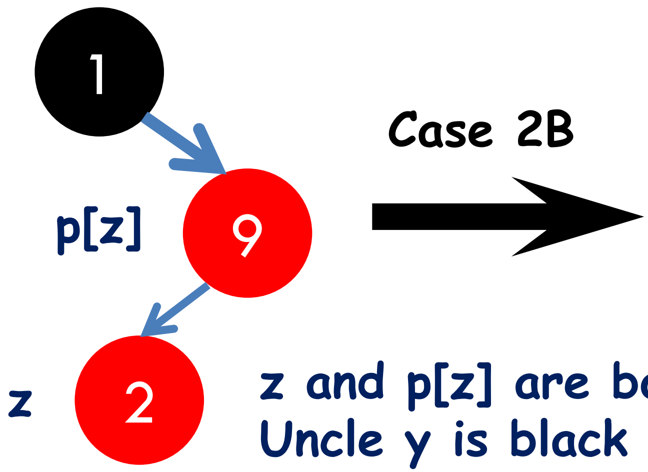


No RED-BLACK Tree properties violation

Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, 4, 6



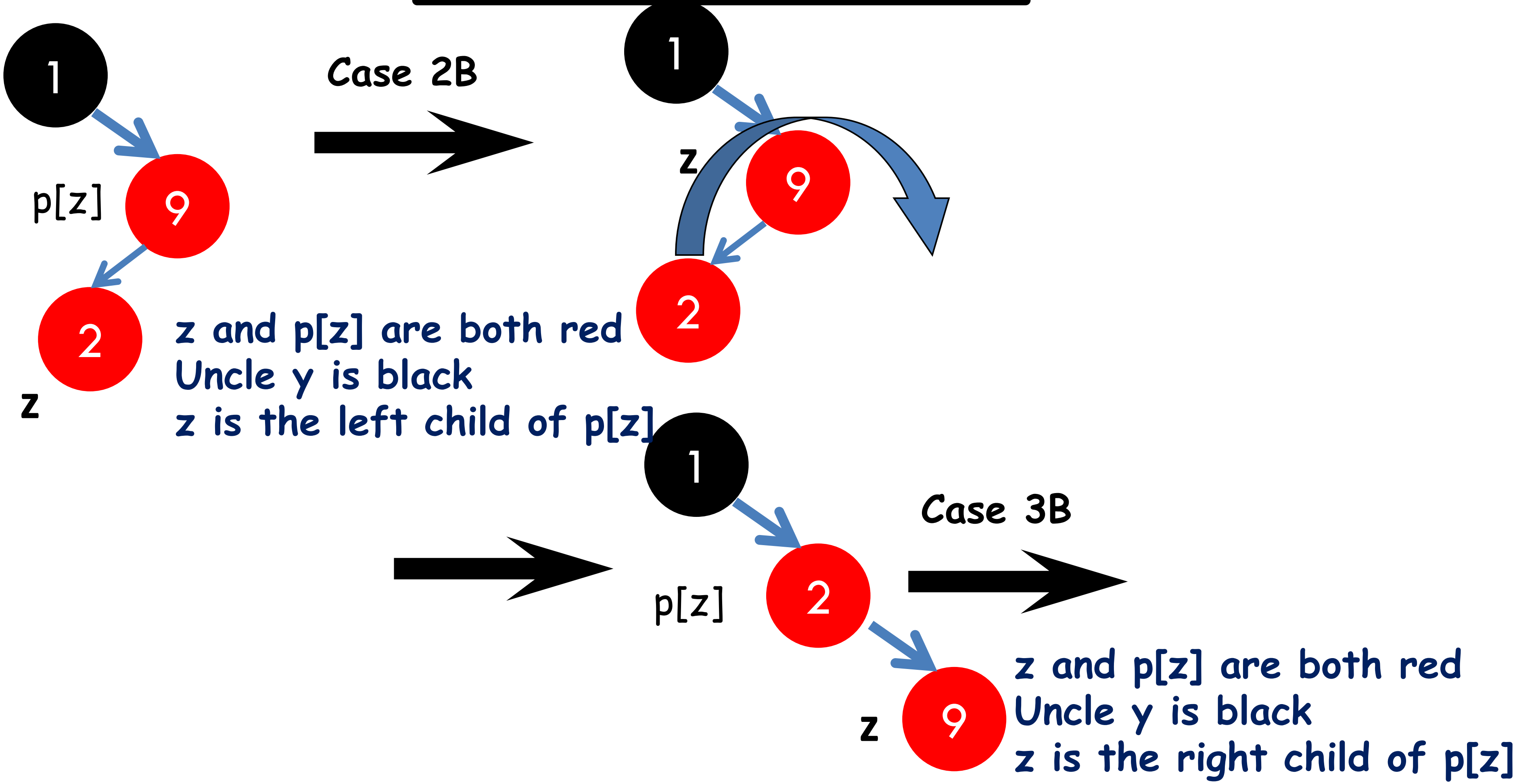
z and p[z] are both red
Uncle y is black (let)
z is the left child of p[z]

Case 2B

Solution: Right Rotation

$z \leftarrow p[z]$

RIGHT –ROTATE(T, z)



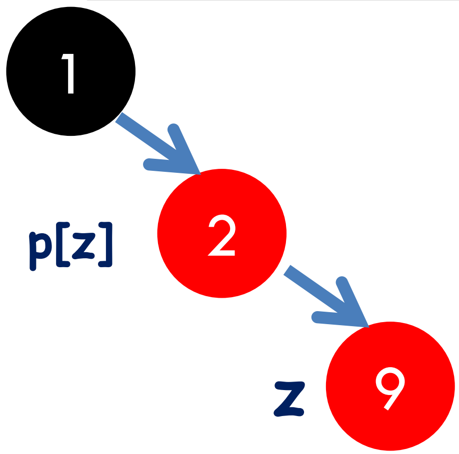
Case 3B

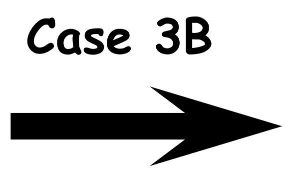
Solution: Recolor + LEFT-ROTATE AROUND p[p[z]]

color[p[z]]⟵BLACK

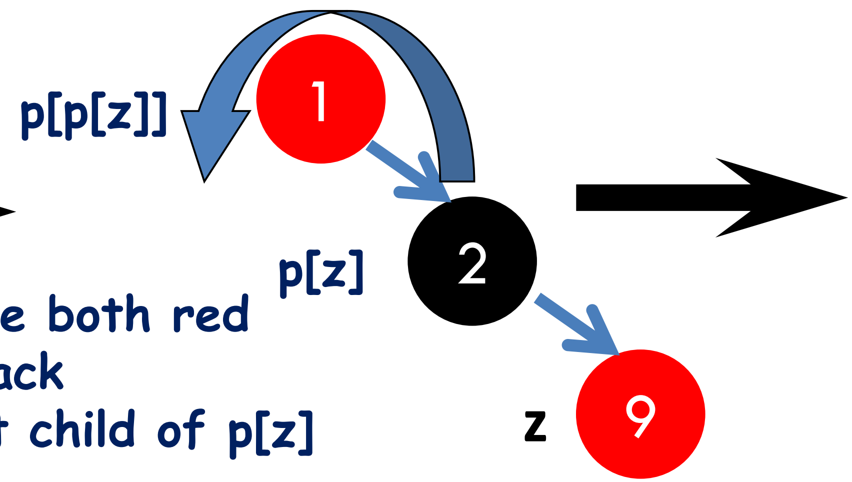
color[p[p[z]] ⟵RED

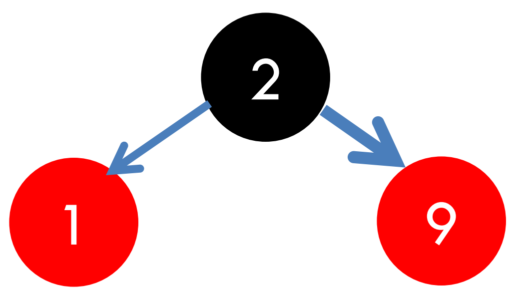
LEFT-ROTATE(T, p[p[z]])





z and p[z] are both red
Uncle y is black
z is the right child of p[z]

