SEARCH TREE ADT

BST

AVL

AVL TREE

ADELSON-VELSKII AND LANDIS' TREE

AVL TREE

A binary search tree with a balance condition.

AVL TREE

For every node in the tree, the height of its left and right subtrees can differ by at most 1.

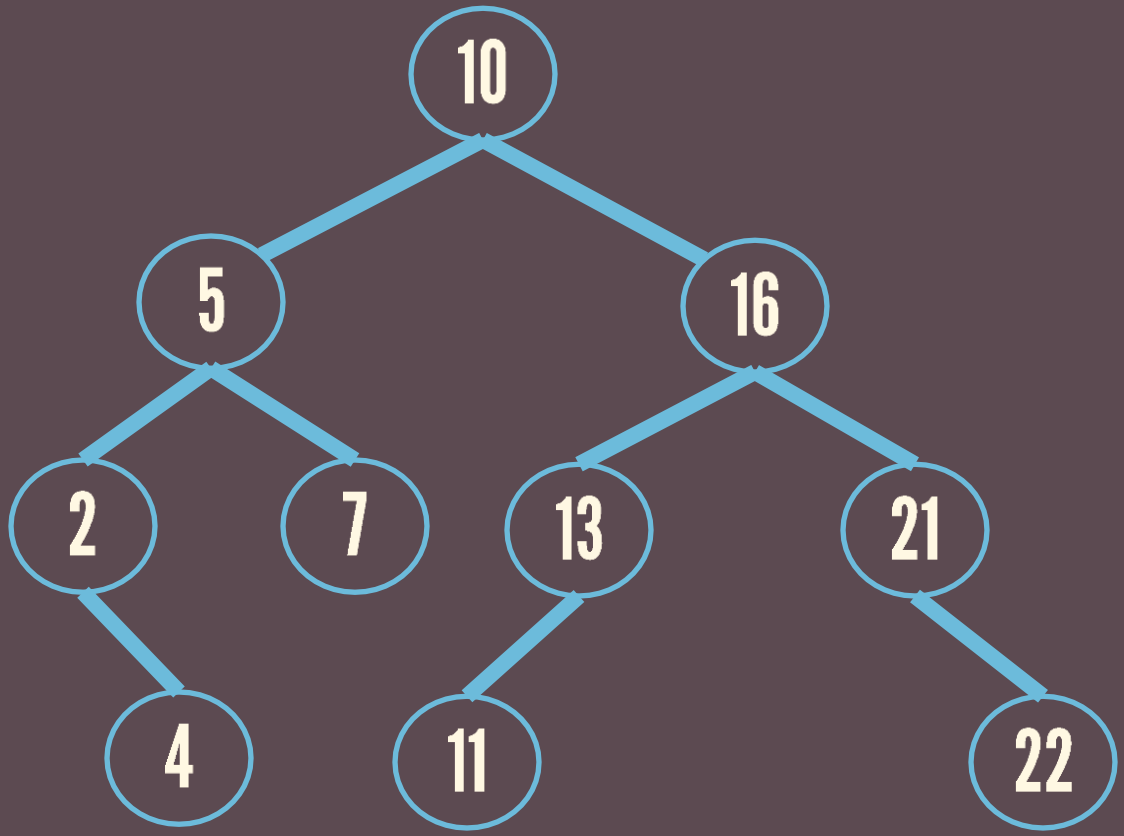
AVL TREE

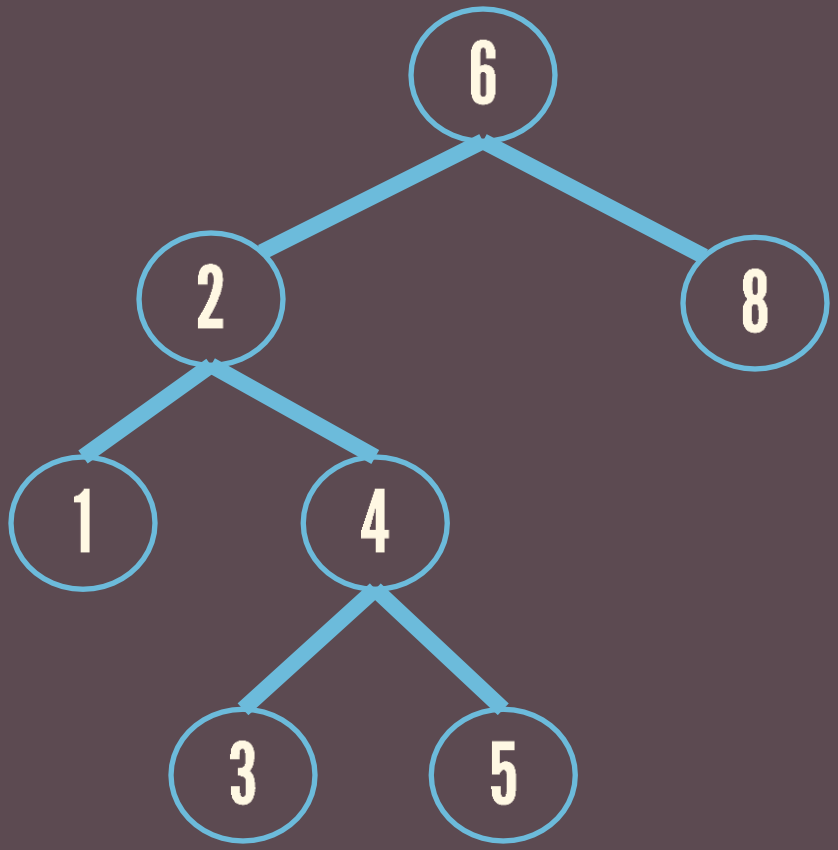
If anytime they differ by more than one, rebalancing is done to restore this property.

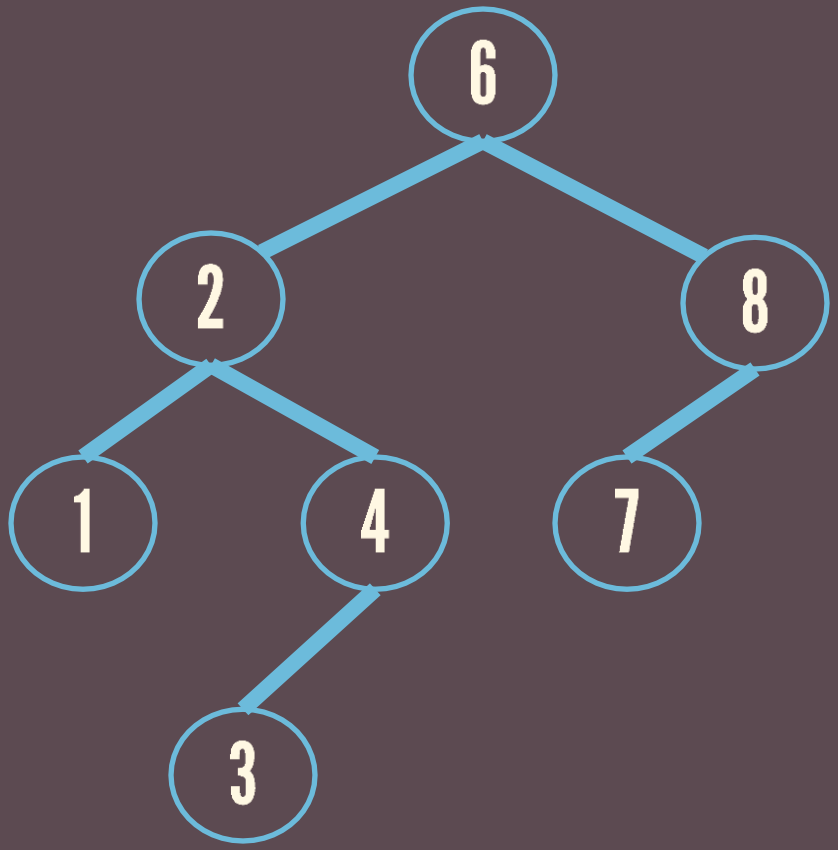
NOTE

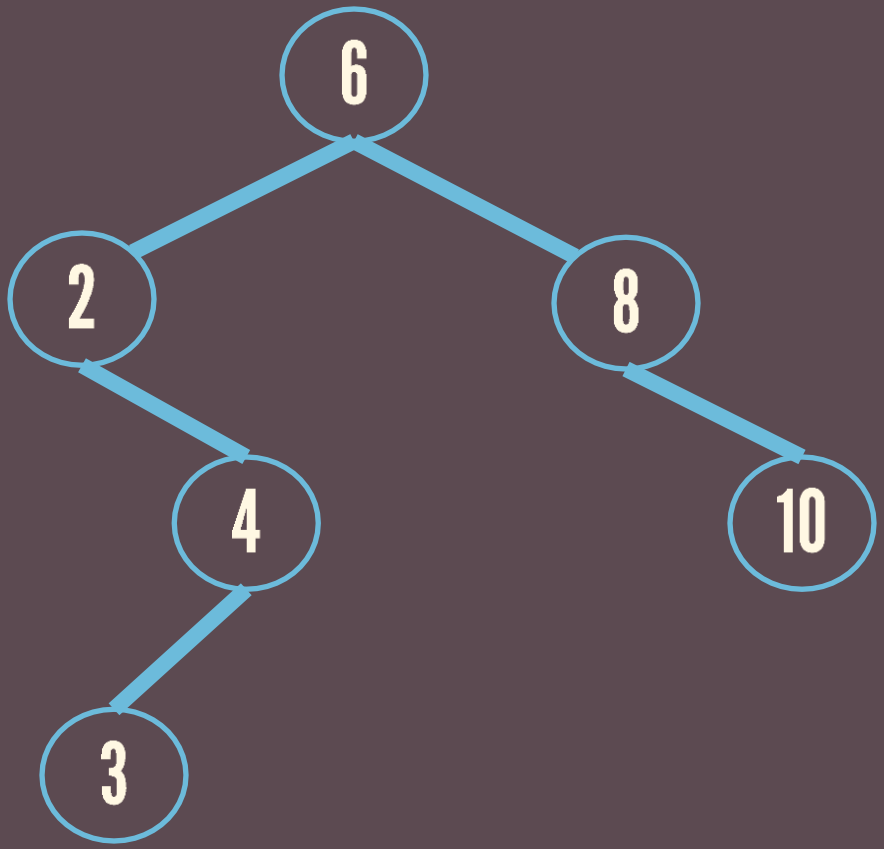
The height of an empty tree is -1.

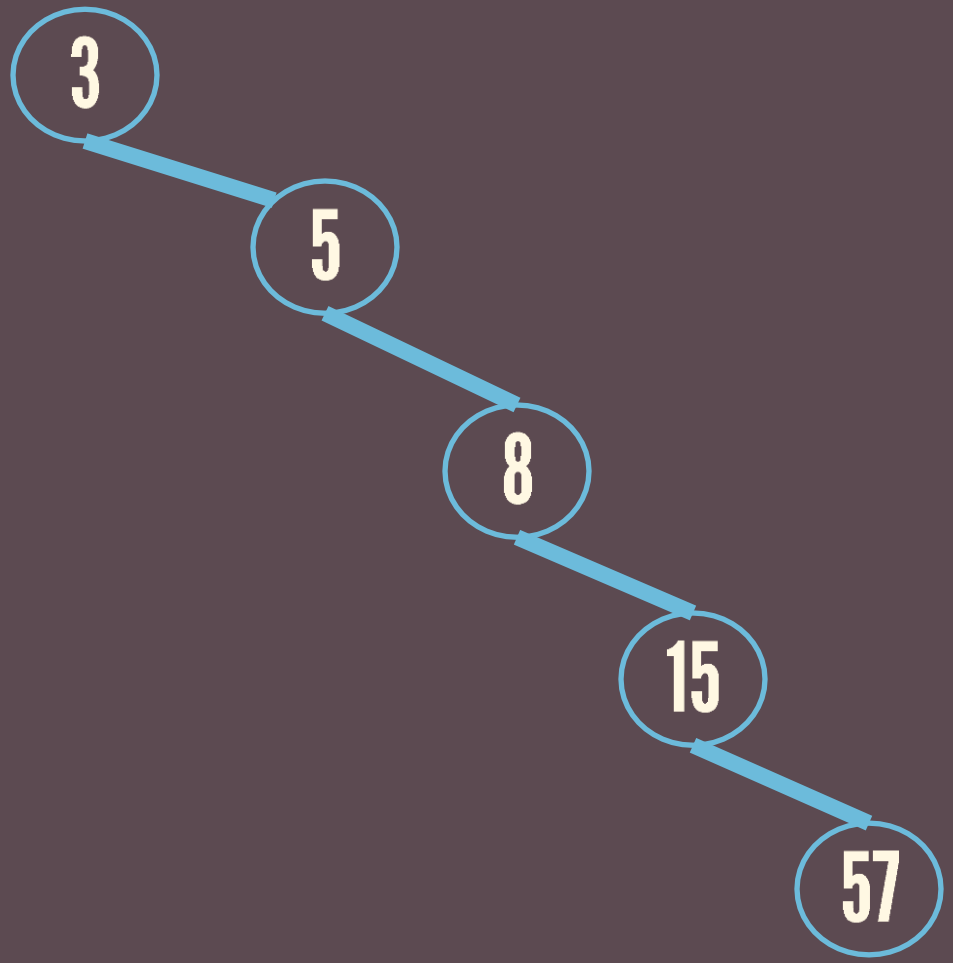
```
class AVLNode:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data
        self.height = 1
```

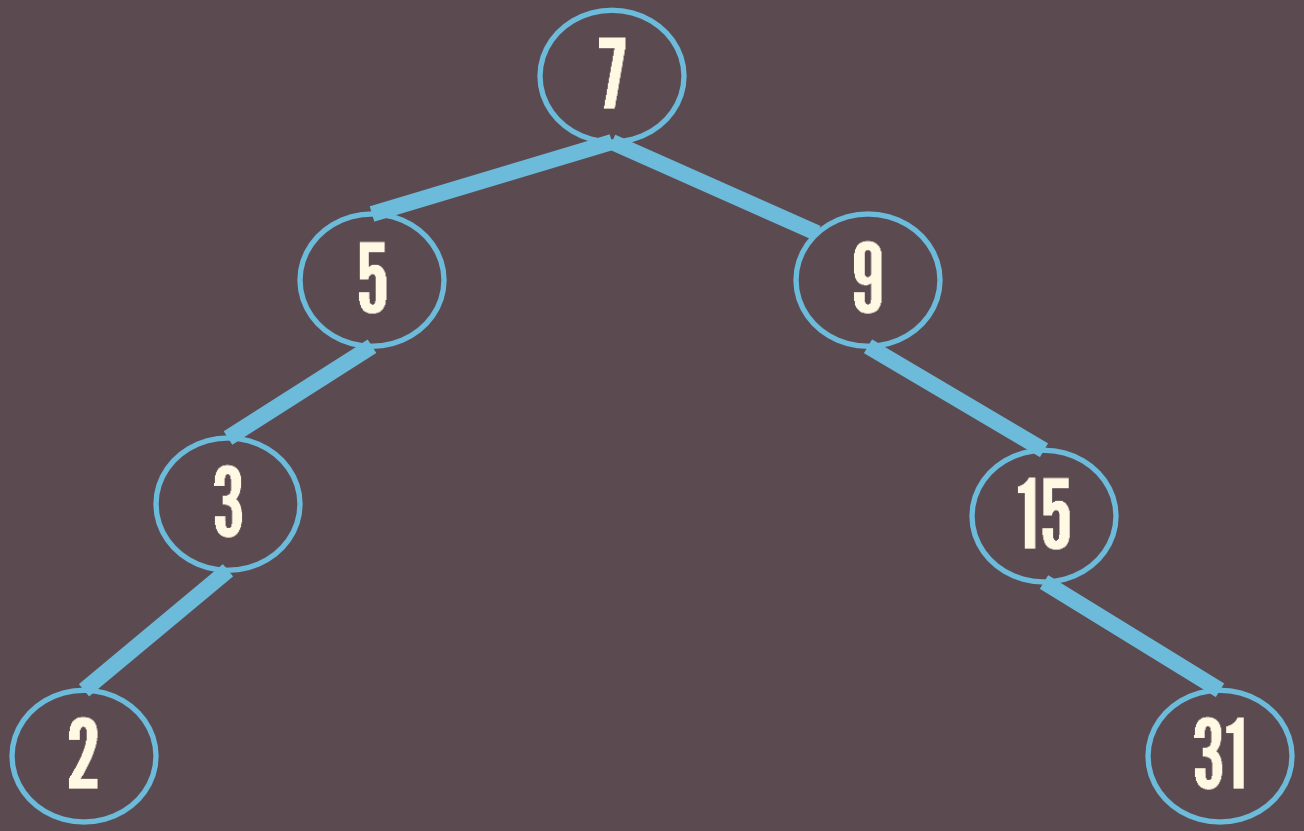

10
5
16
2
7
13
21
4
11
22


6
2
8
1
4
3
5


6
2
8
1
4
7
3


6
2
8
4
10
3


3
5
8
15
57


7
5
9
3
15
2
31

AVL

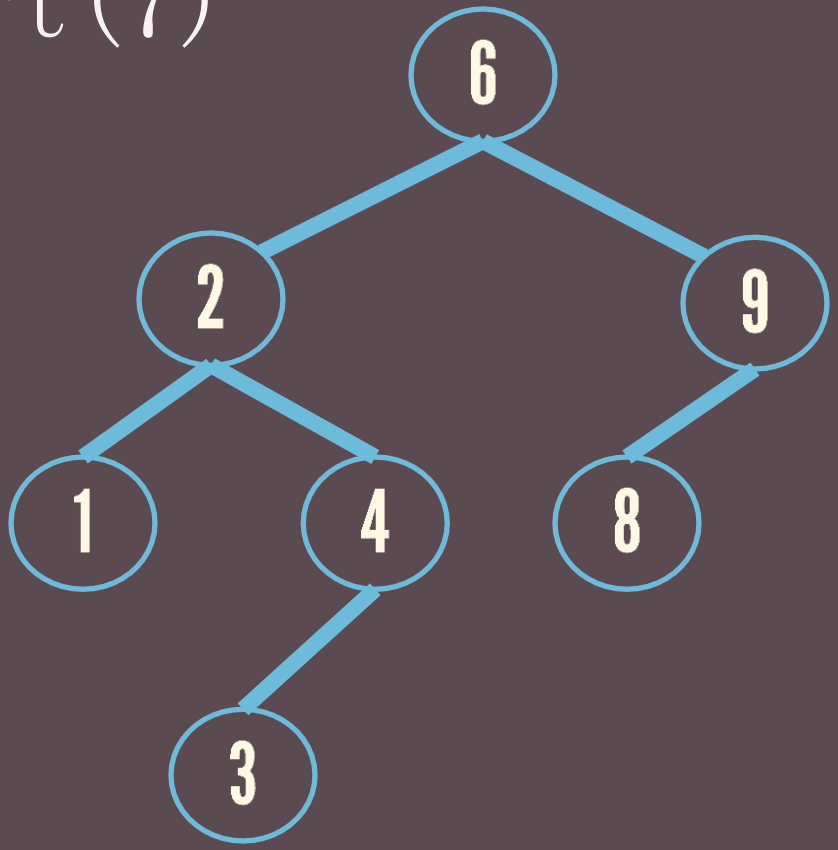
OPERATIONS

find
insert (with rotations)
delete (with rotations)
minimum
maximum
successor
predecessor

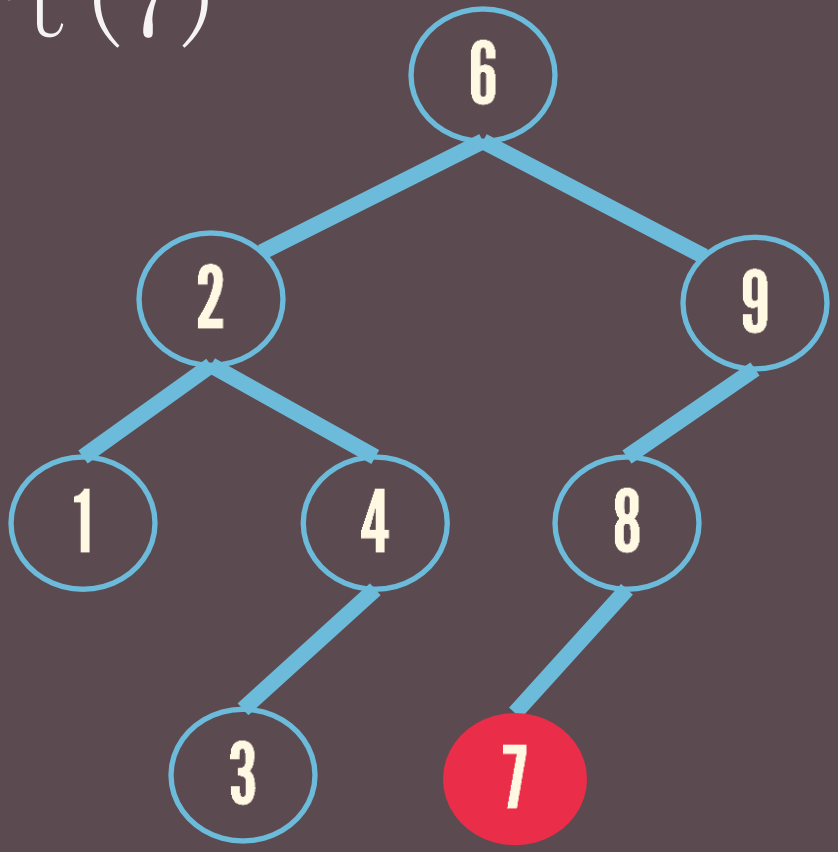
ROTATIONS

Rotations are done to maintain the AVL property.

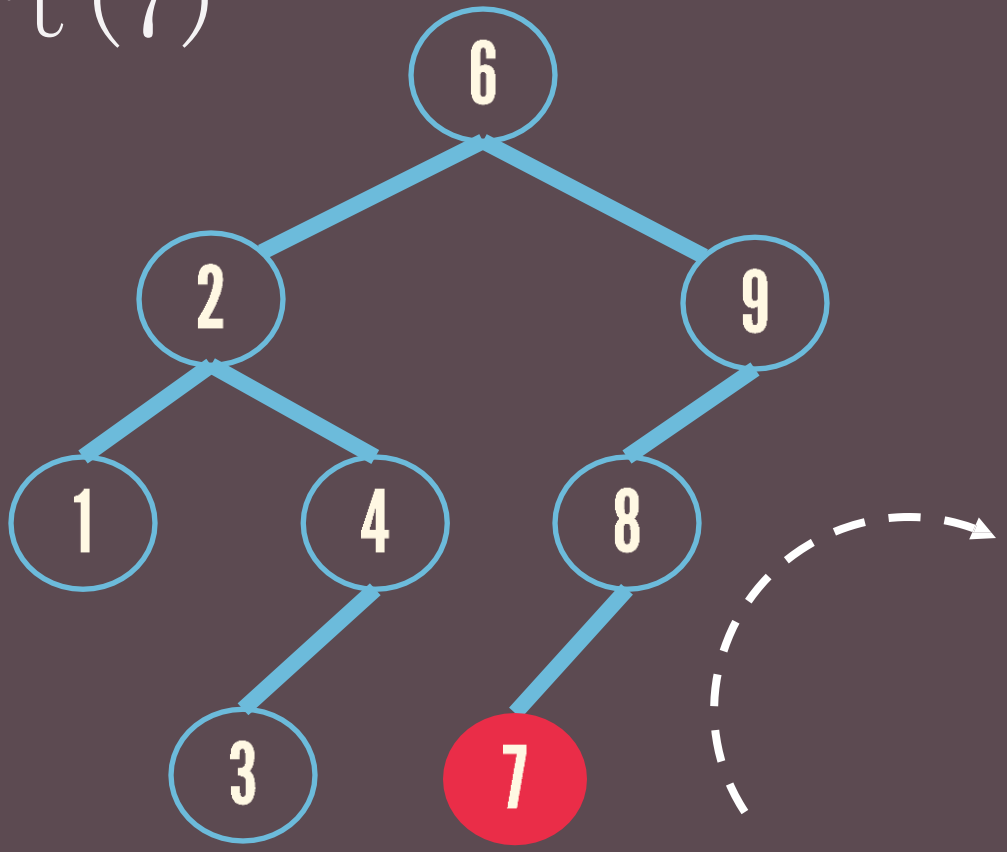
insert(7)



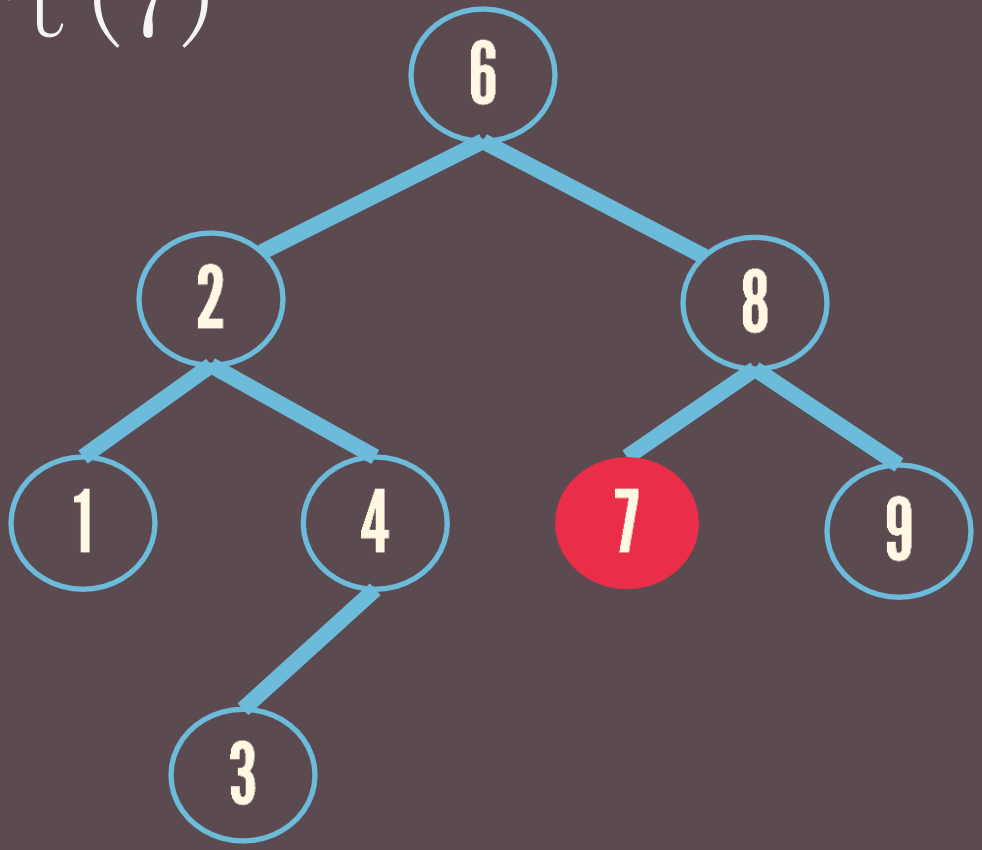
insert(7)



insert(7)



insert(7)



ROTATIONS

INSERT OPERATION

Single:

Left Rotate
Right Rotate

ROTATIONS

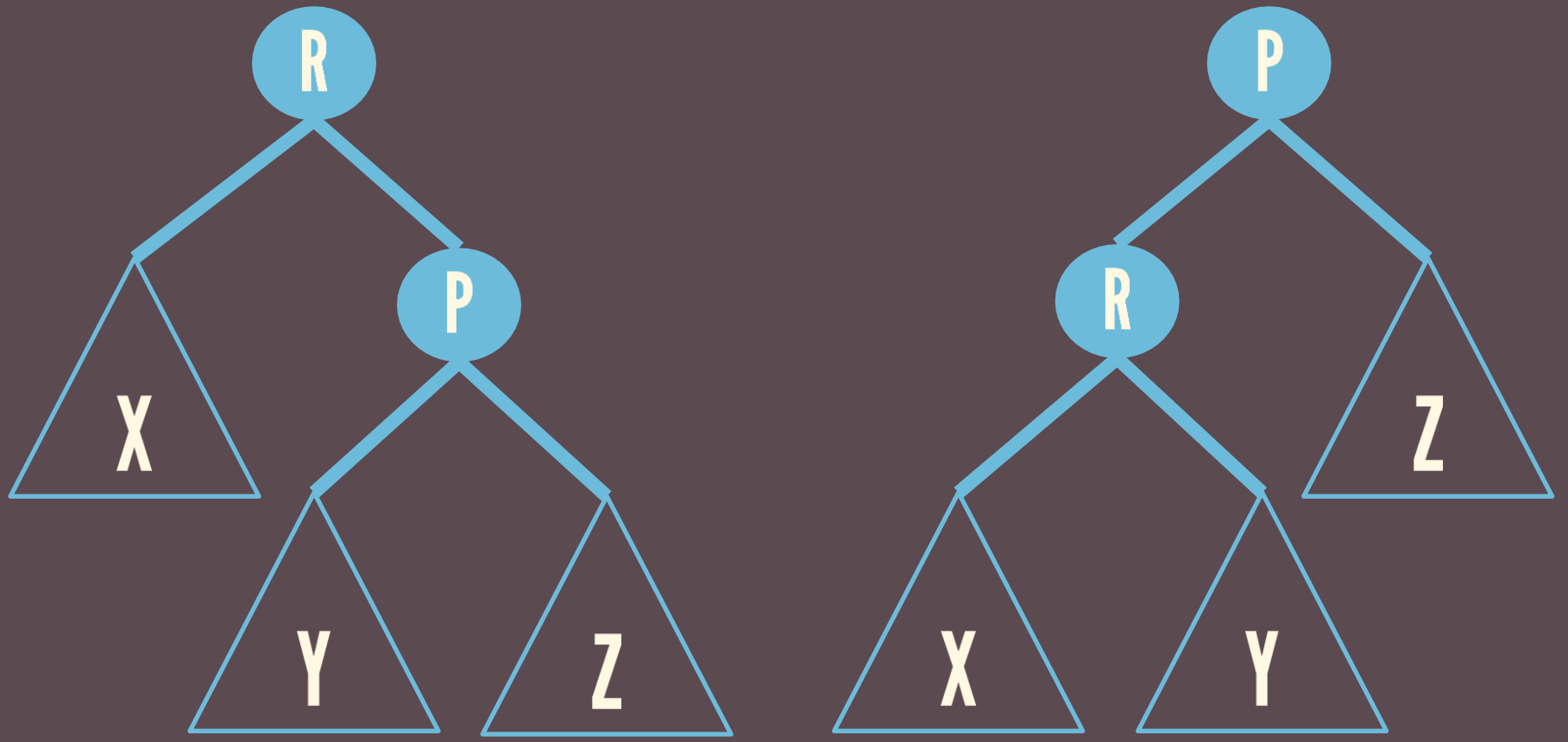
INSERT OPERATION

Double:

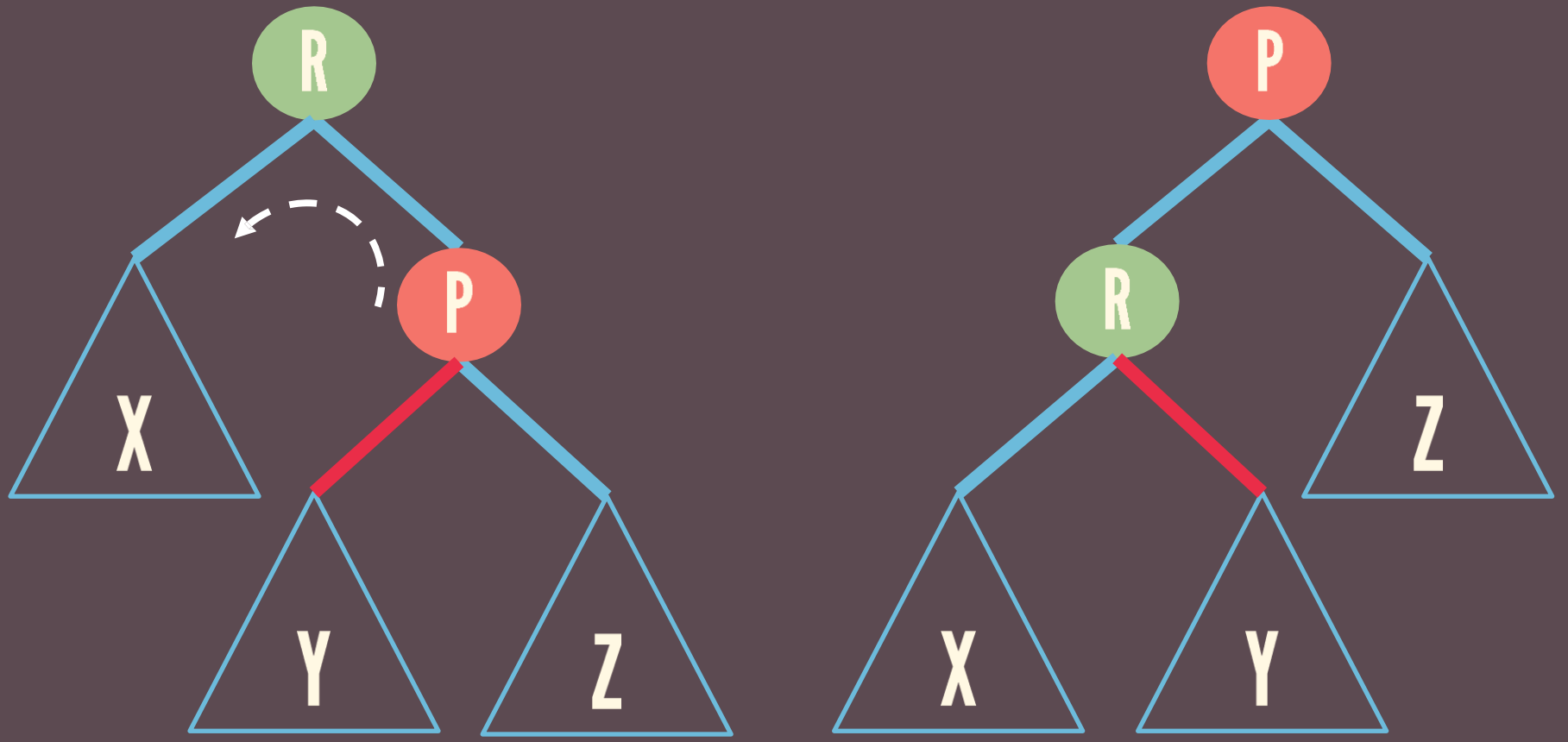
Left-right Rotate
Right-left Rotate

ROTATIONS

ILLUSTRATED

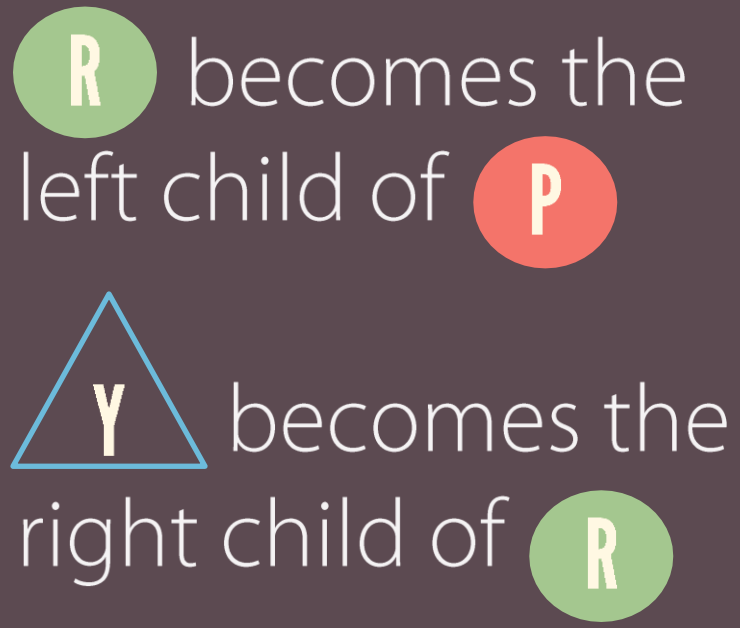


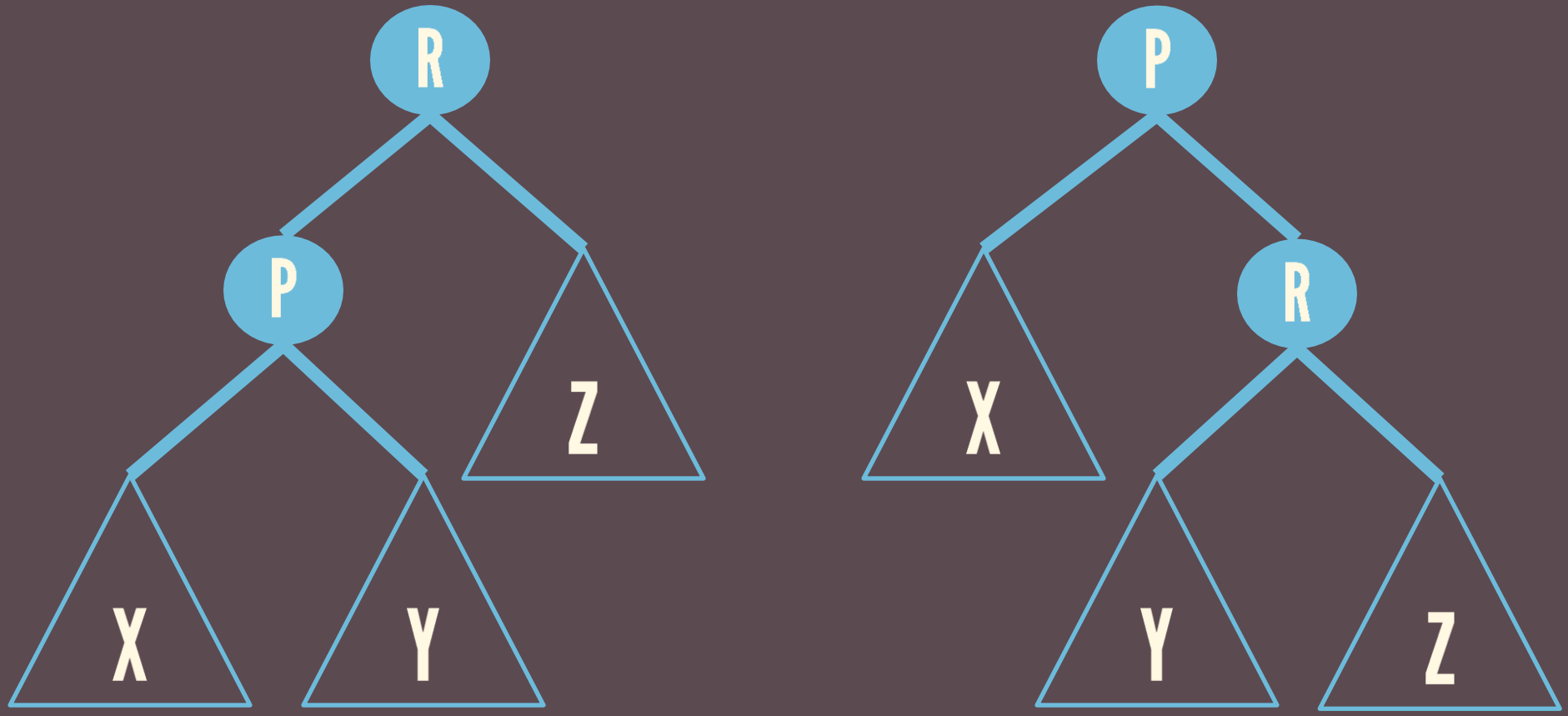
LEFT ROTATE



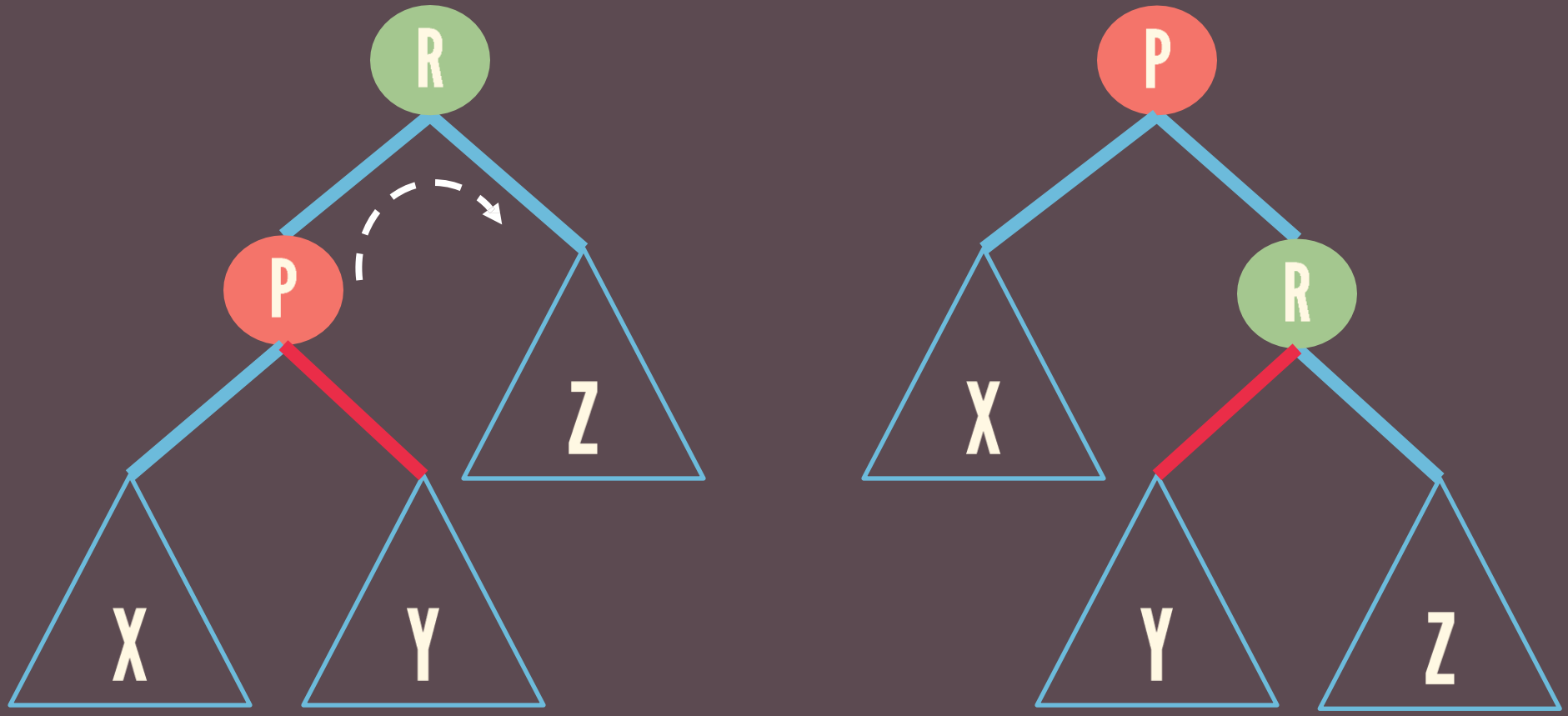
LEFT ROTATE

LEFT ROTATE



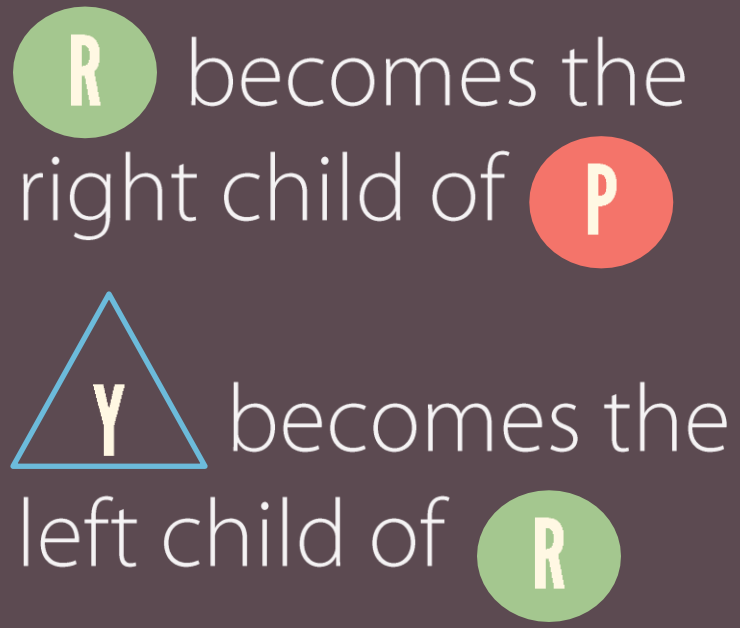


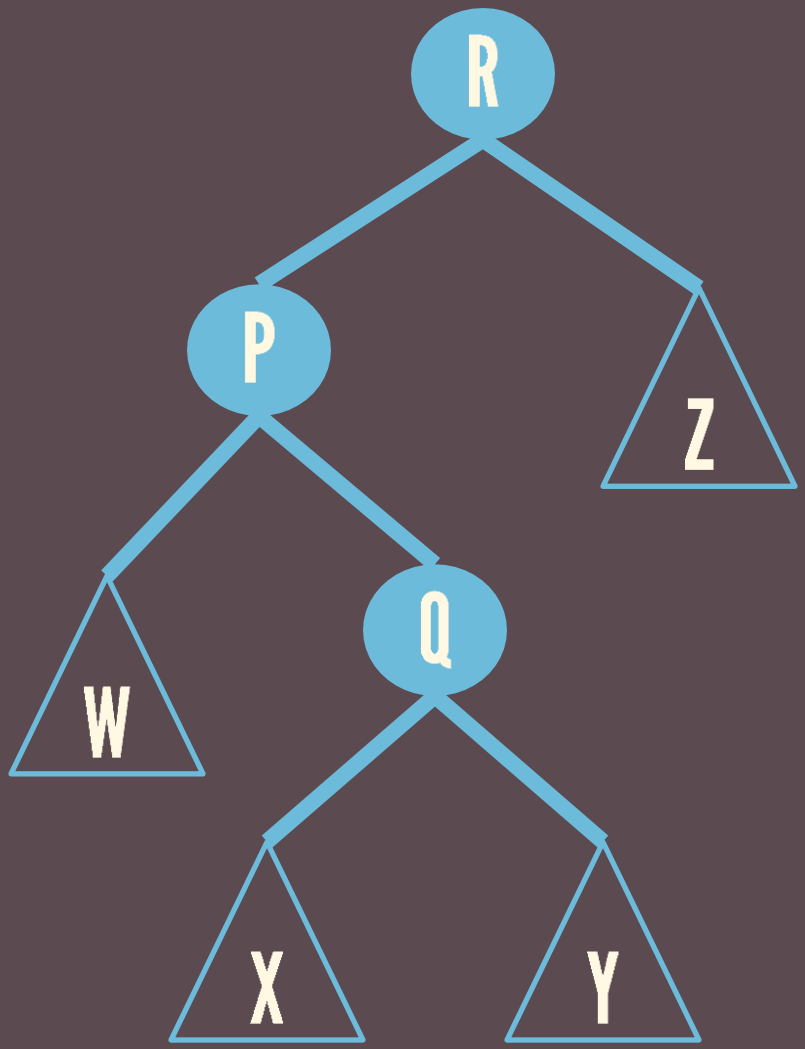
RIGHT ROTATE

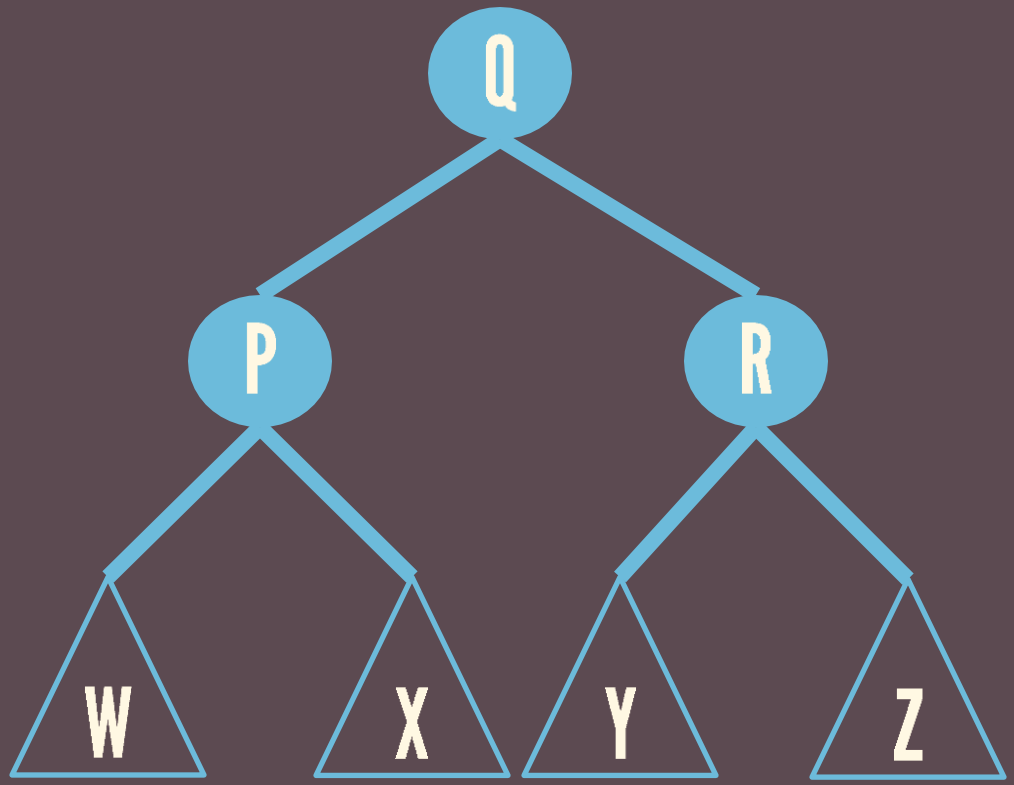


RIGHT ROTATE