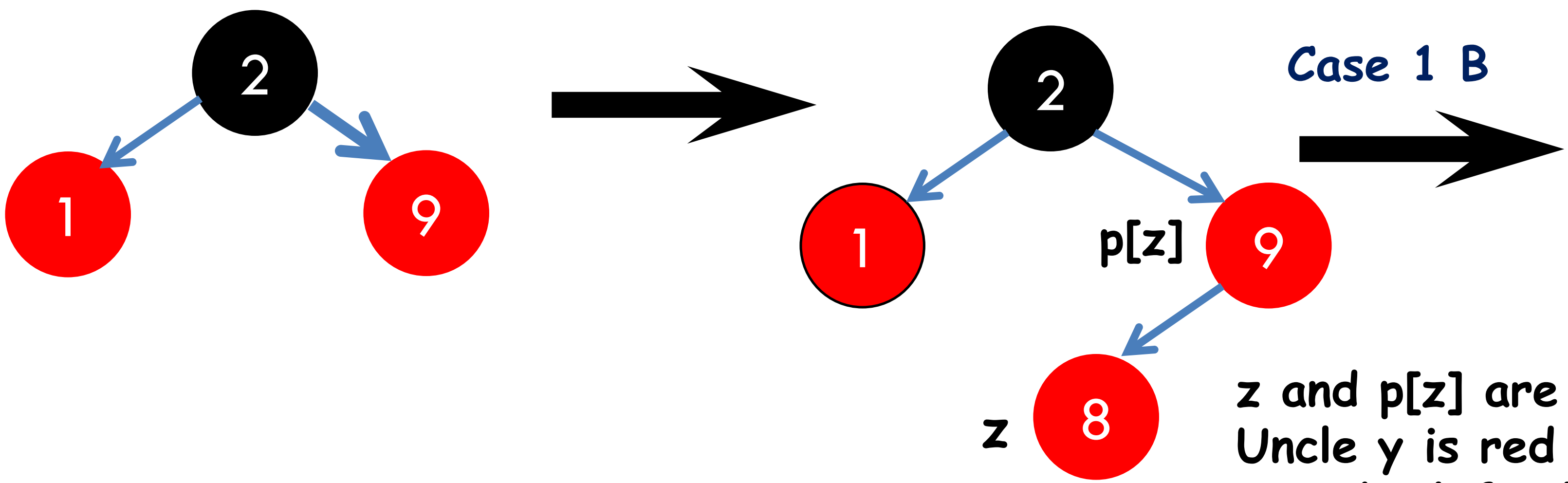




Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, 4, 6

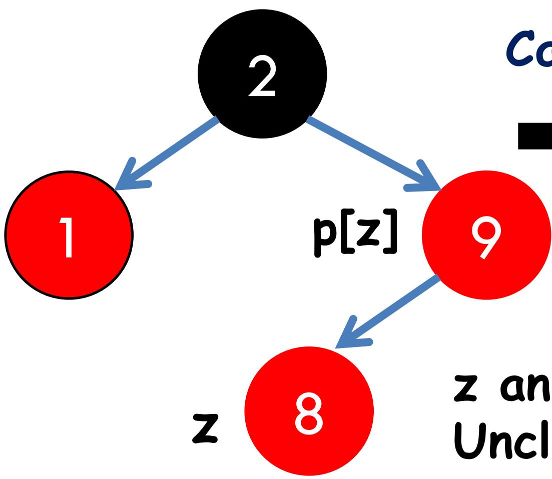


z and p[z] are both red
Uncle y is red
z is the left child of p[z]

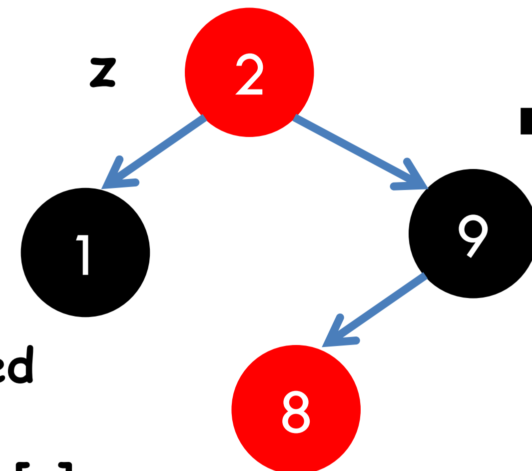
Case 1 B

Solution: Recolor

color[p[z]]←BLACK
color[y] ←BLACK
color[p[p[z]] ←RED
z=p[p[z]]

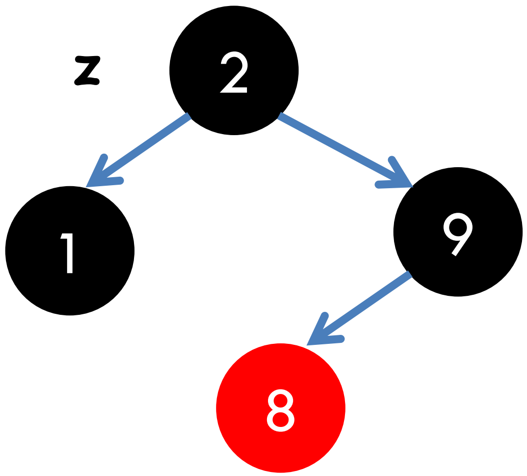


Case 1 B



z and p[z] are both red
Uncle y is red
z is the left child of p[z]

z is Root. Color it BLACK

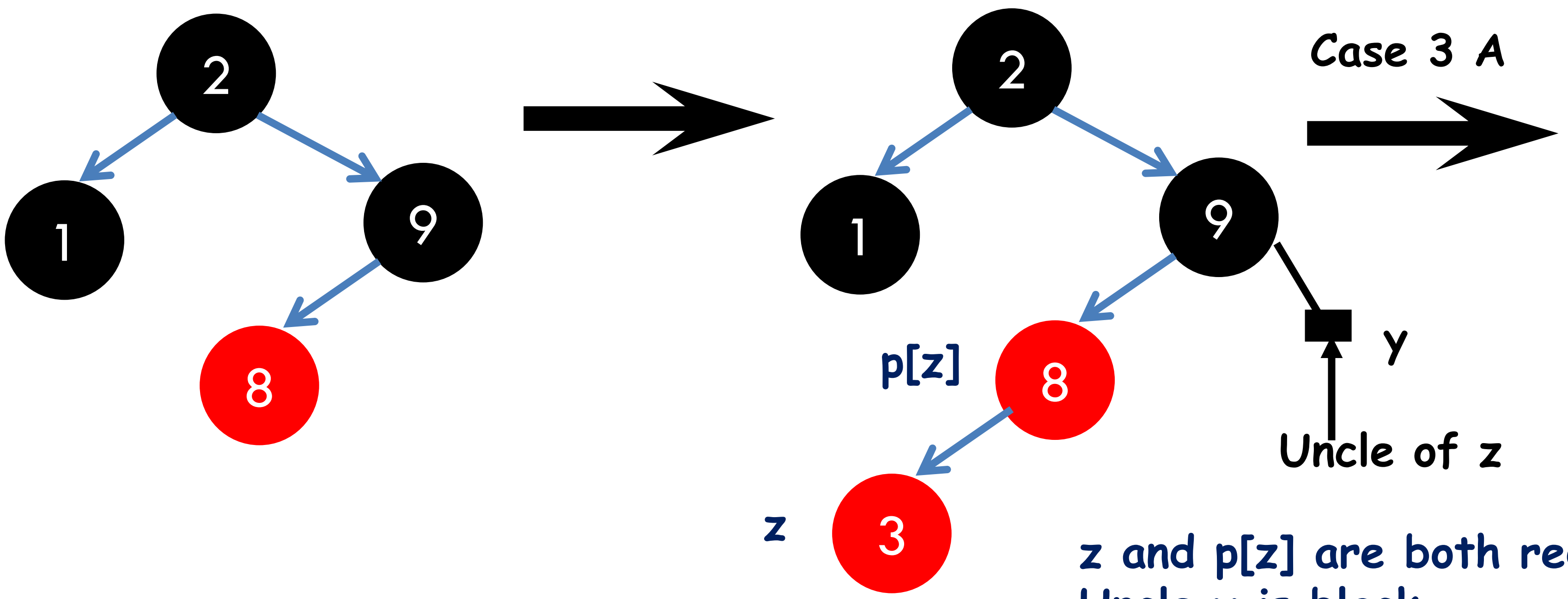

z
2
1
9
8

Learn DAA: From B K Sharma

Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, 4, 6



z and p[z] are both red
Uncle y is black
z is the left child of p[z]