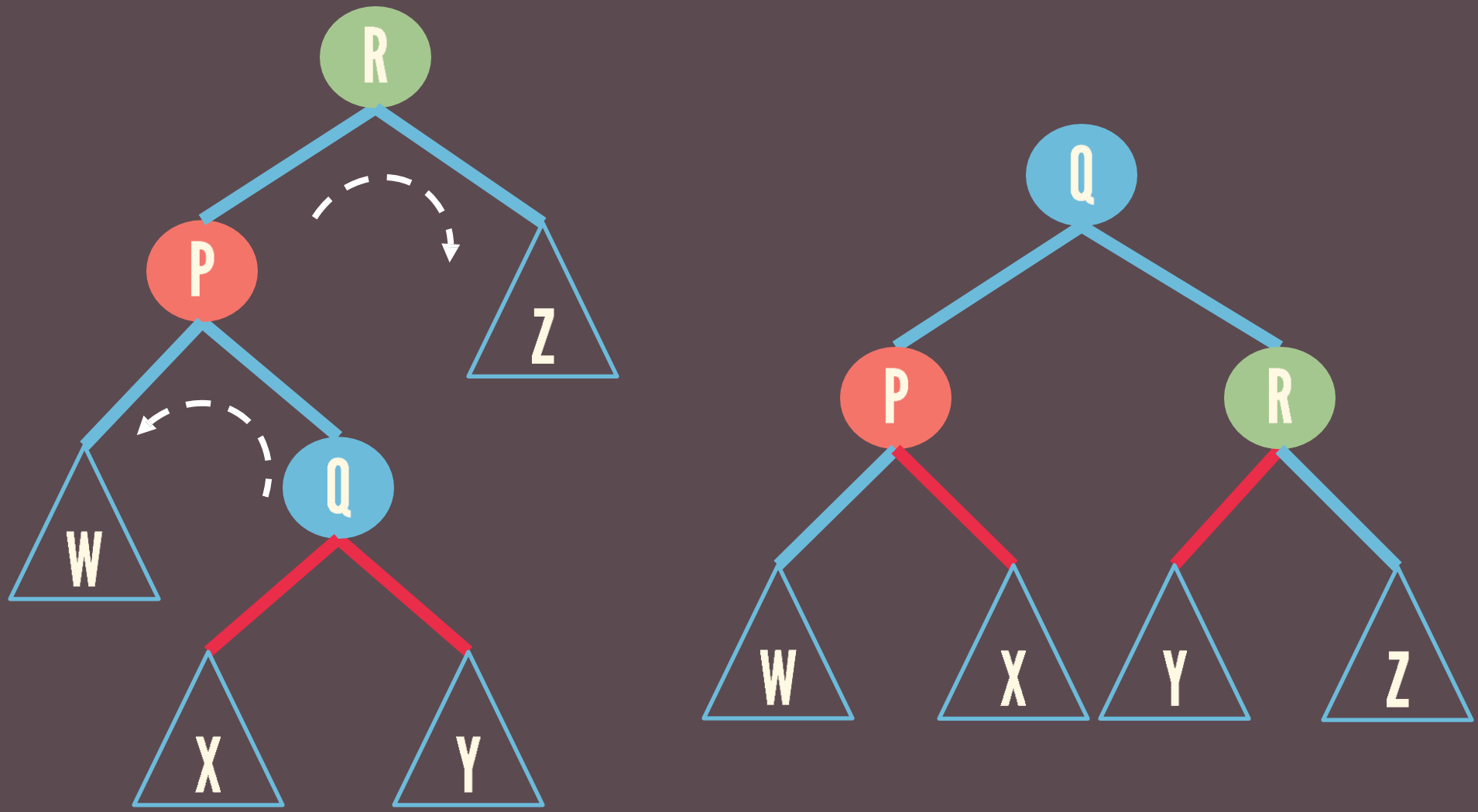
RIGHT ROTATE



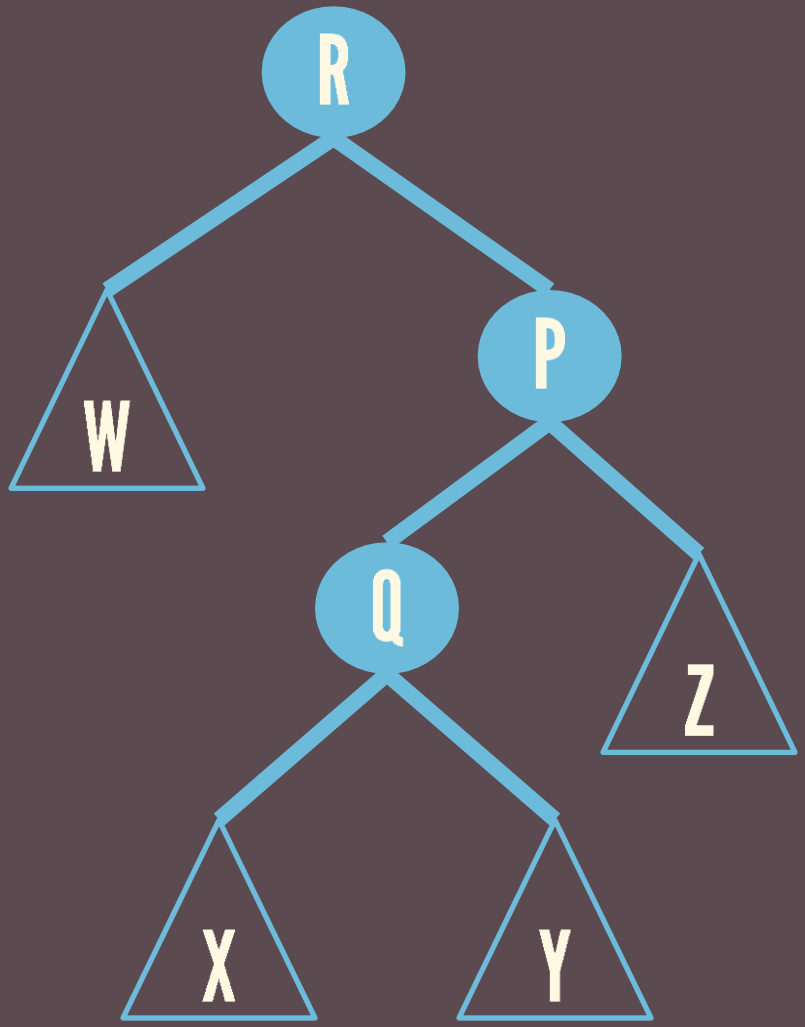


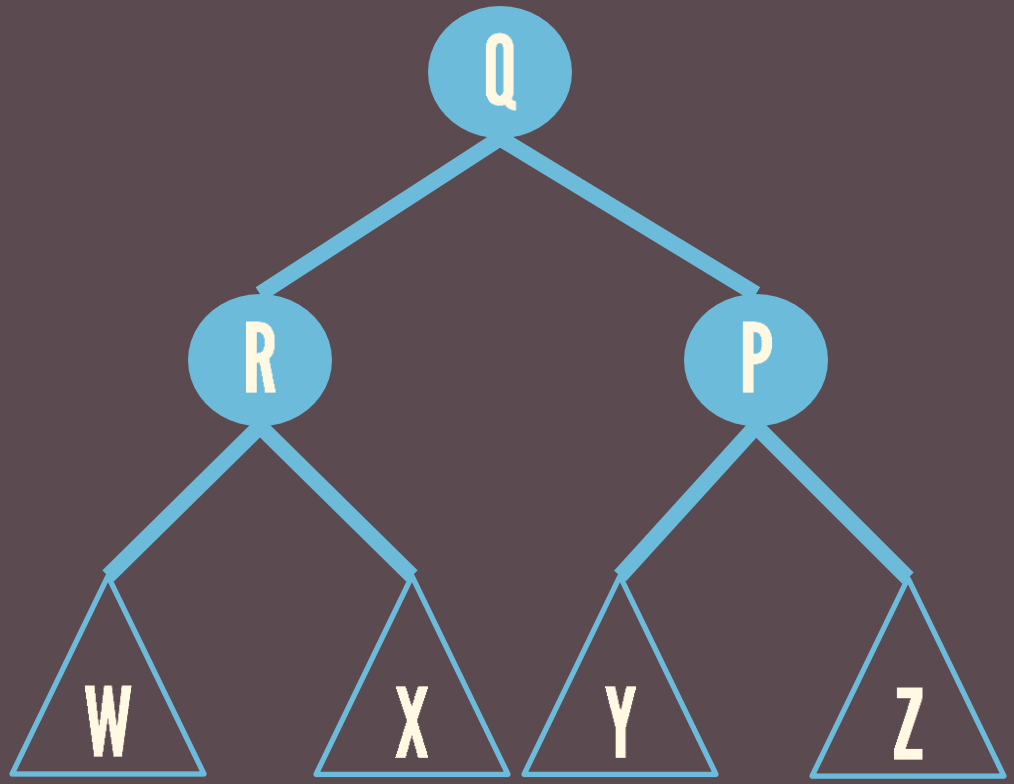


LEFT-RIGHT ROTATE

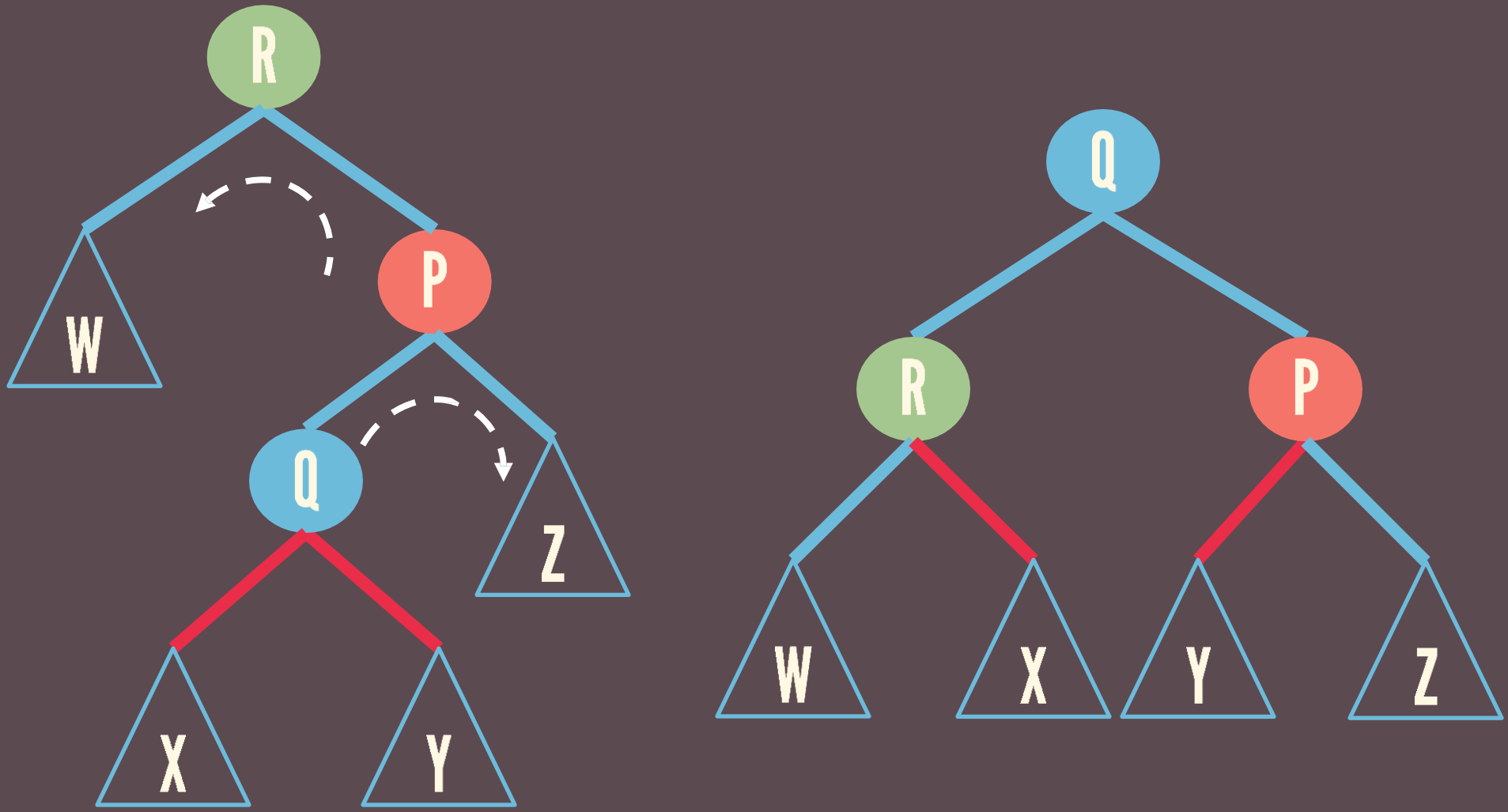


LEFT-RIGHT ROTATE





RIGHT-LEFT ROTATE

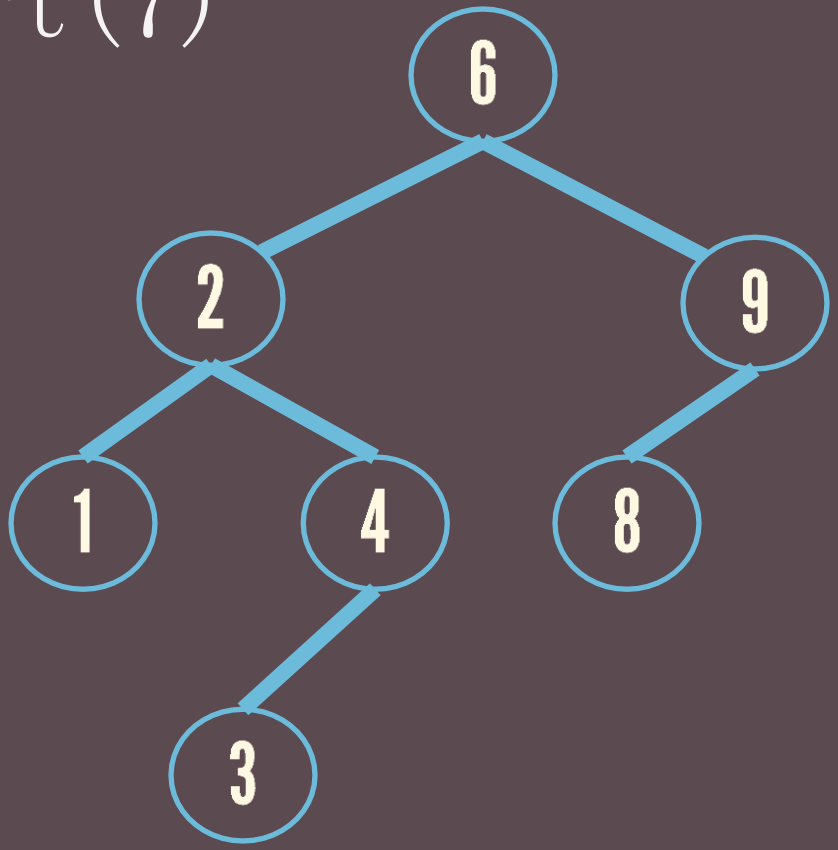


RIGHT-LEFT ROTATE

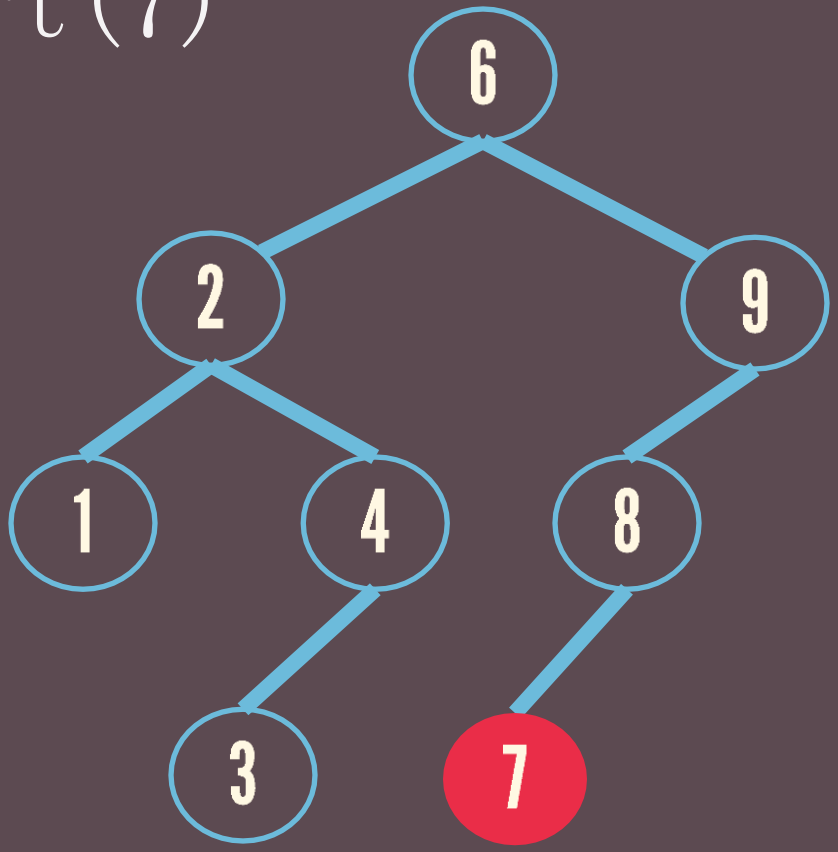
WHEN TO USE WHAT ROTATION

4 CASES

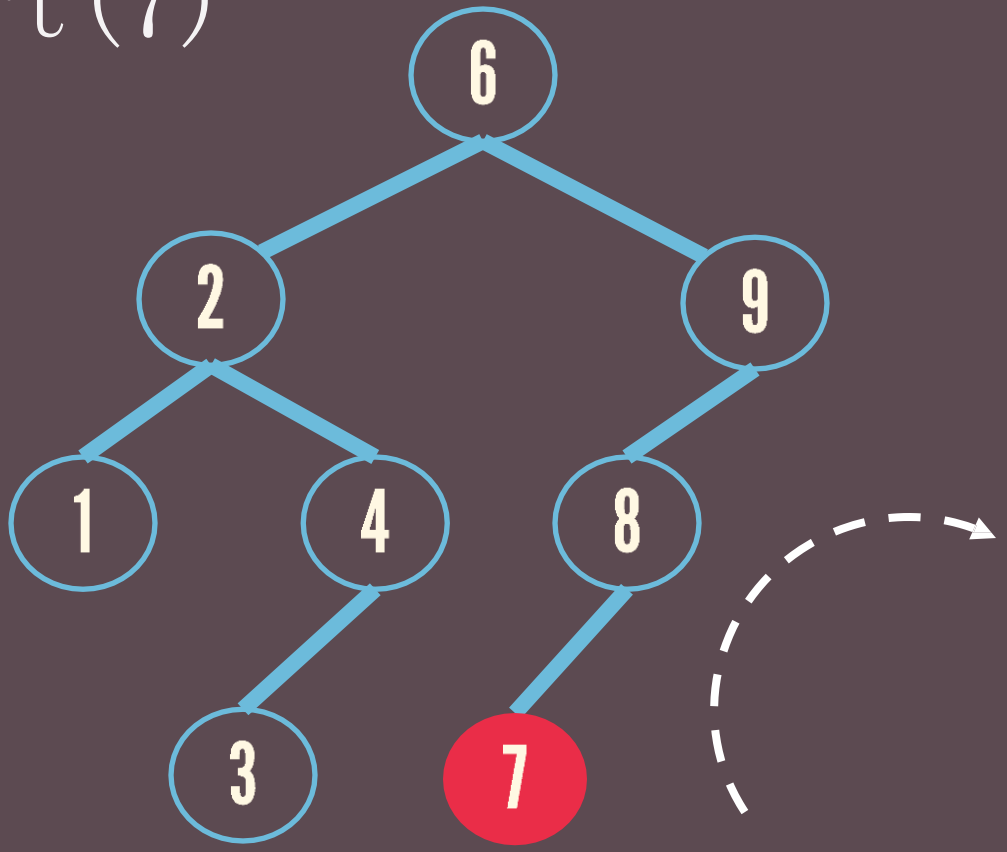
insert(7)



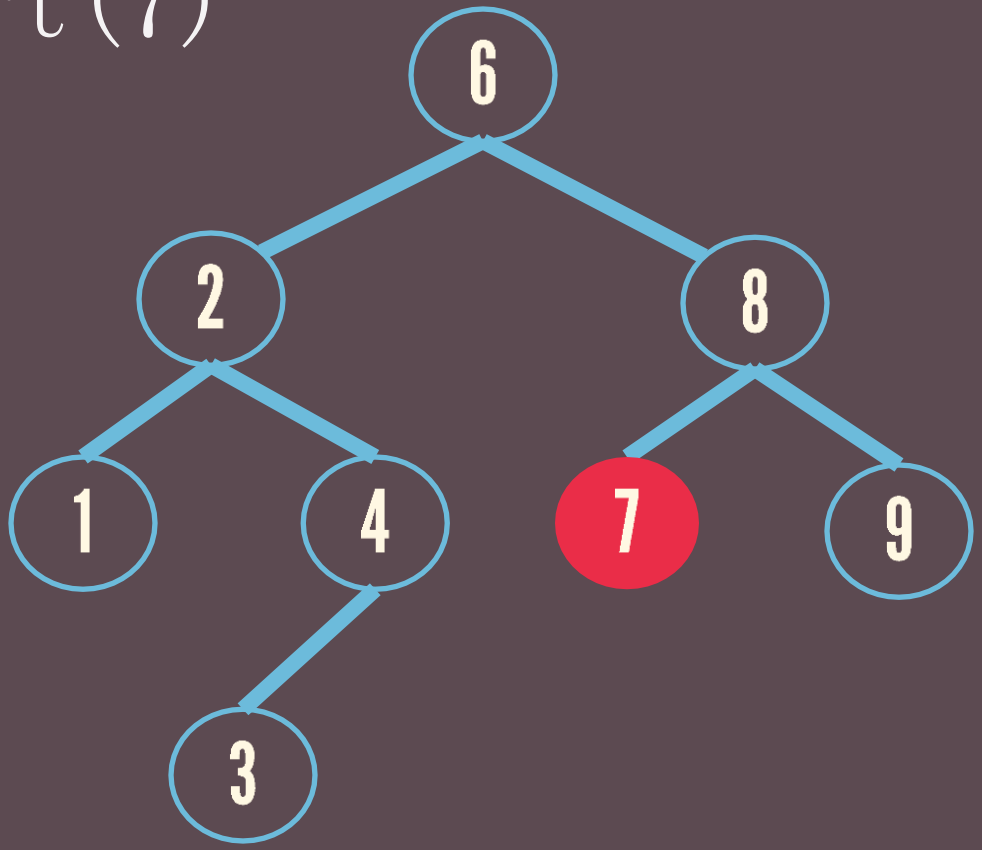
insert(7)



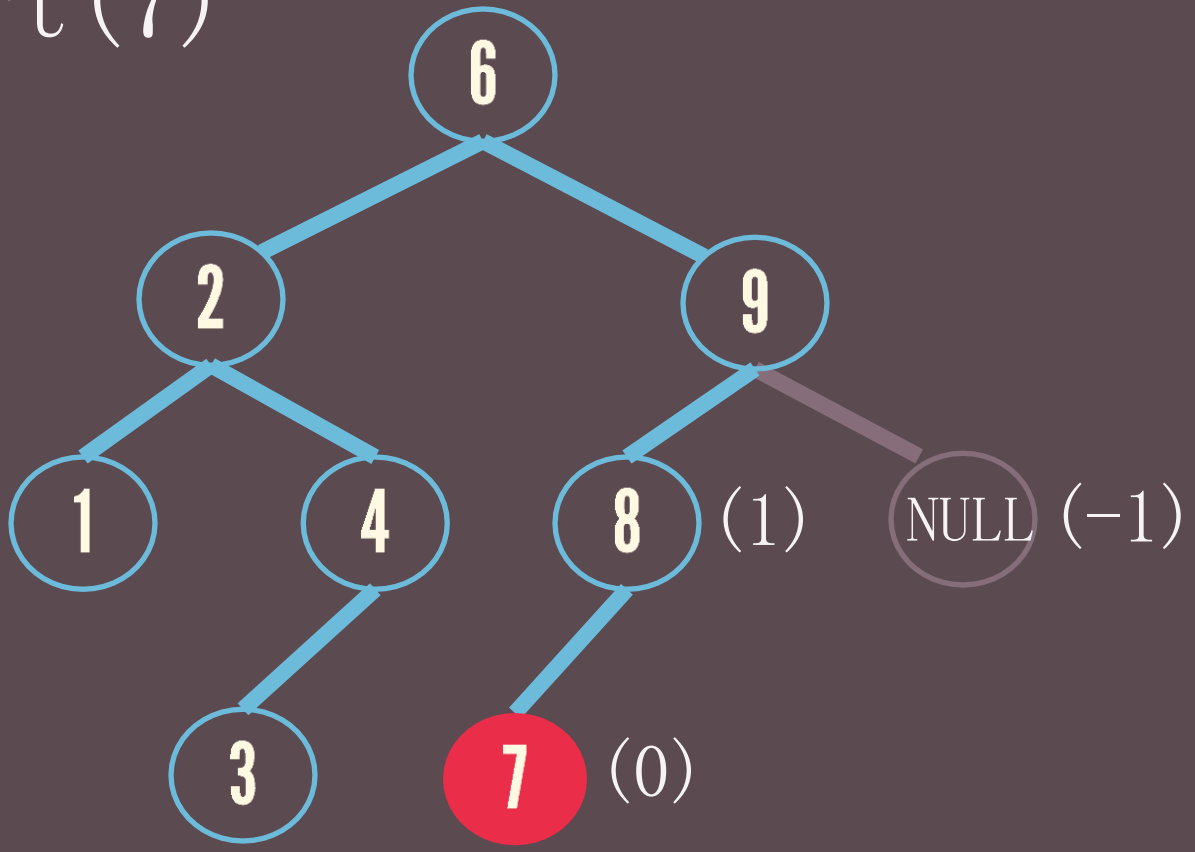
insert(7)



insert(7)



insert(7)



```
insertNode(){

   algorithm for inserting a node.
   update height of nodes.
   fixUp()

}
```

```
fixUp(){

   start at the node inserted and travel
   up the tree:
      if an imbalance is found,
         check the four cases and do the
         appropriate rotation.
      update height of the nodes.

}
```

fixUp()

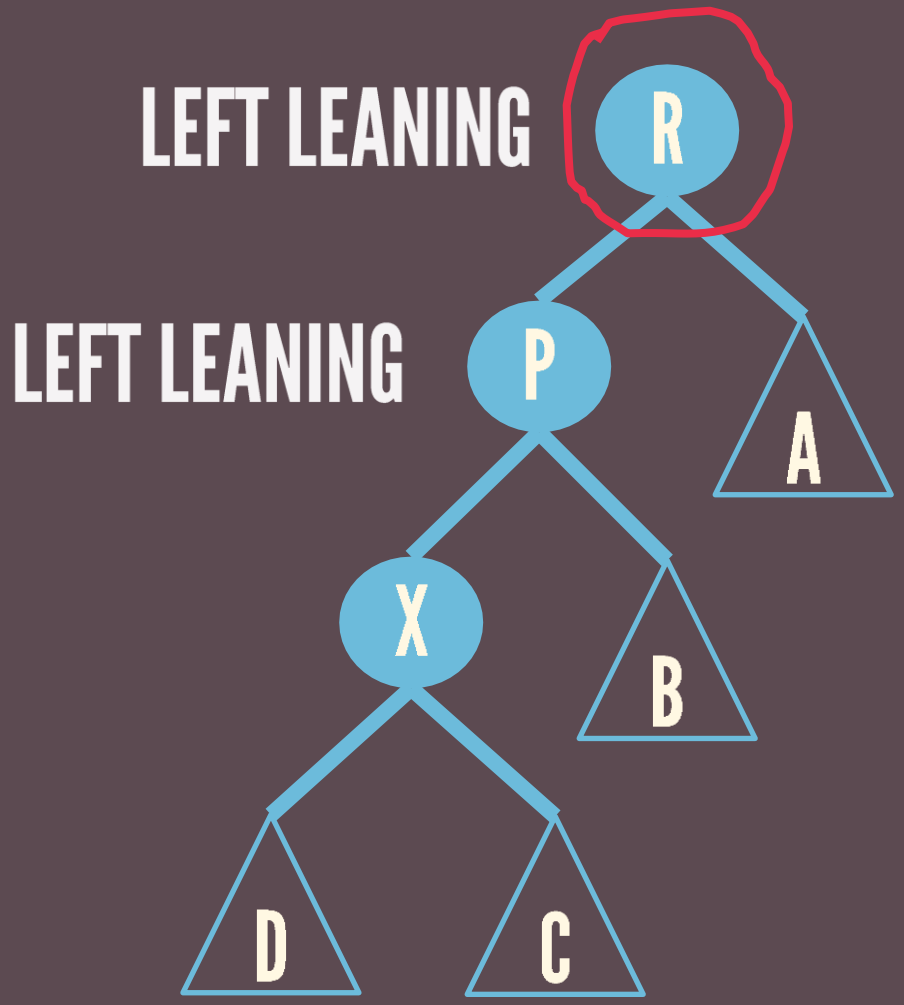
rotation is made where the imbalance is found.

#1

LEFT LEFT CASE

RIGHT ROTATE

#1

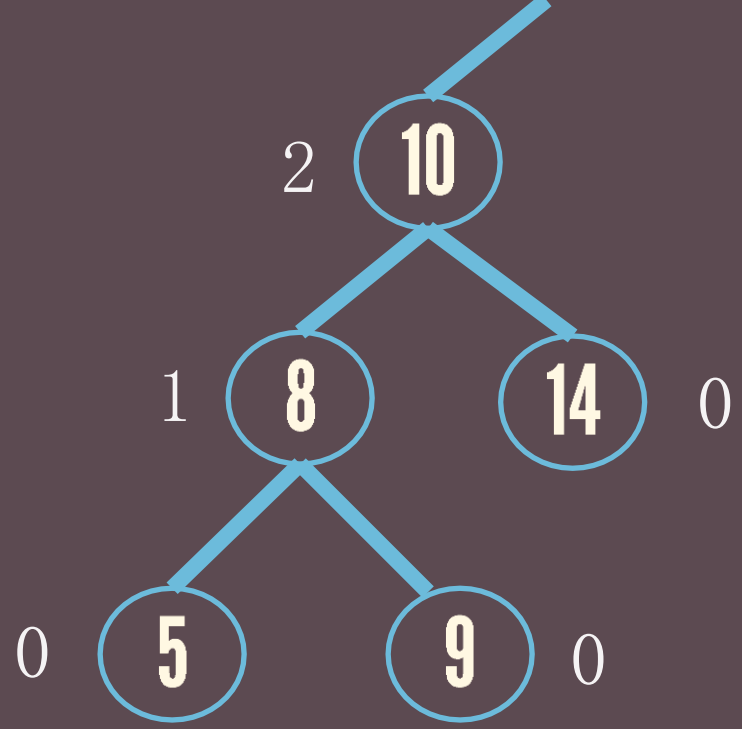


R – root

P – pivot

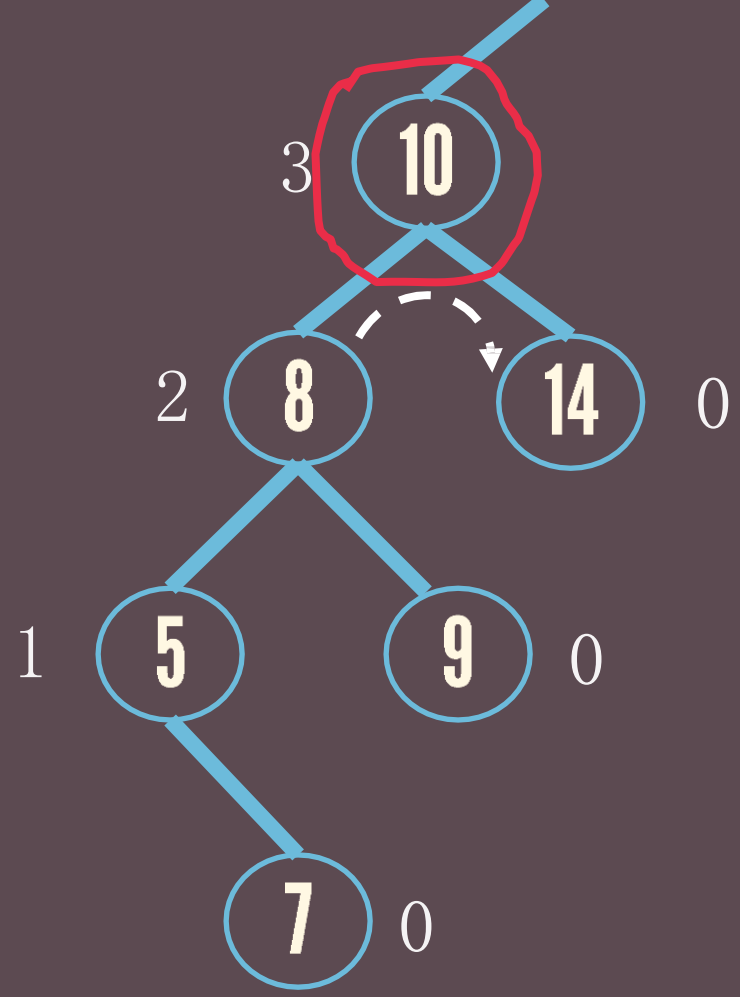
#1

```
fixUp(){

   start at the node inserted and travel
   up the tree:
      if an imbalance is found,
         if pivot is left leaning and
         root is left leaning
            do a right rotation on root.
      update height of the nodes.

}
```



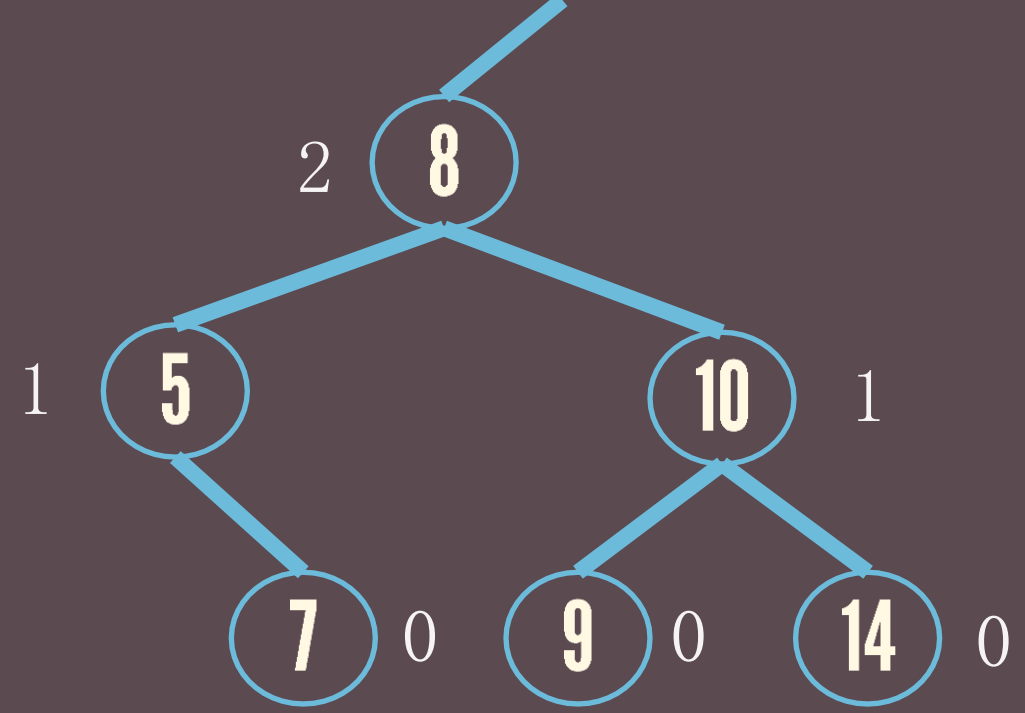
#1

insert(7)



#1

insert(7)



#1

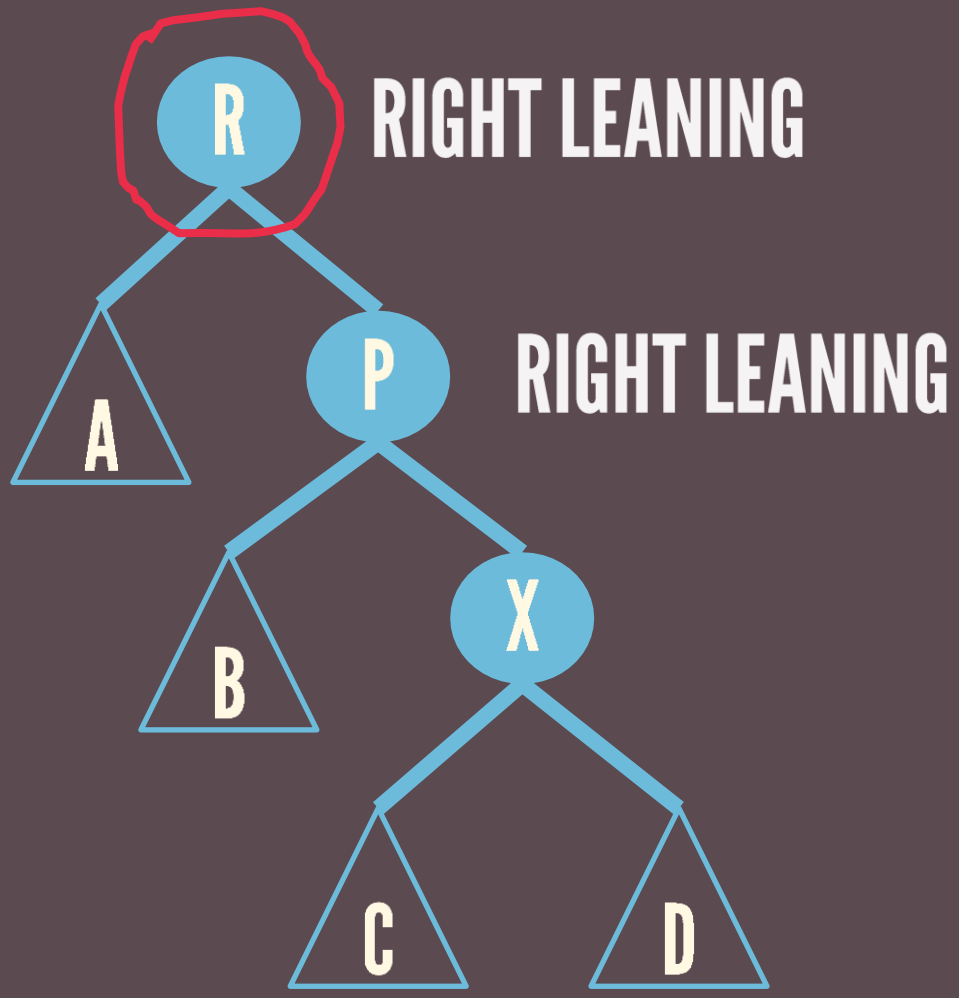
insert(7)