Learn DAA: From B K Sharma

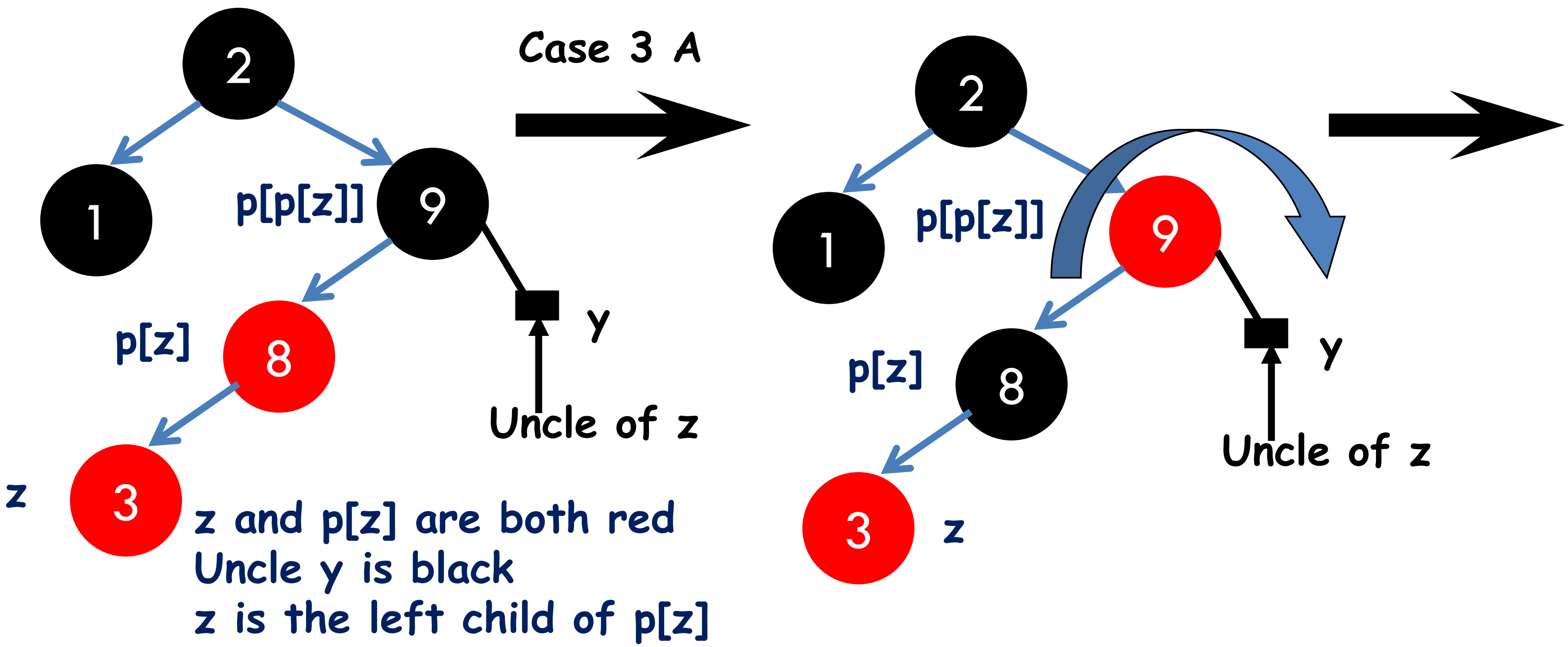
Case 3 A

Solution: Recolor + RIGHT-ROTATE AROUND p[p[z]]

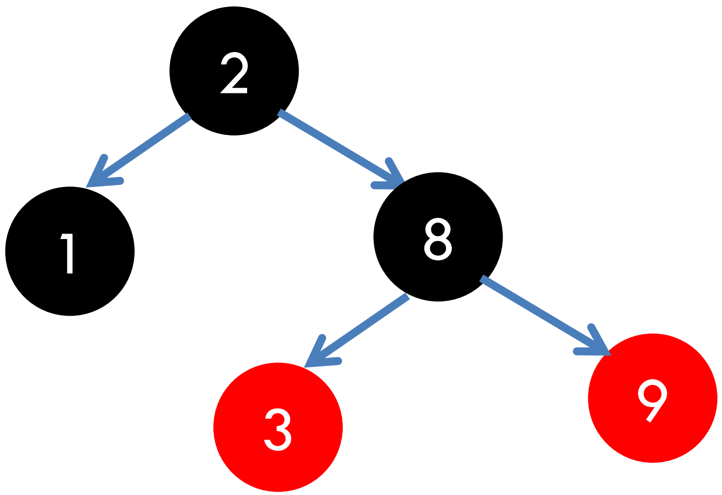
color[p[z]]←BLACK

color[p[p[z]] ←RED

RIGHT-ROTATE(T, p[p[z]])