#2

RIGHT RIGHT CASE

LEFT ROTATE

#2

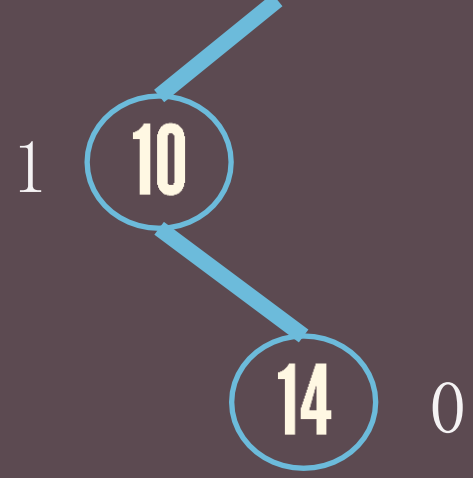


R – root

P – pivot

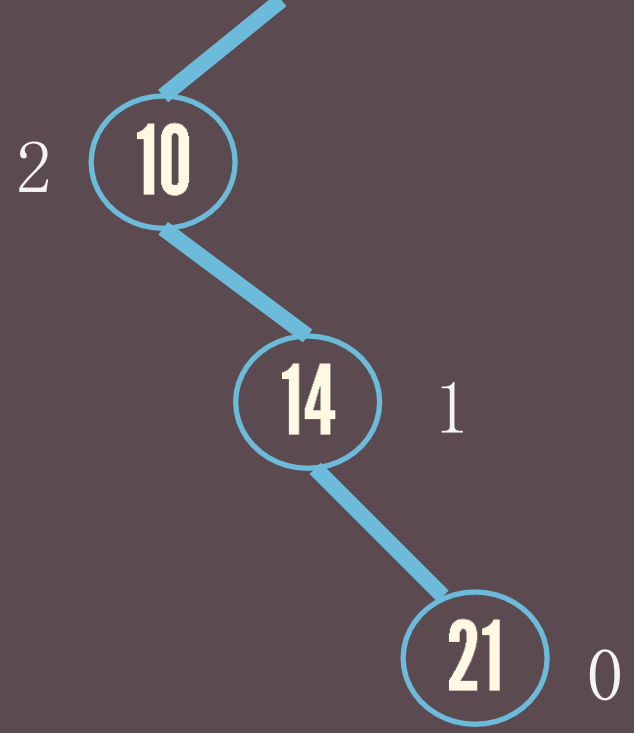
#2

```
fixUp(){

   start at the node inserted and travel
   up the tree:
      if an imbalance is found,
         if pivot is right leaning and
         root is right leaning
            do a left rotation on root.
      update height of the nodes.

}
```



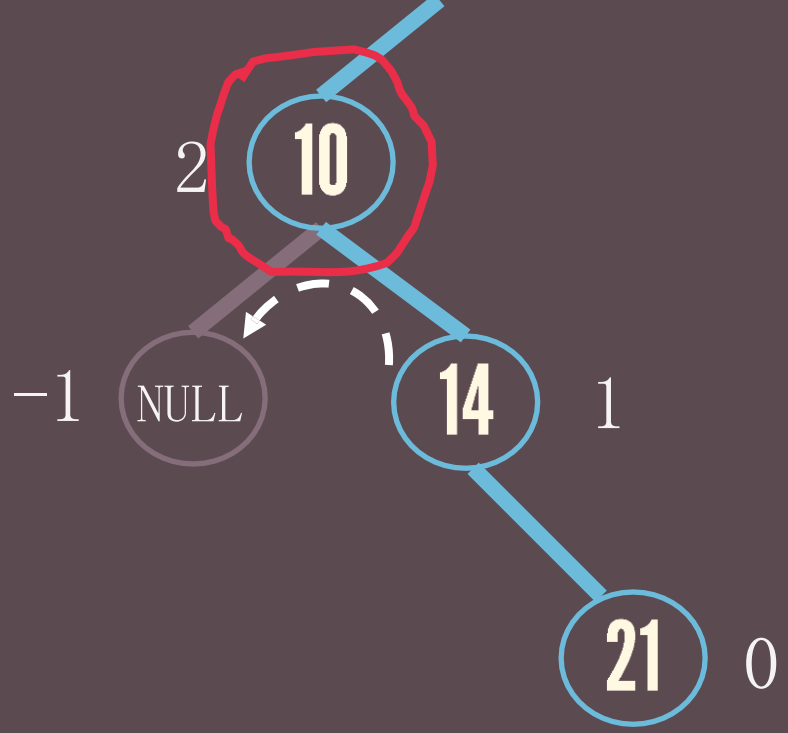
#2

insert(21)



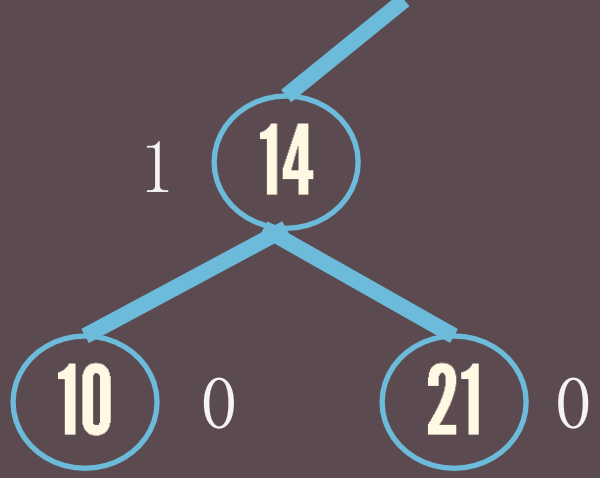
#2

insert(21)



#2

insert(21)



#2

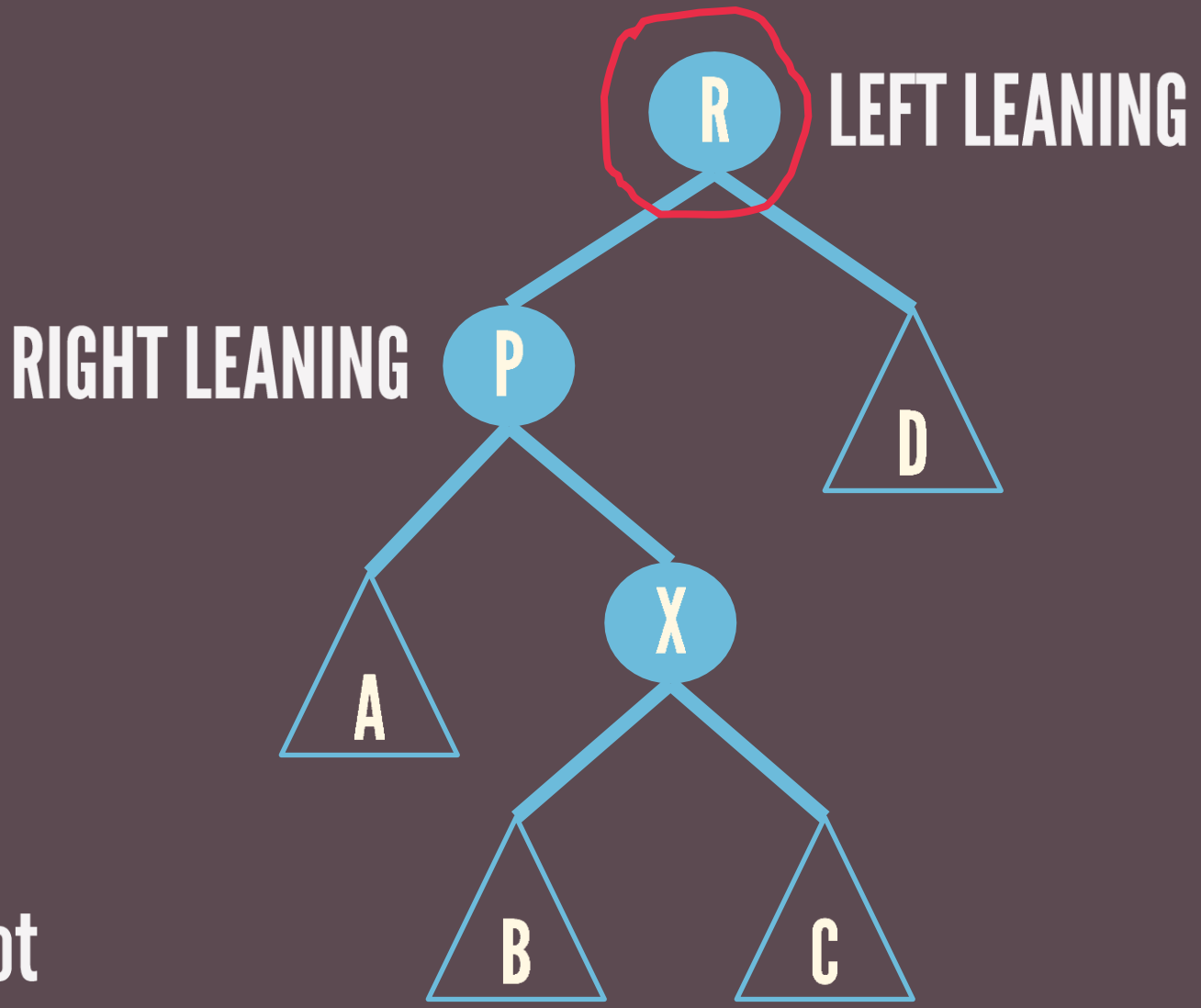
insert(21)

#3

LEFT RIGHT CASE

LEFT RIGHT ROTATE

#3

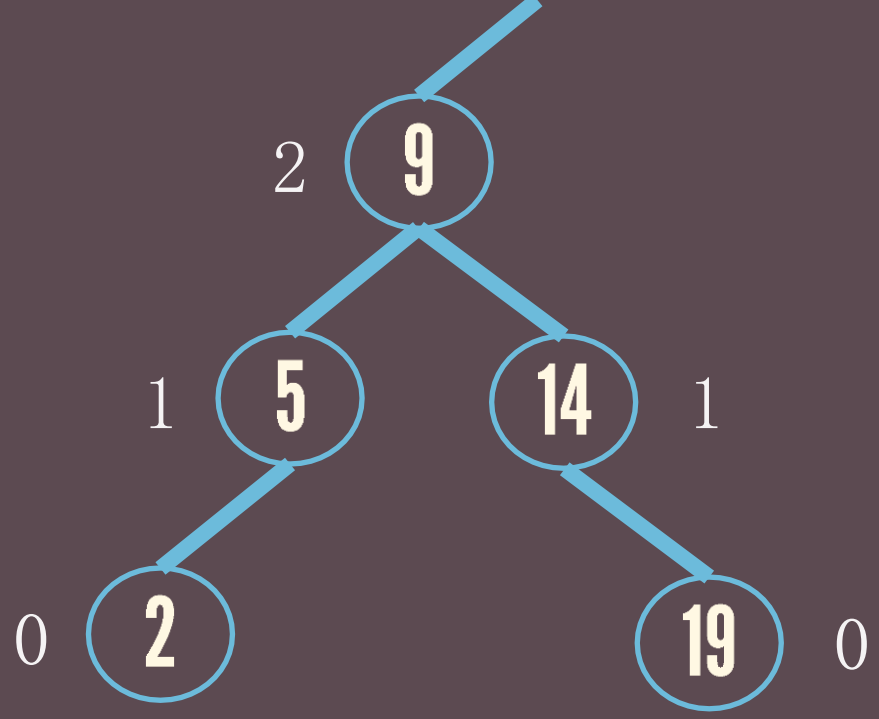


R – root

P – pivot

#3

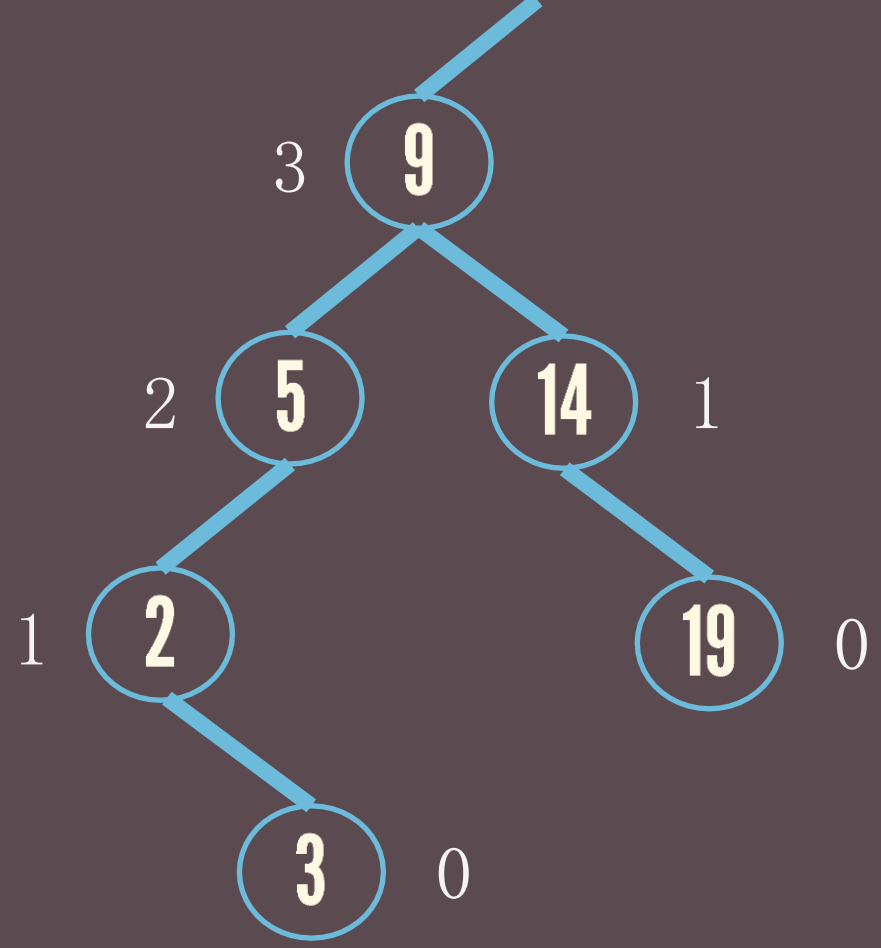
```
fixUp(){

   start at the node inserted and travel
   up the tree:
      if an imbalance is found,
         if pivot is right leaning and
         root is left leaning
            do a left rotation on pivot.
            do a right rotation on root.
      update height of the nodes.

}
```



#3

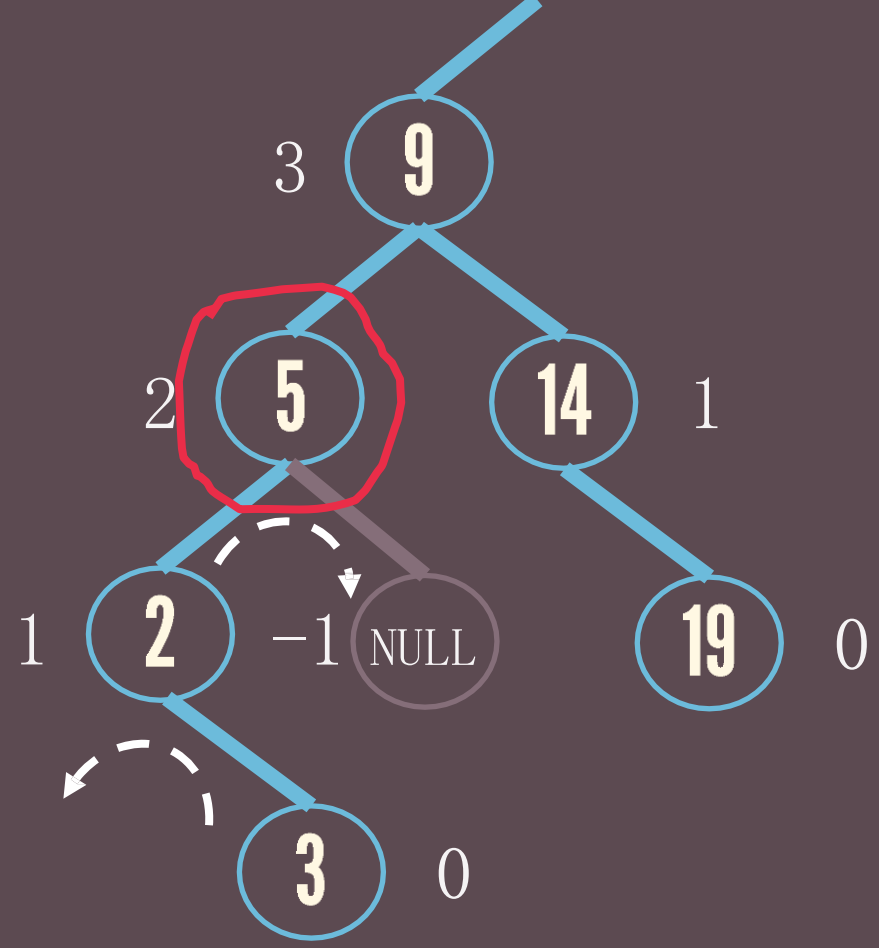
insert(3)