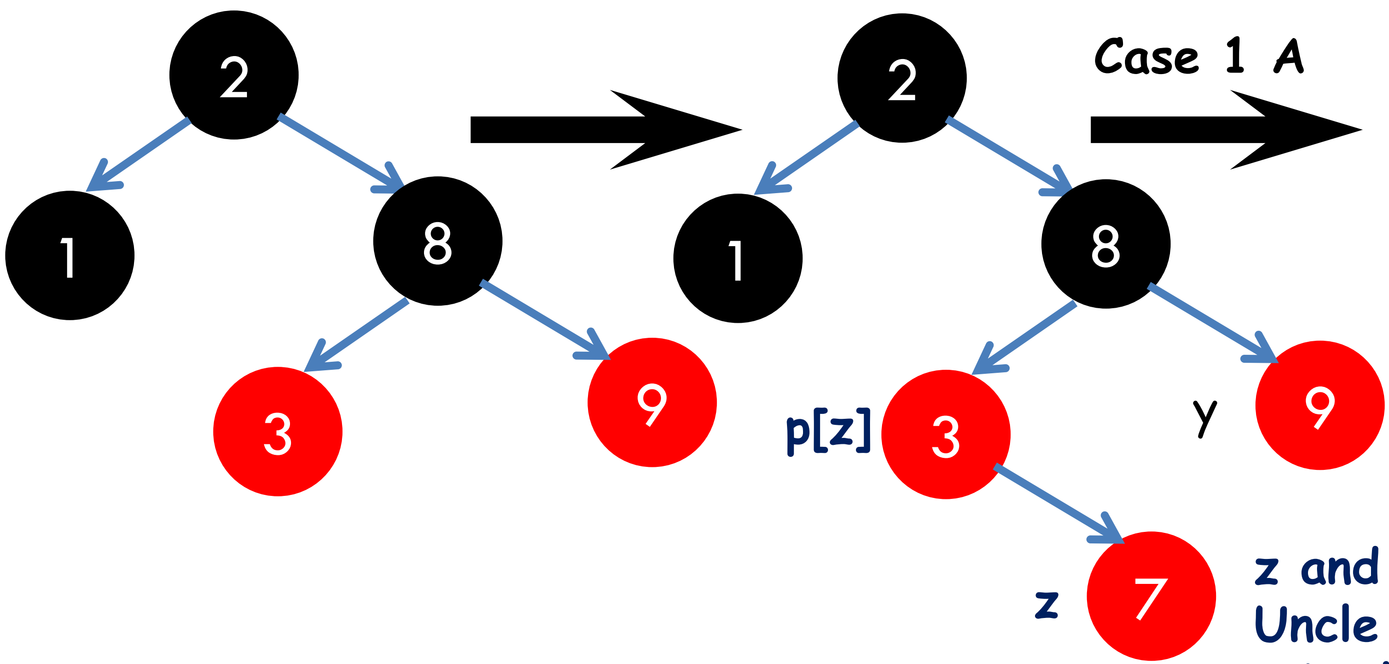
Example



Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

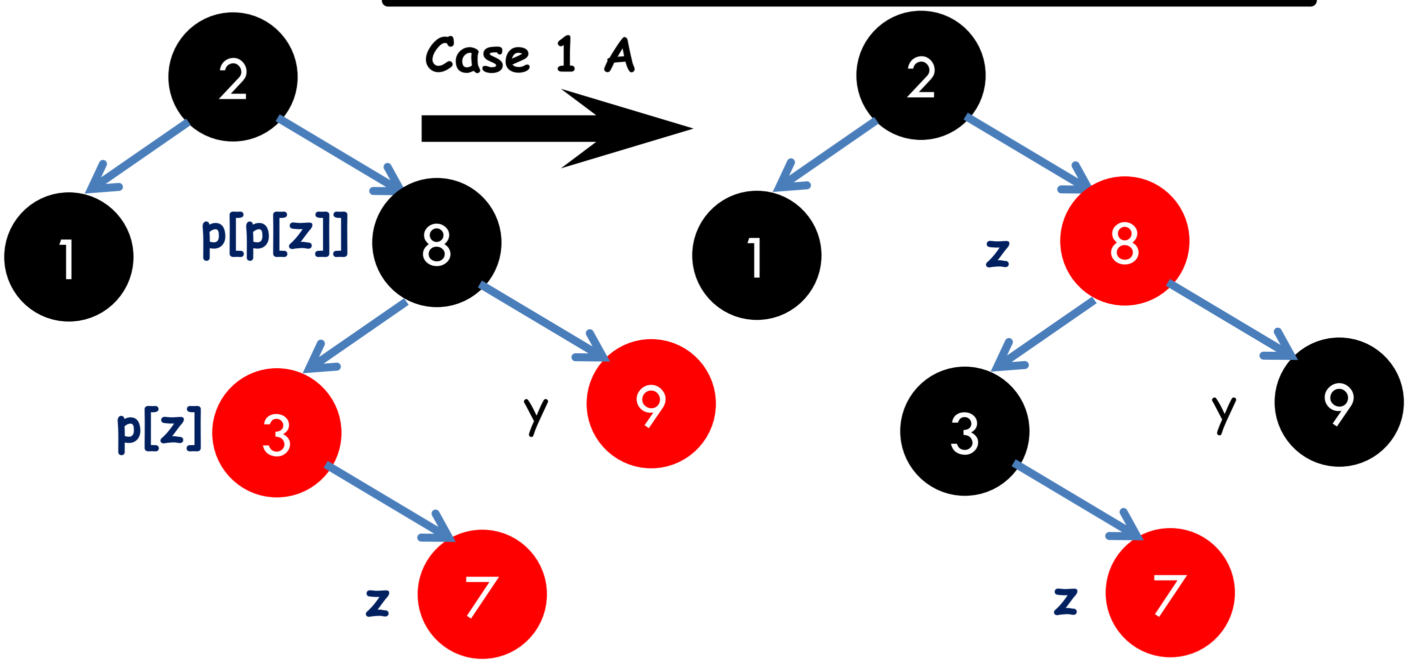
1, 9, 2, 8, 3, 7, 4, 6



z and p[z] are both red
Uncle y is red
z is the right child of p[z]

Case 1(A)

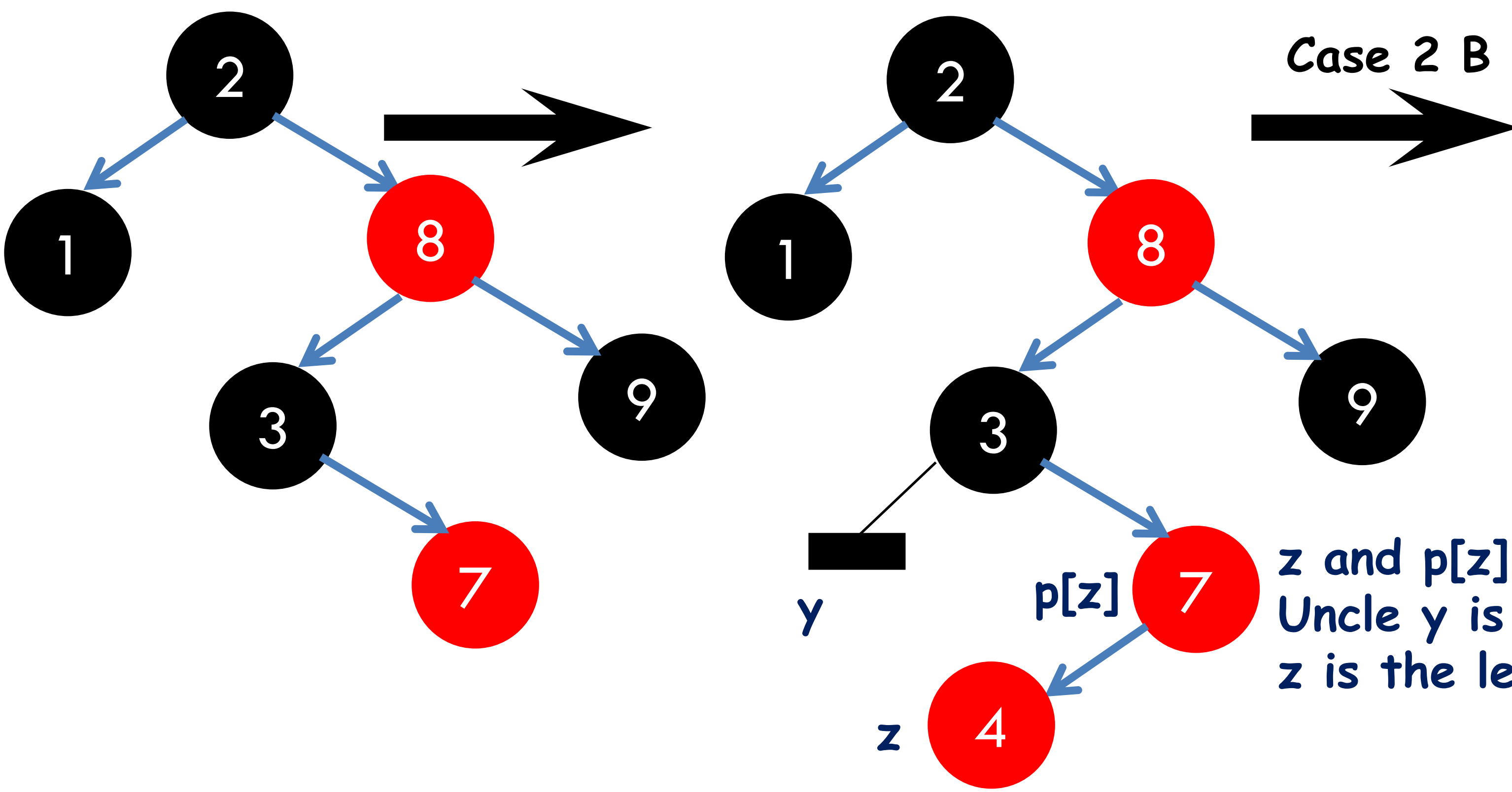
Solution: Recolor
color[p[z]]←BLACK
color[y] ←BLACK
color[p[p[z]] ←RED
z=p[p[z]]



Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, **4**, 6



z and p[z] are both red
Uncle y is black
z is the left child of p[z]