#3

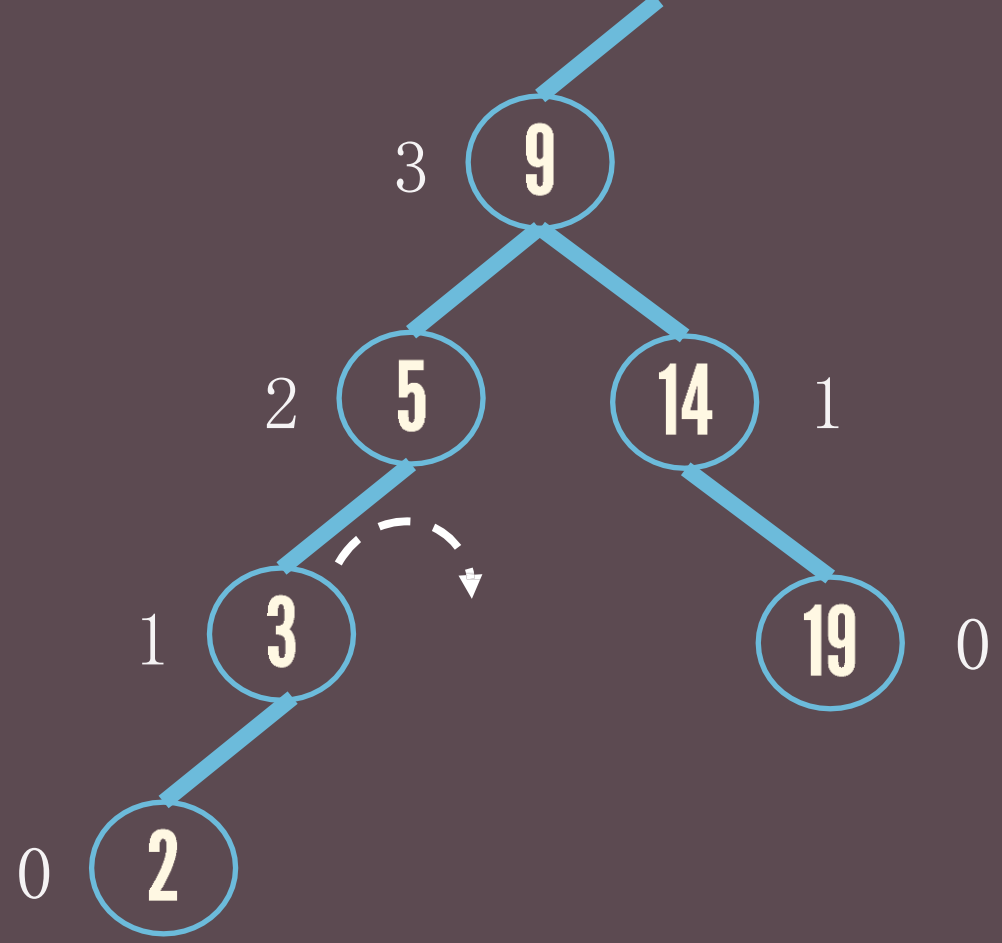


insert(3)

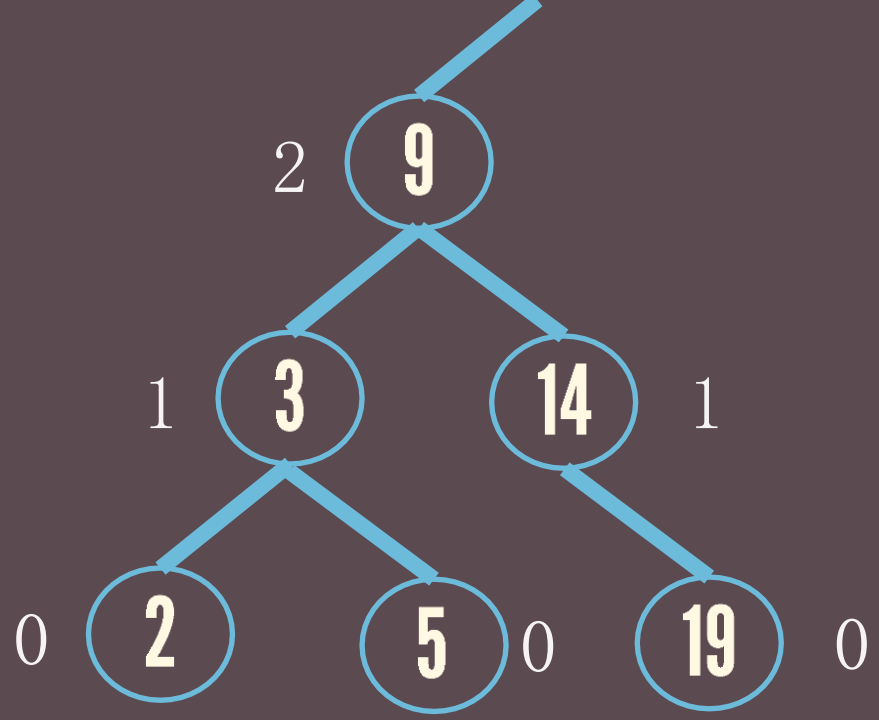
#3



insert(3)



insert(3)



#3

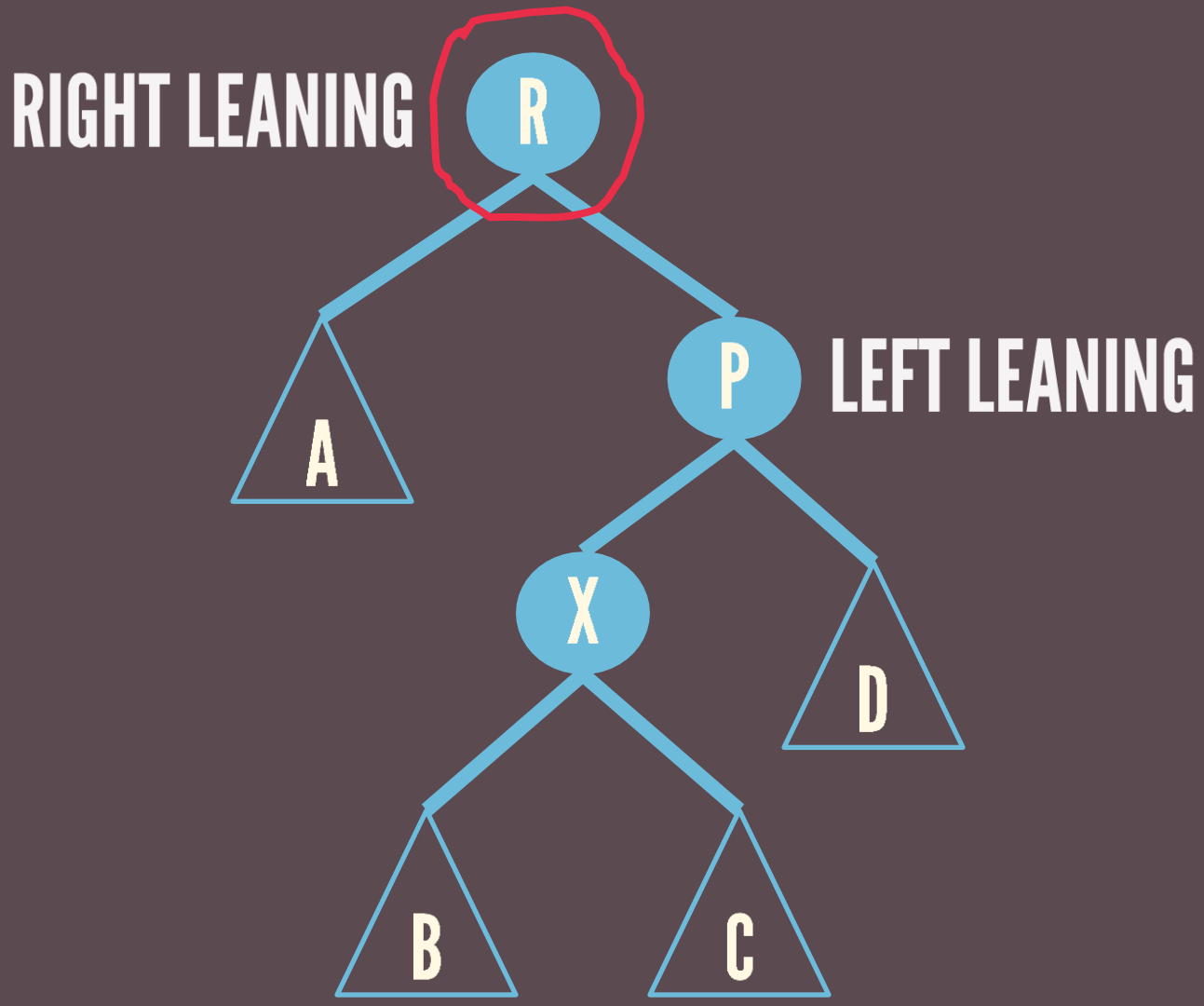
insert(3)

#4

RIGHT LEFT CASE

RIGHT LEFT ROTATE

#4



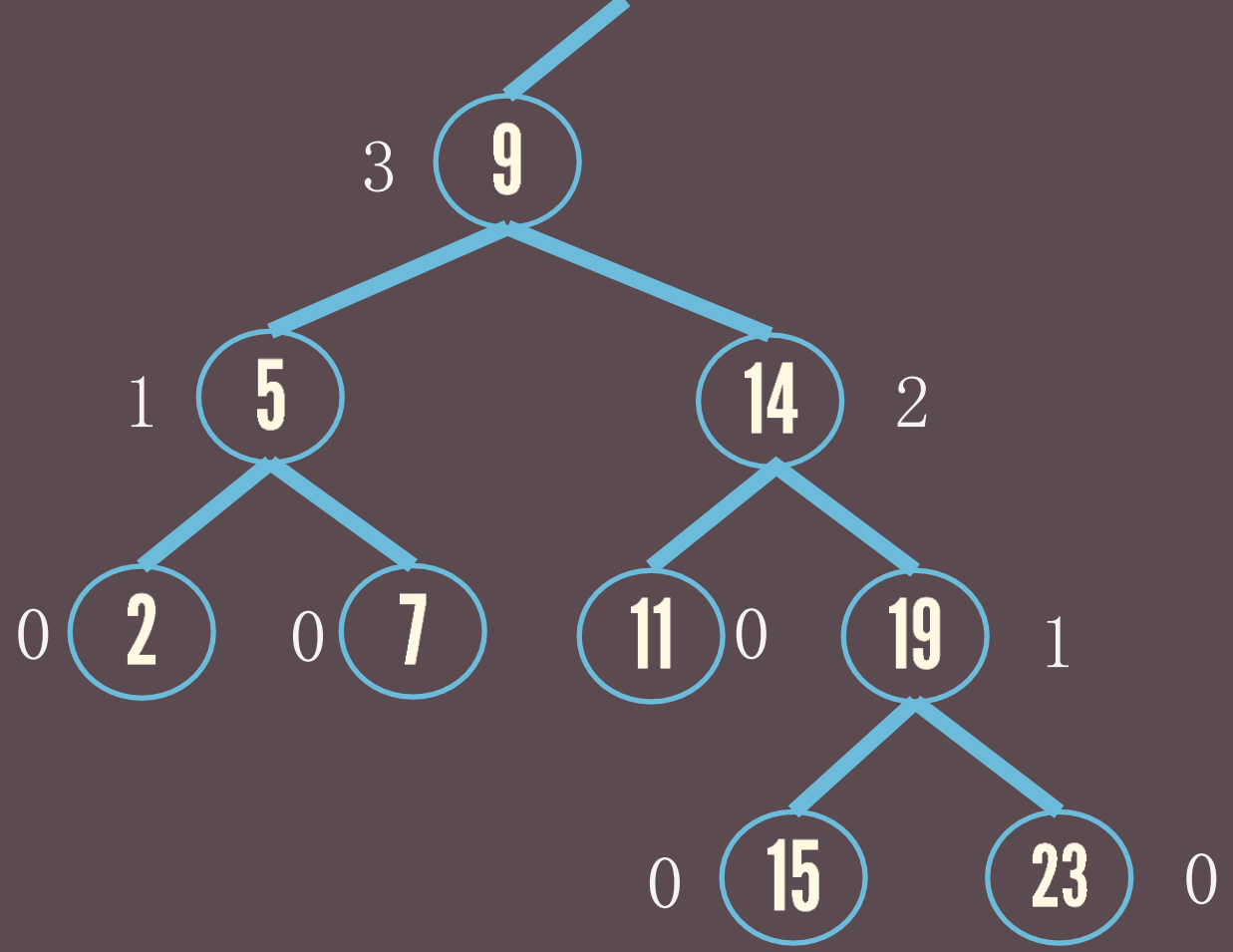
R – root

P – pivot

#4

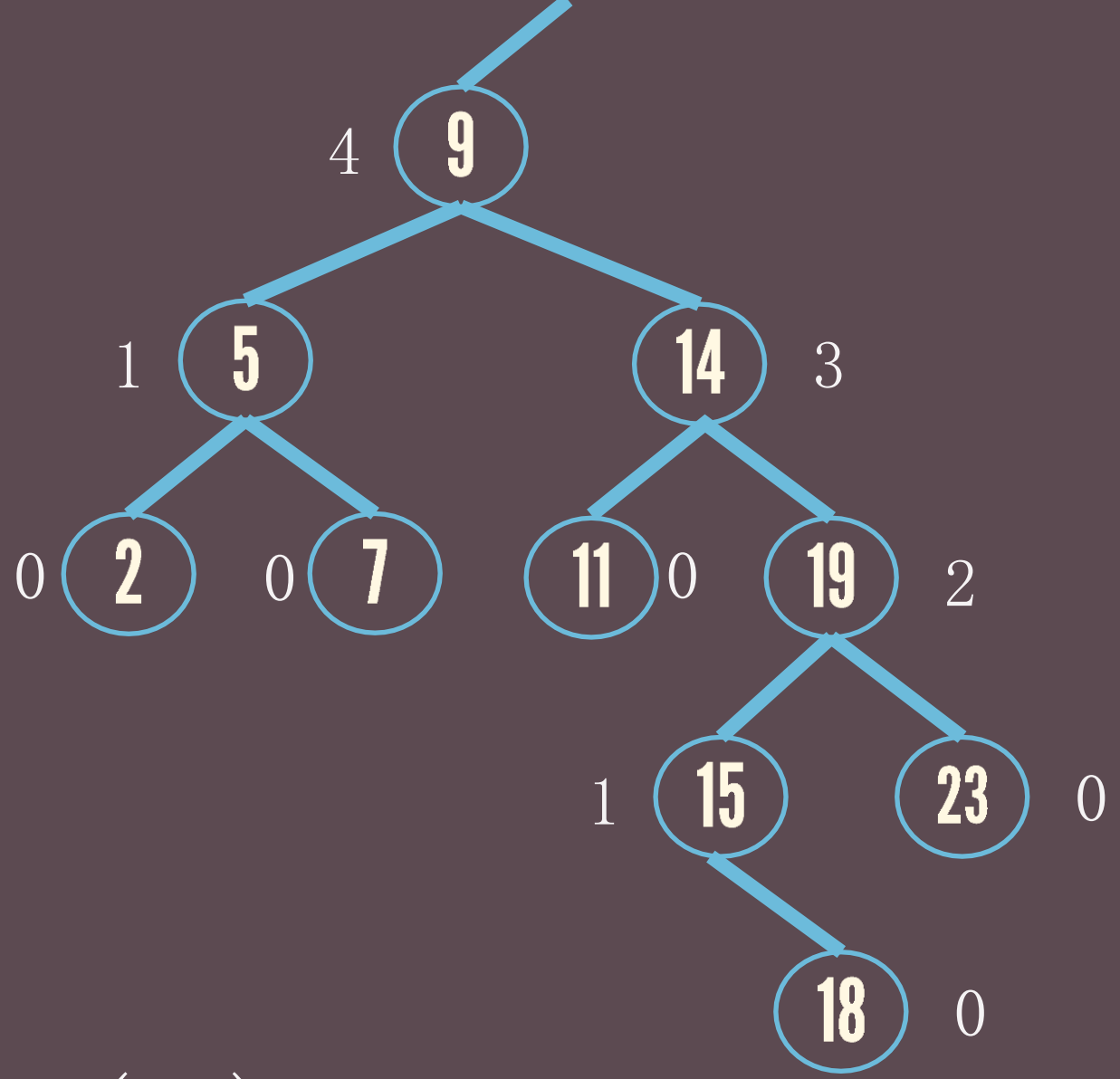
```
fixUp(){

   start at the node inserted and travel
   up the tree:
      if an imbalance is found,
         if pivot is left leaning and
         root is right leaning
            do a right rotation on pivot.
            do a left rotation on root.
      update height of the nodes.

}
```