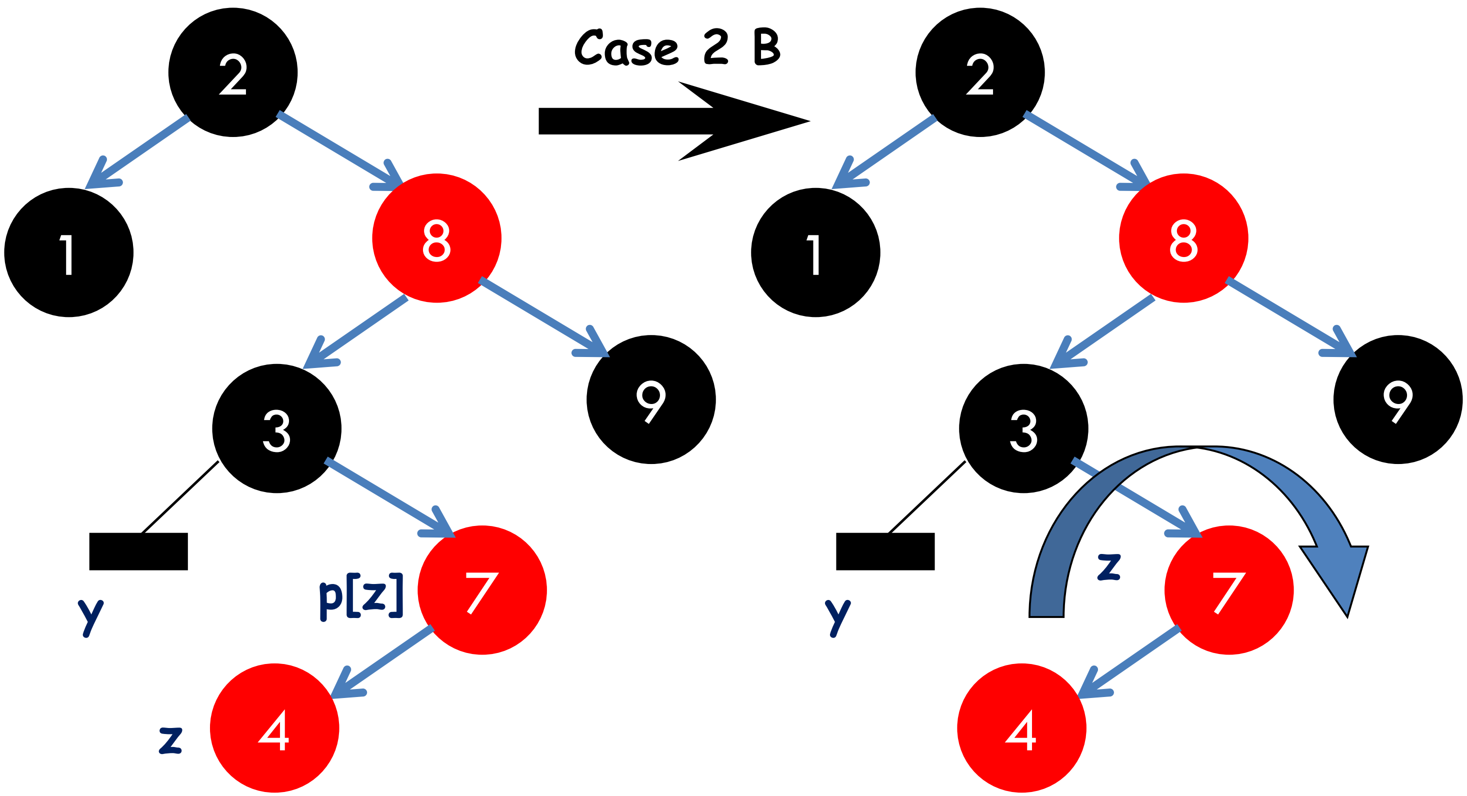
Learn DAA: From B K Sharma

Case 2B

Solution: Right Rotation

z ← p[z]

RIGHT –ROTATE(T, z)



Learn DAA: From B K Sharma

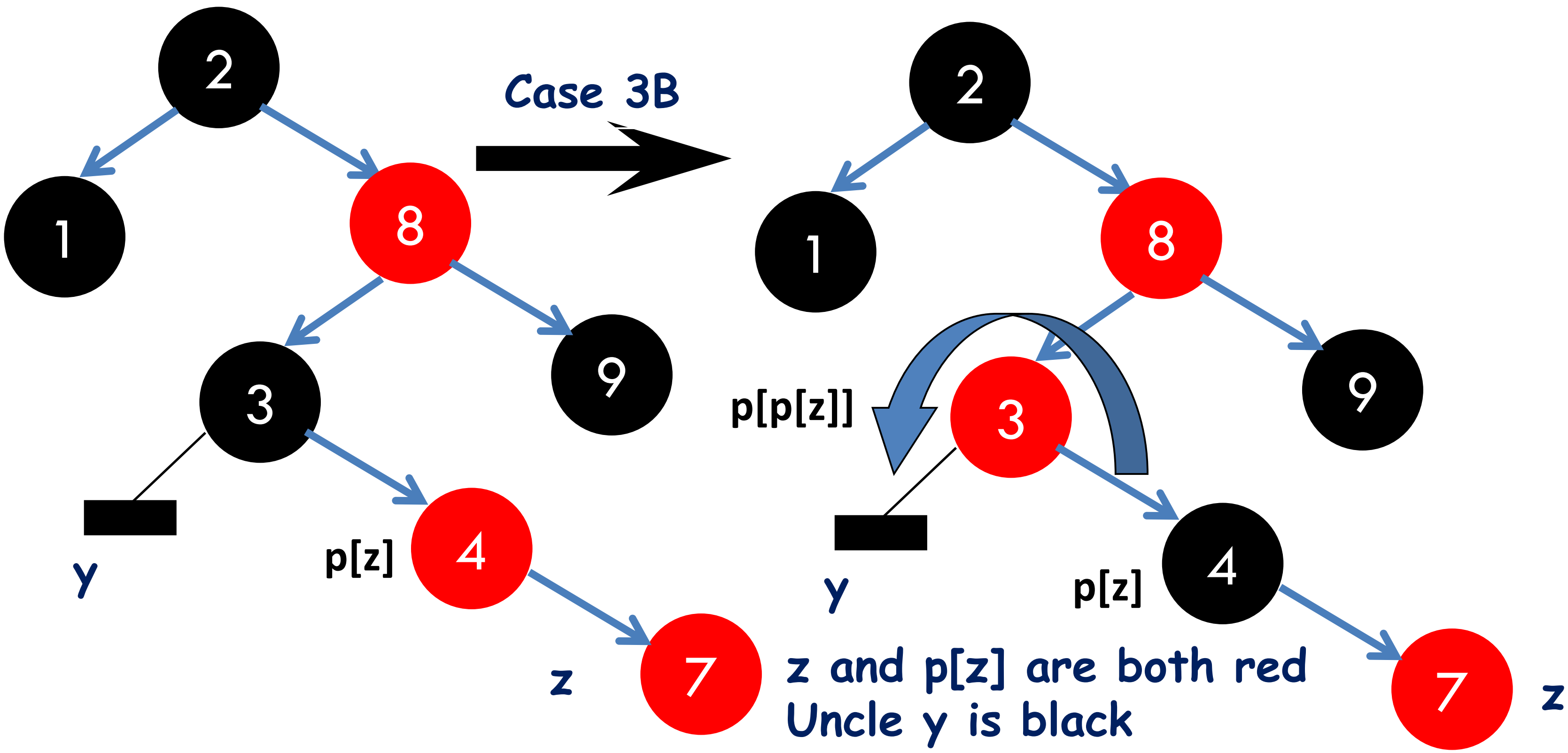
Case 3B

Solution: Recolor + LEFT-ROTATE AROUND p[p[z]]

color[p[z]]←**BLACK**

color[p[p[z]] ←**RED**

LEFT-ROTATE(T, p[p[z]])

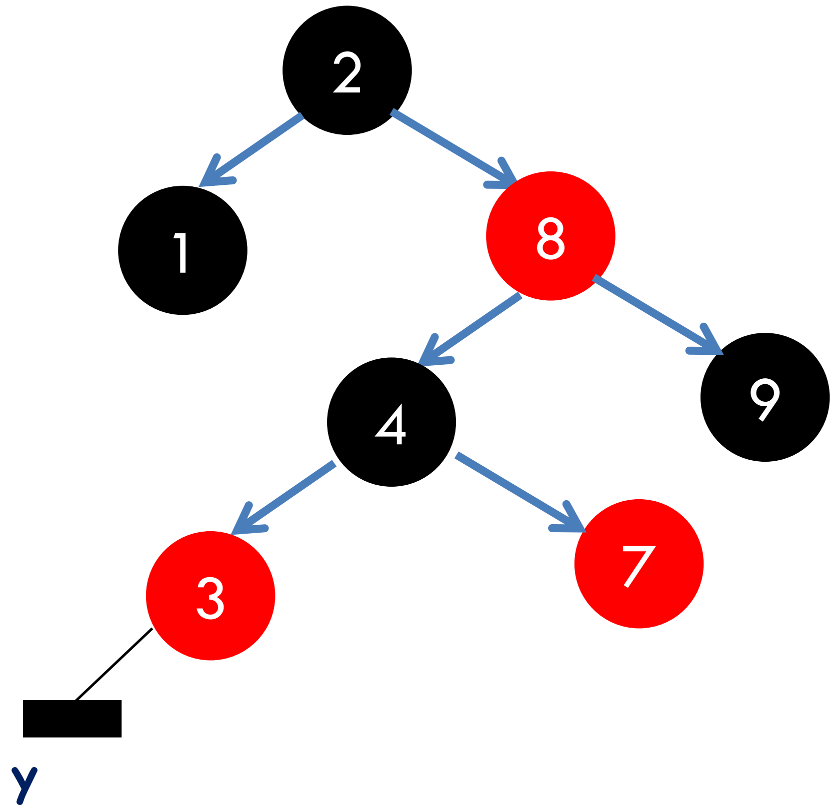


z and p[z] are both red

Uncle y is black

z is the right child of p[z]

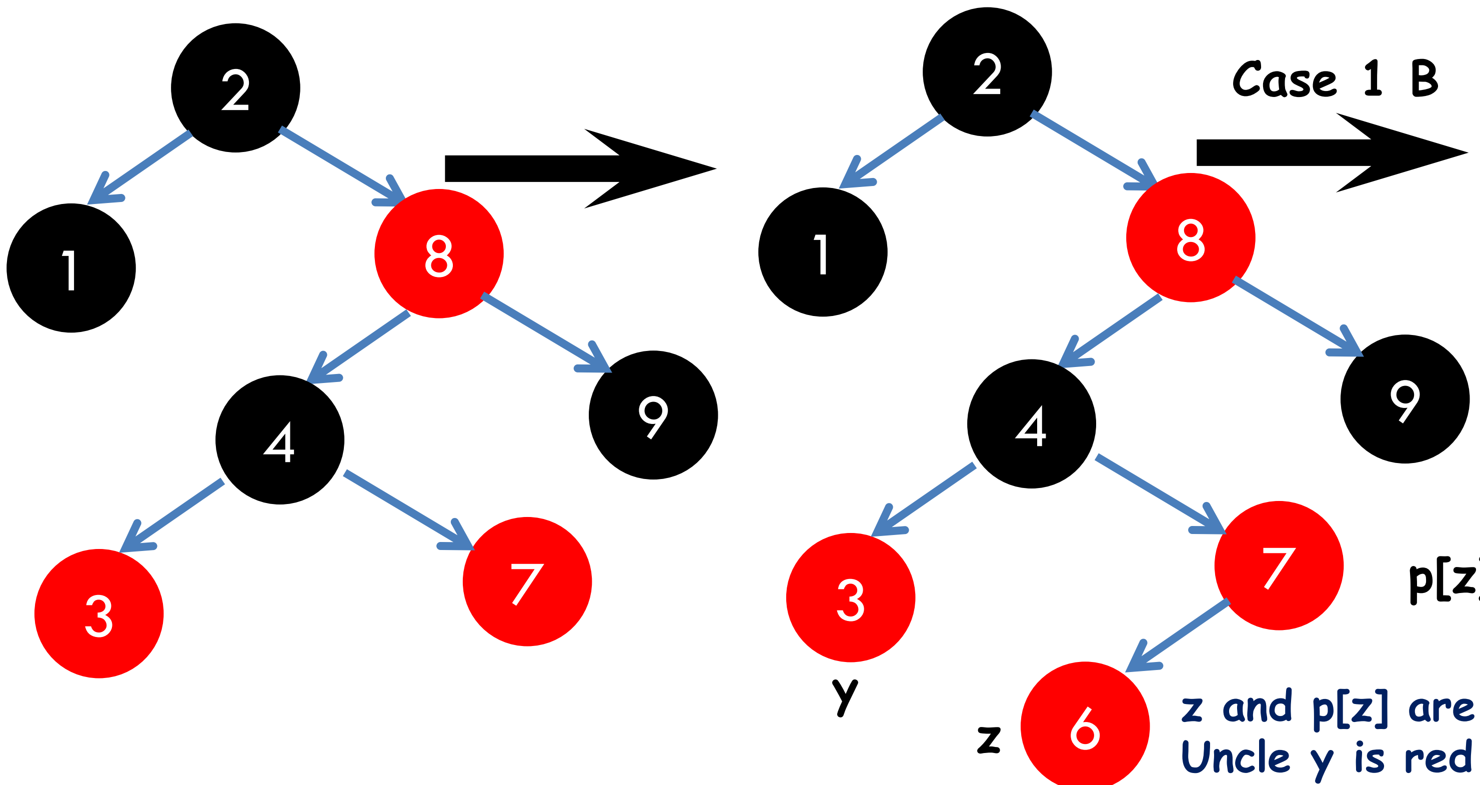
Learn DAA: From B K Sharma



Example

Inserting the following nodes in an empty Red-Black Tree in sequence:

1, 9, 2, 8, 3, 7, 4, 6



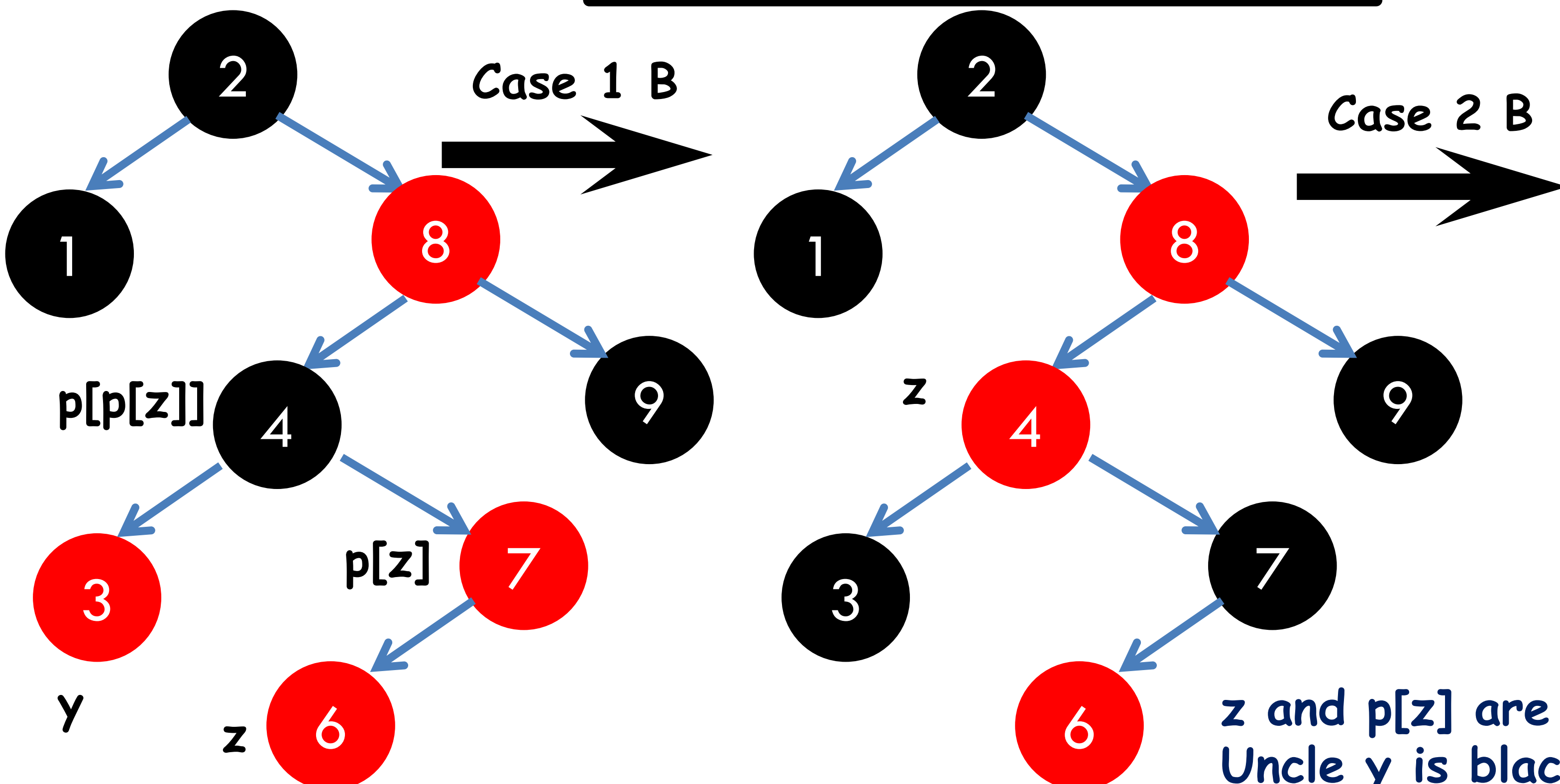
z and p[z] are both red
Uncle y is red
z is the left child of p[z]

Learn DAA: From B K Sharma

Case 1 B

Solution: Recolor

color[p[z]]←BLACK
color[y] ←BLACK
color[p[p[z]] ←RED
z=p[p[z]]



z and p[z] are both red
Uncle y is black
z is the left child of p[z]

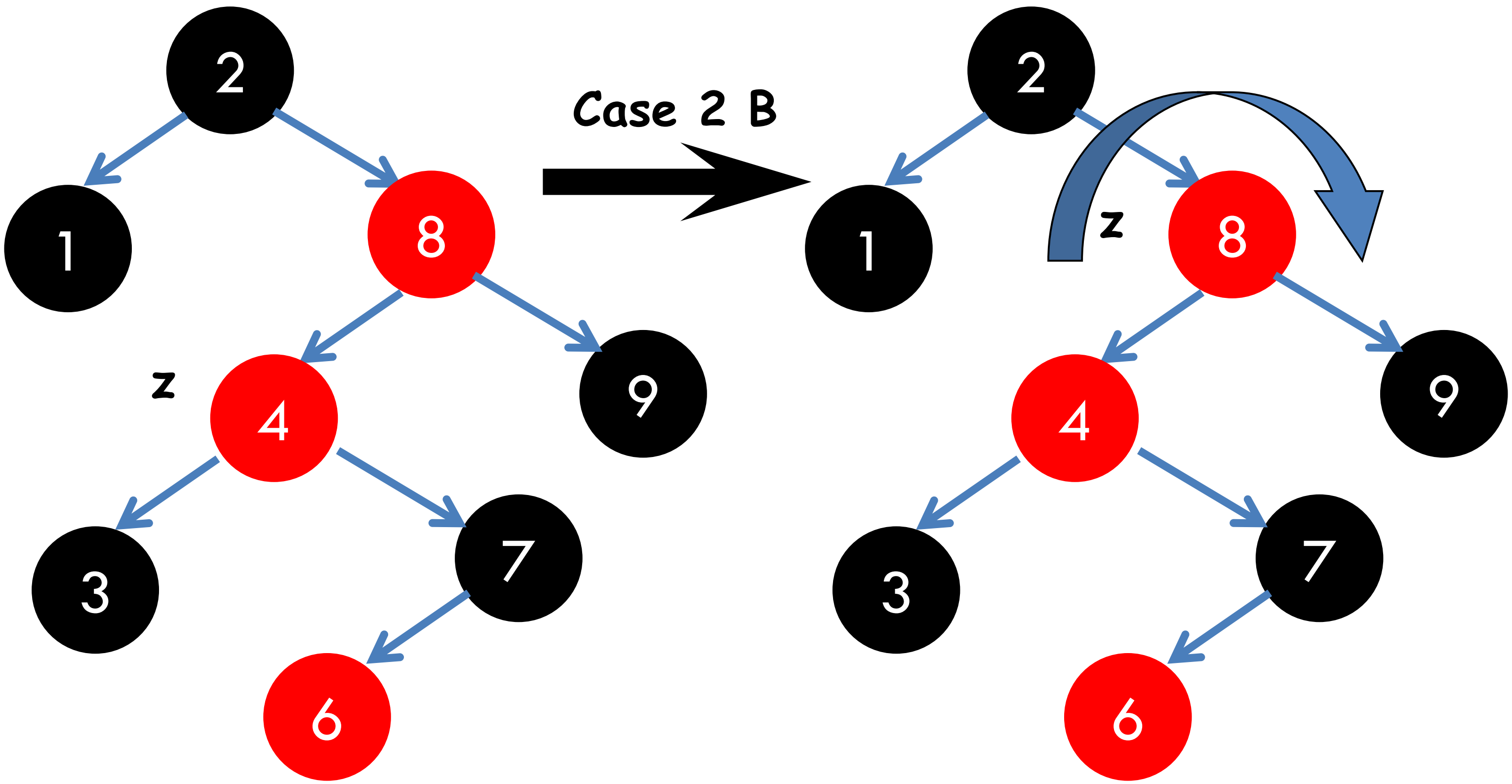
Learn DAA: From B K Sharma

Case 2B

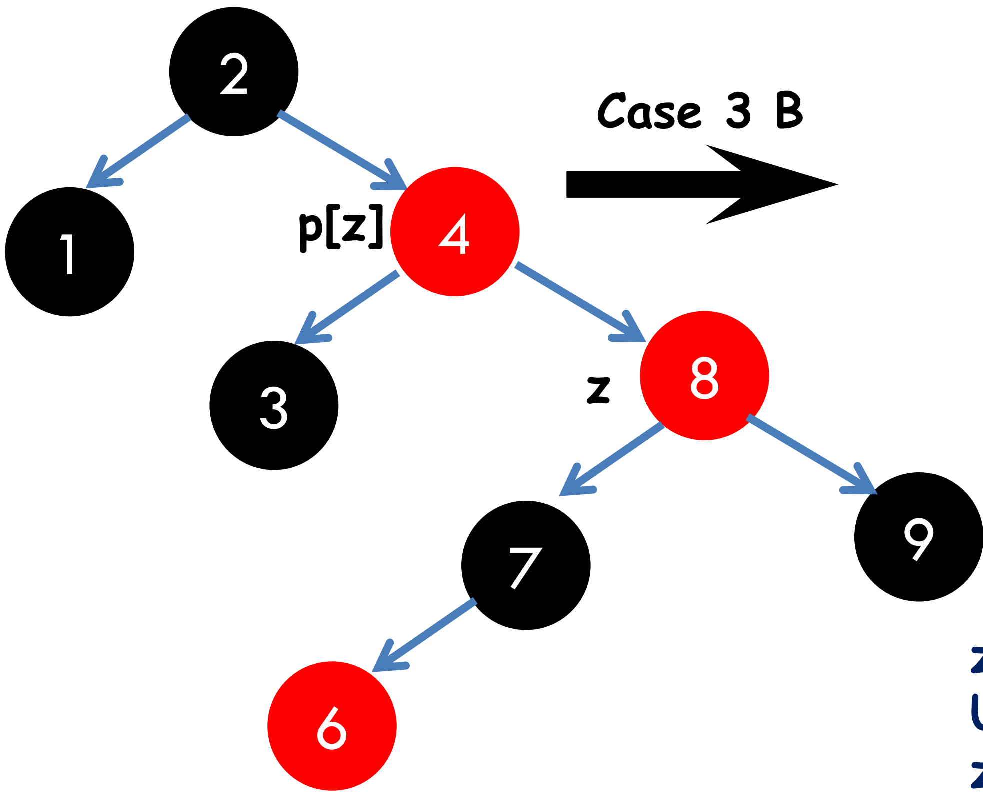
Solution: Right Rotation

$z \leftarrow p[z]$

RIGHT –ROTATE(T, z)



Learn DAA: From B K Sharma



z and p[z] are both red
Uncle y is black
z is the right child of p[z]

Learn DAA: From B K Sharma

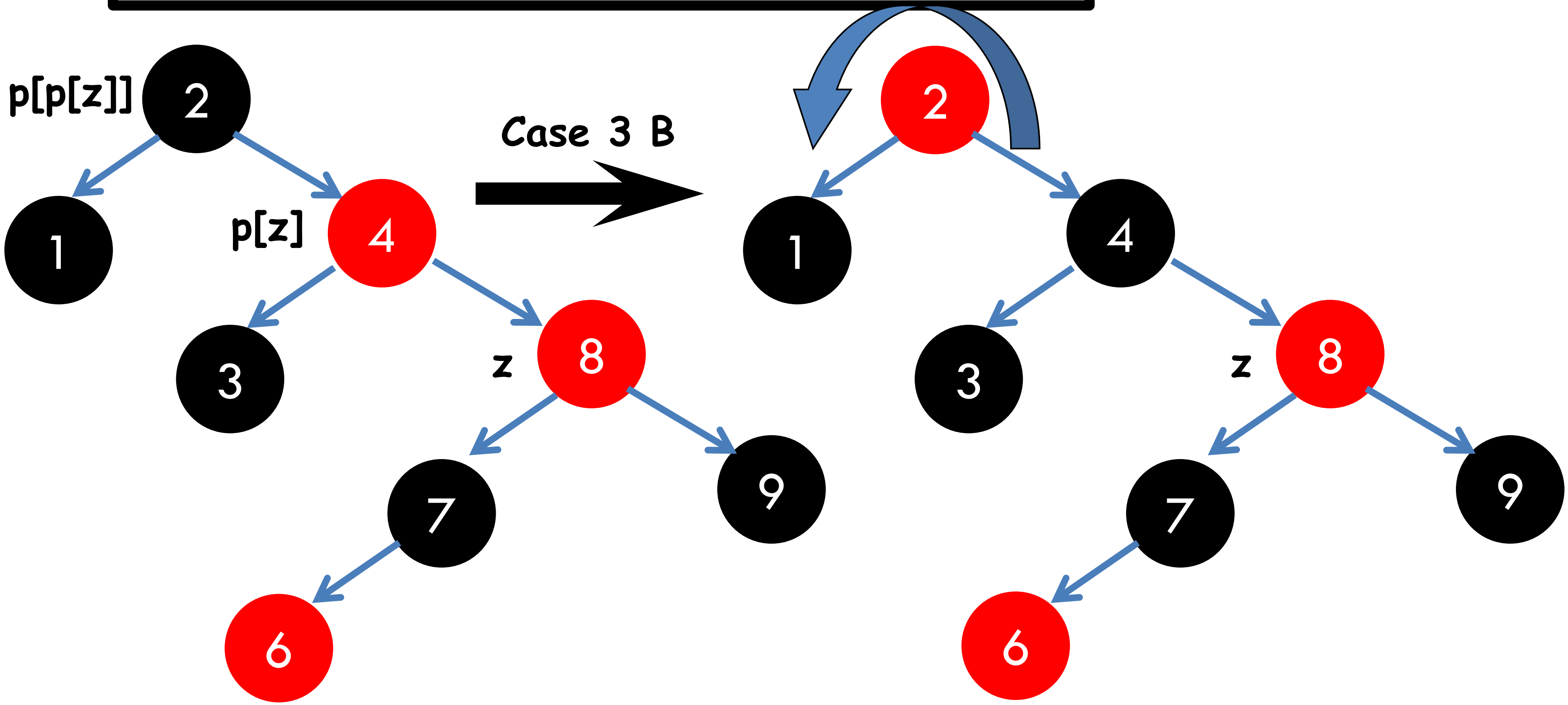
Case 3B

Solution: Recolor + LEFT-ROTATE AROUND p[p[z]]

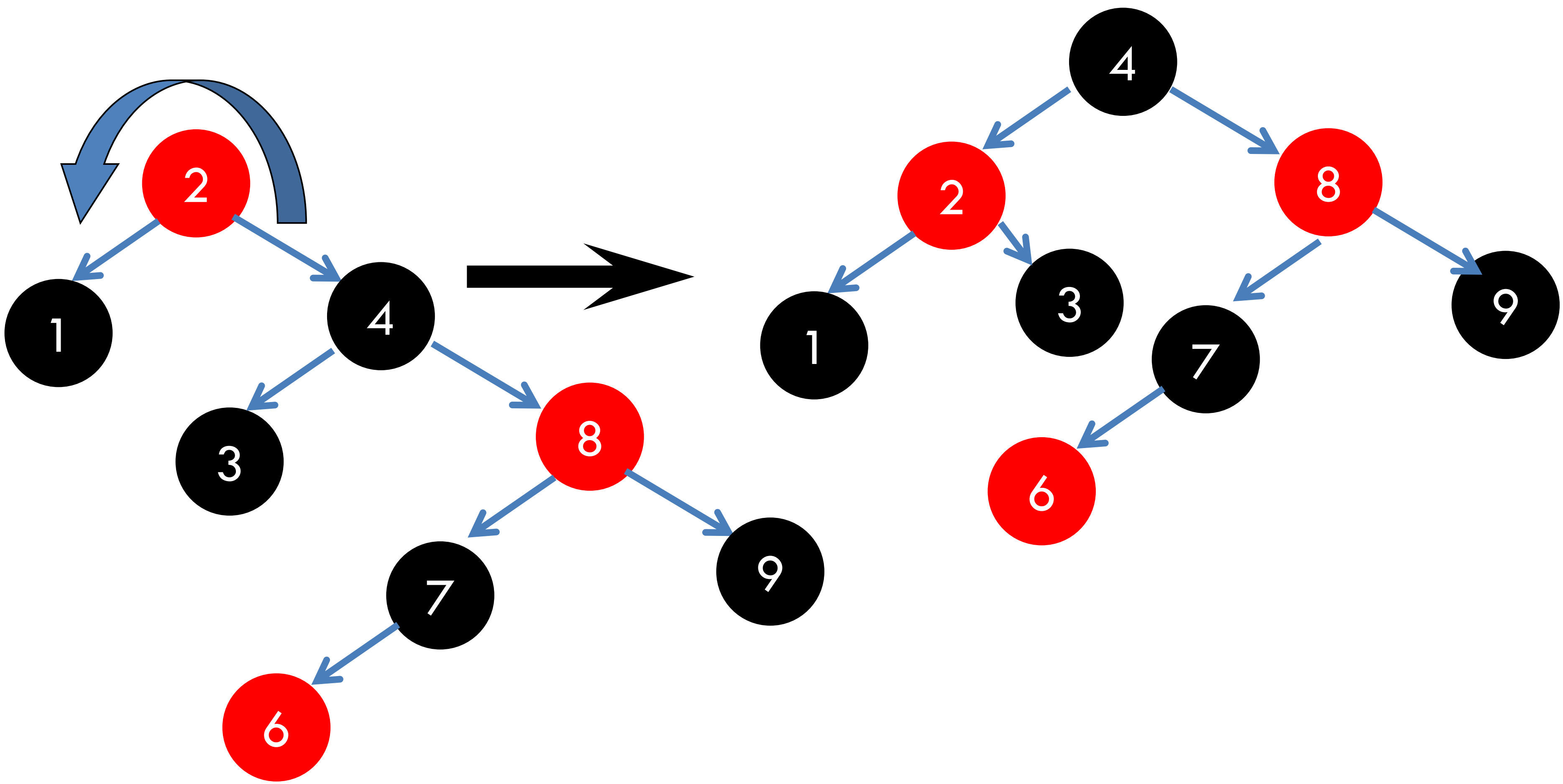
color[p[z]] $\leftarrow$ BLACK

color[p[p[z]] $\leftarrow$ RED

LEFT-ROTATE(T, p[p[z]])



Learn DAA: From B K Sharma



RB Properties Affected by Insert

1. Every node is either red or black OK!

2. The root is black If z is the root ⇒ not OK

3. Every leaf (NIL) is black OK!

4. If a node is red, then both its children are black

If p(z) is red ⇒ not OK
z and p(z) are both red

OK!

5. For each node, all paths from the node to descendant leaves contain the same number of black nodes

Why is the *colour* of new node made red, why not black?

The color of new node(z) is made red because we have to maintain the property (5) which says

> For each node, all paths from that node to descendent leaves contain the same number of black nodes