#4



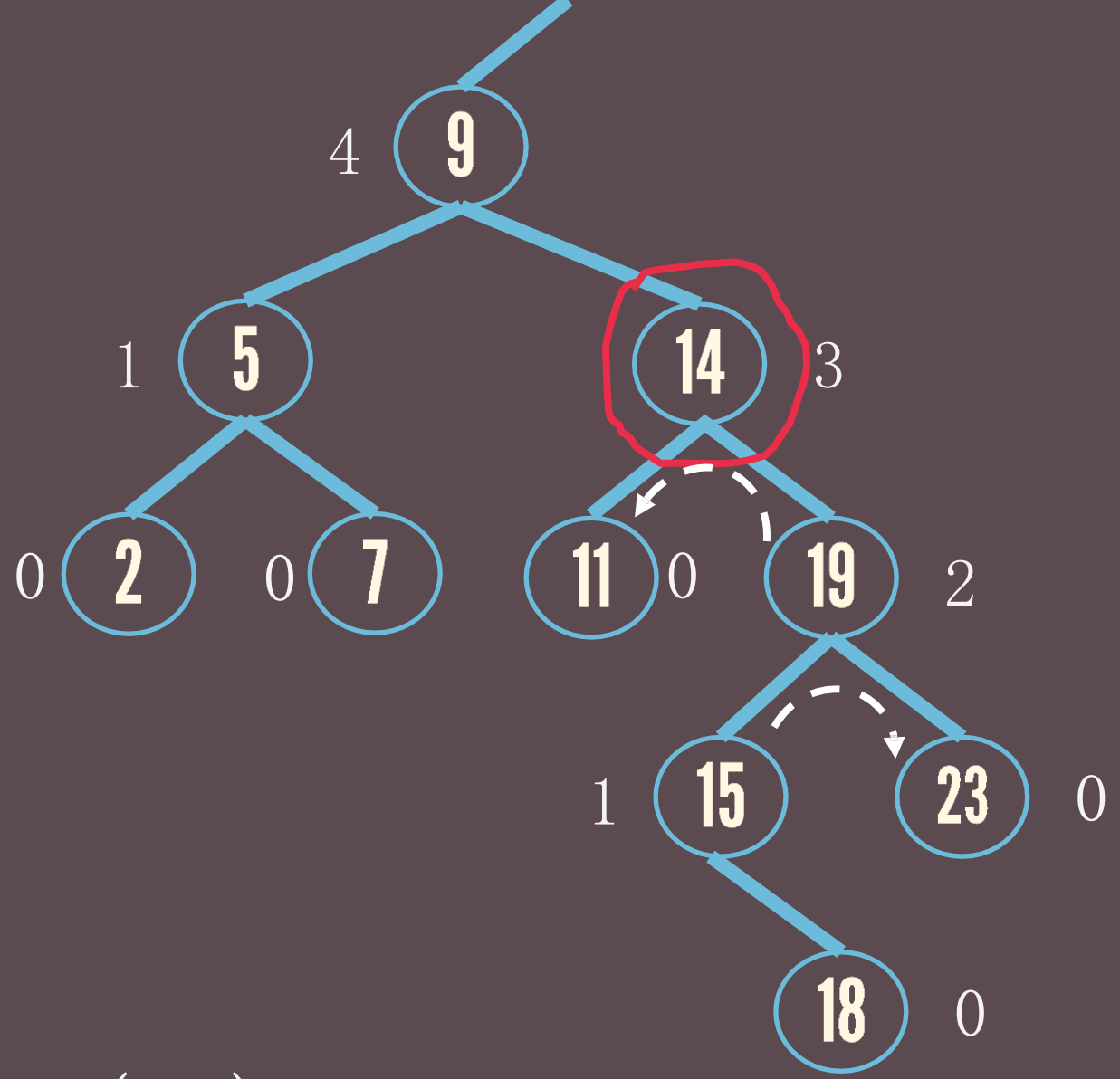
insert(18)

#4



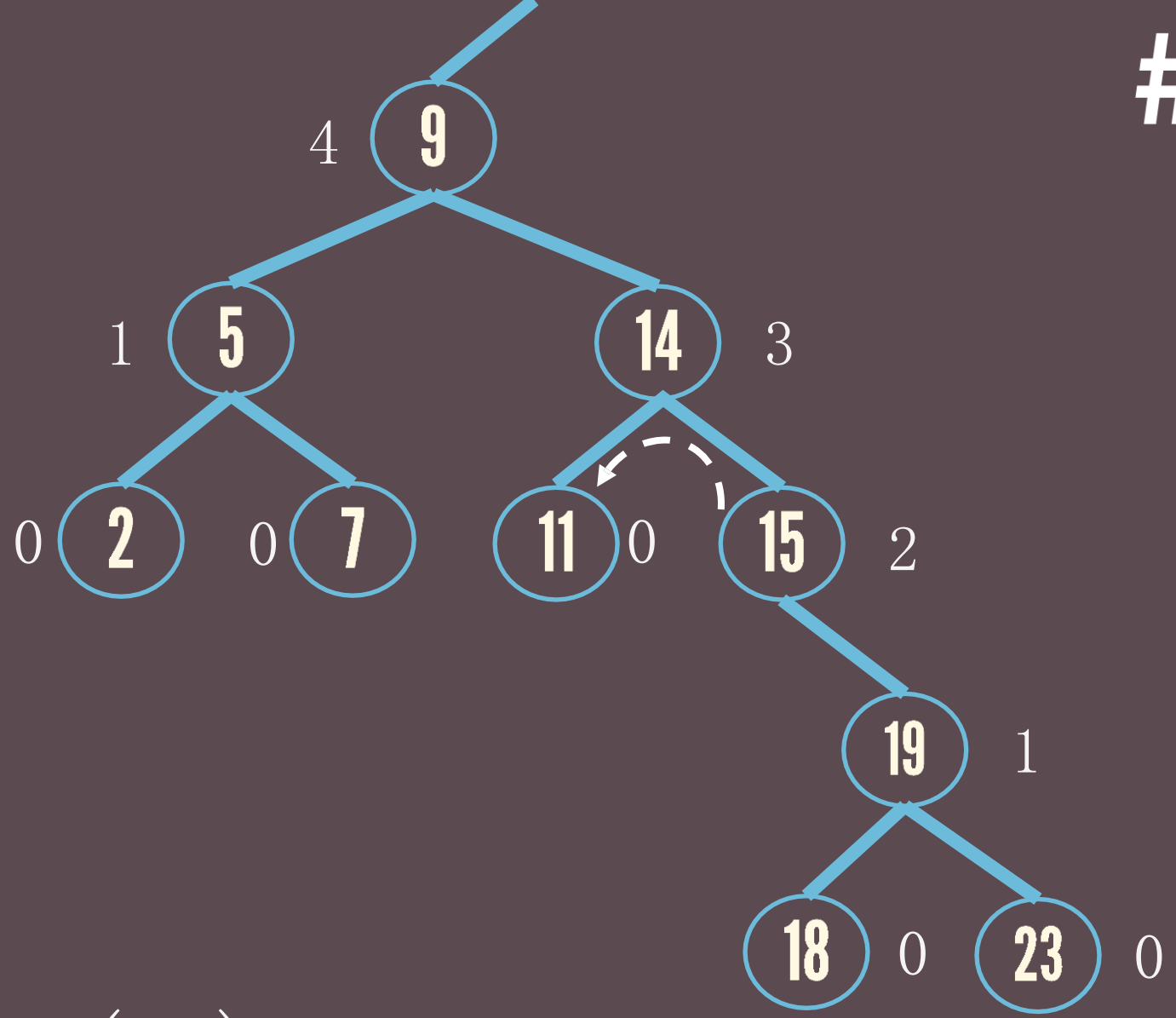
insert(18)

#4



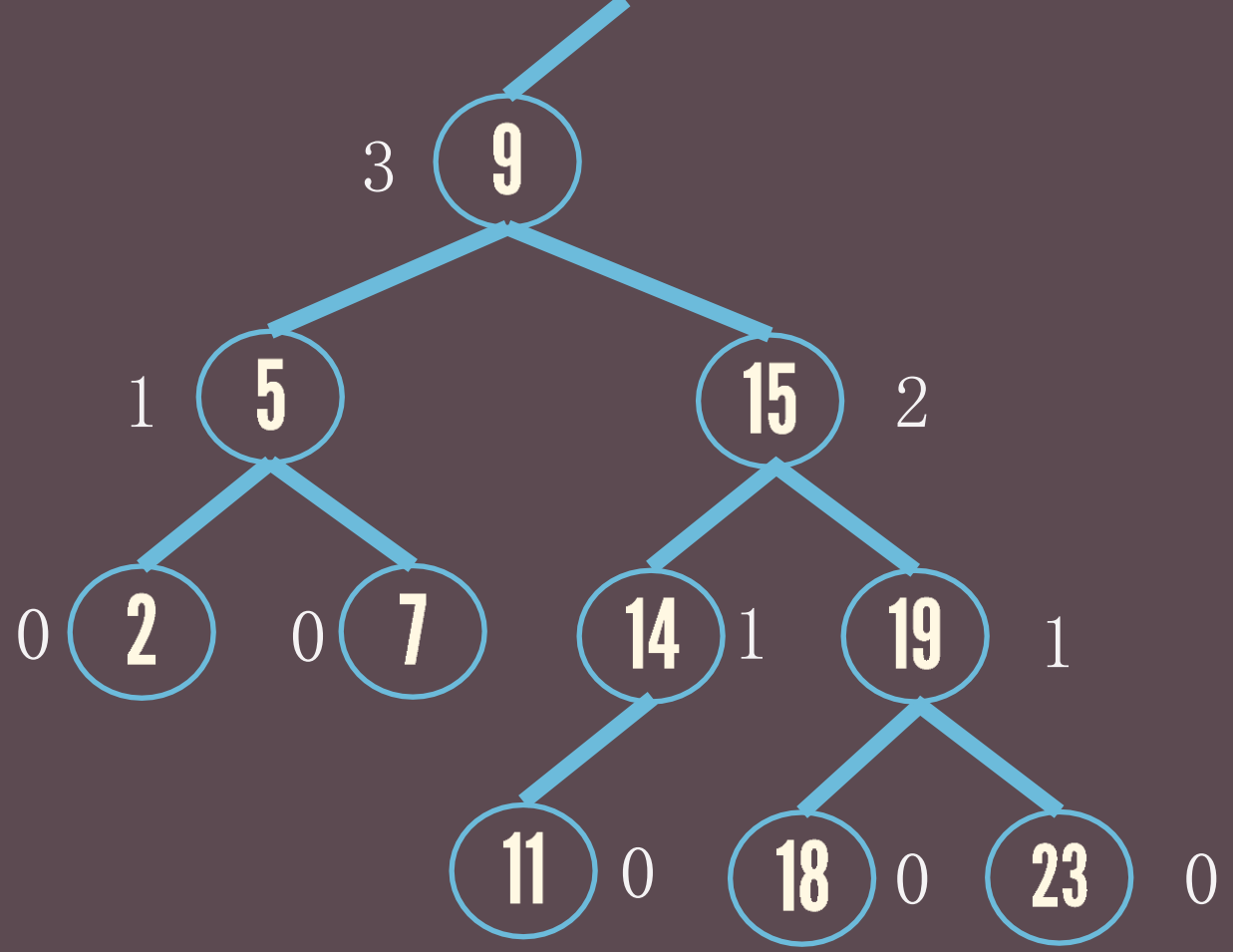
insert(18)

#4



insert(18)

#4



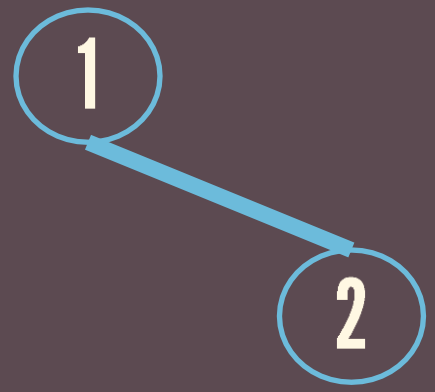
insert(18)

```
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)
```

insert(1)

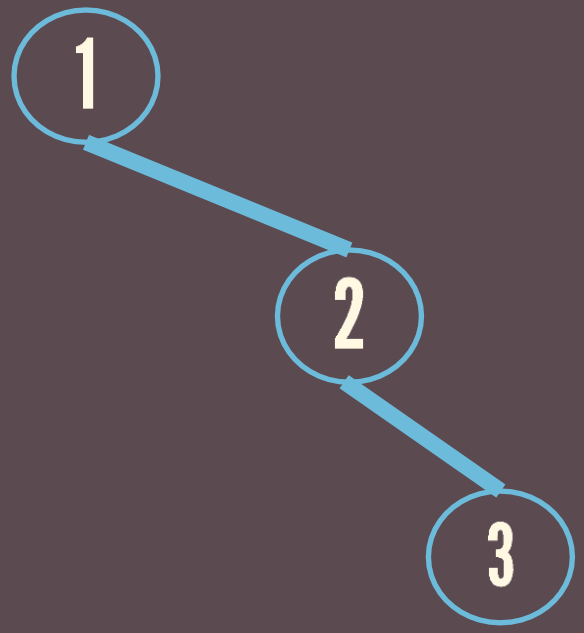
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(2)



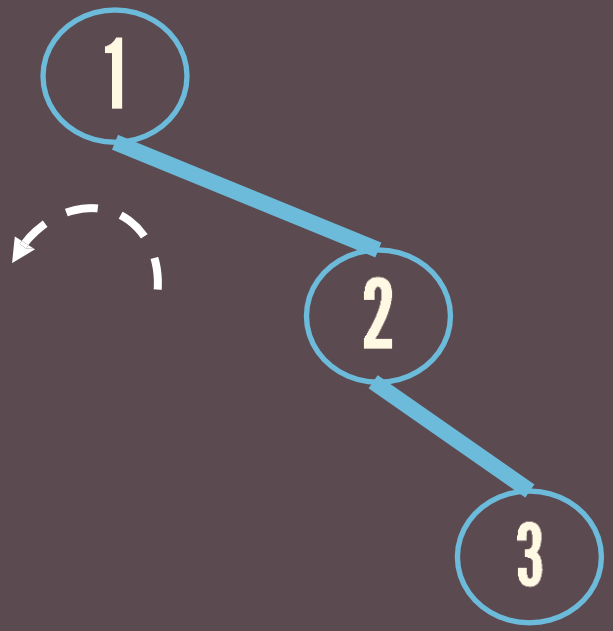
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(3)



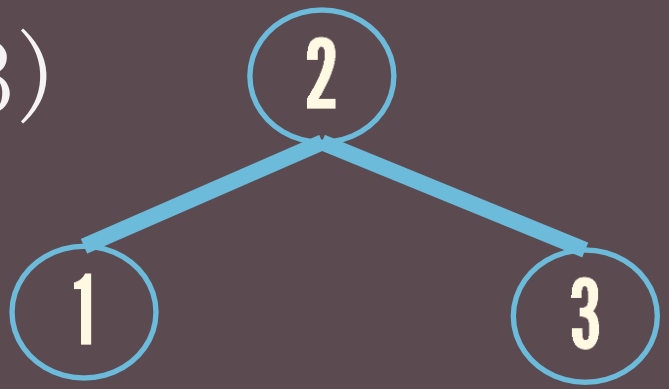
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(3)



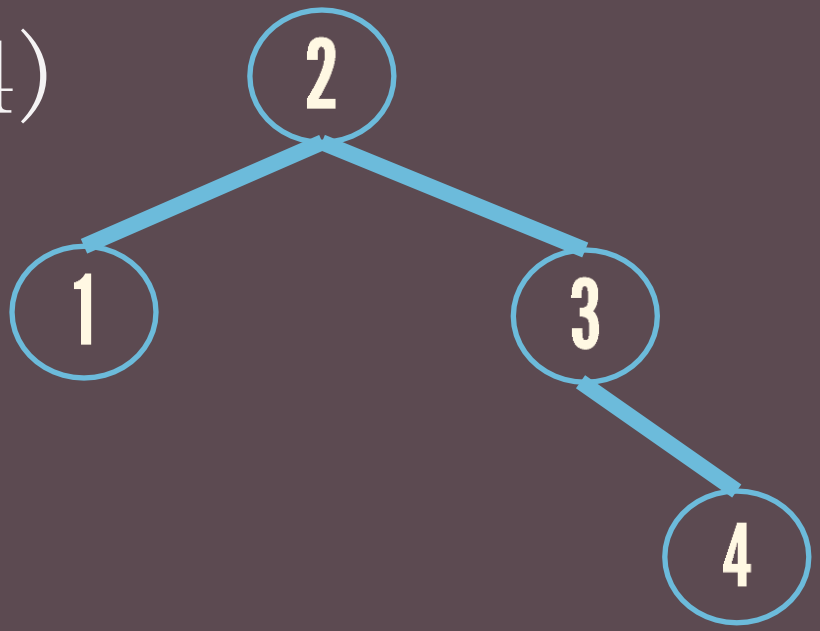
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(3)



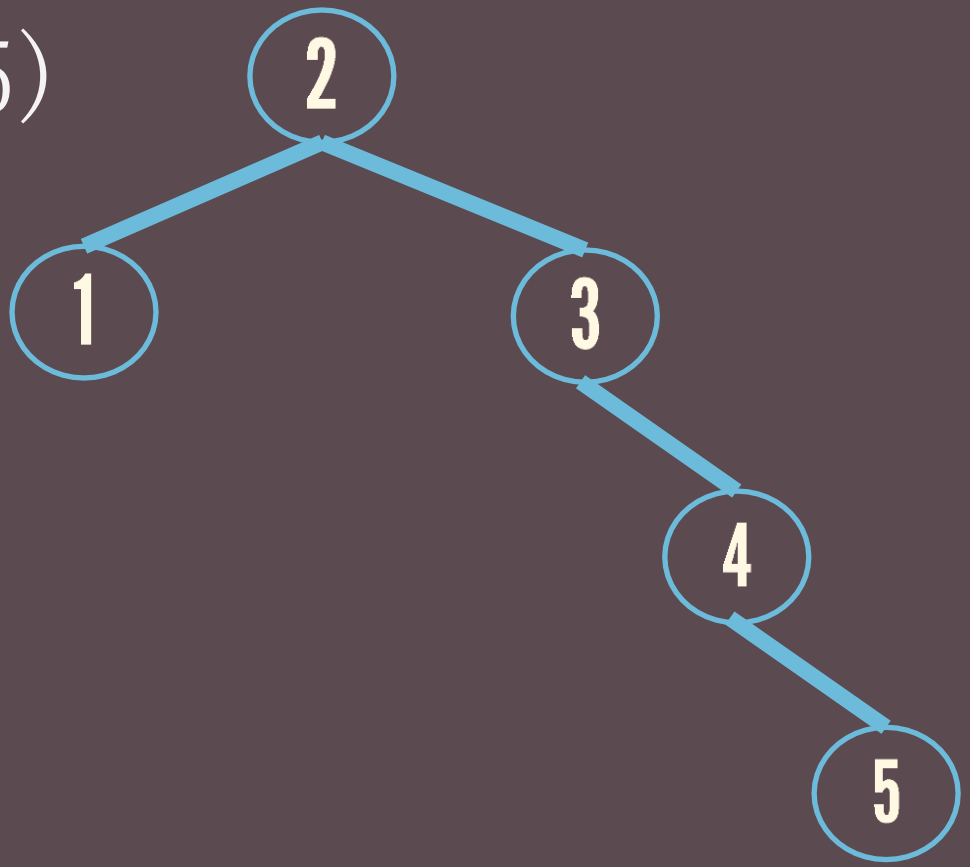
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(4)



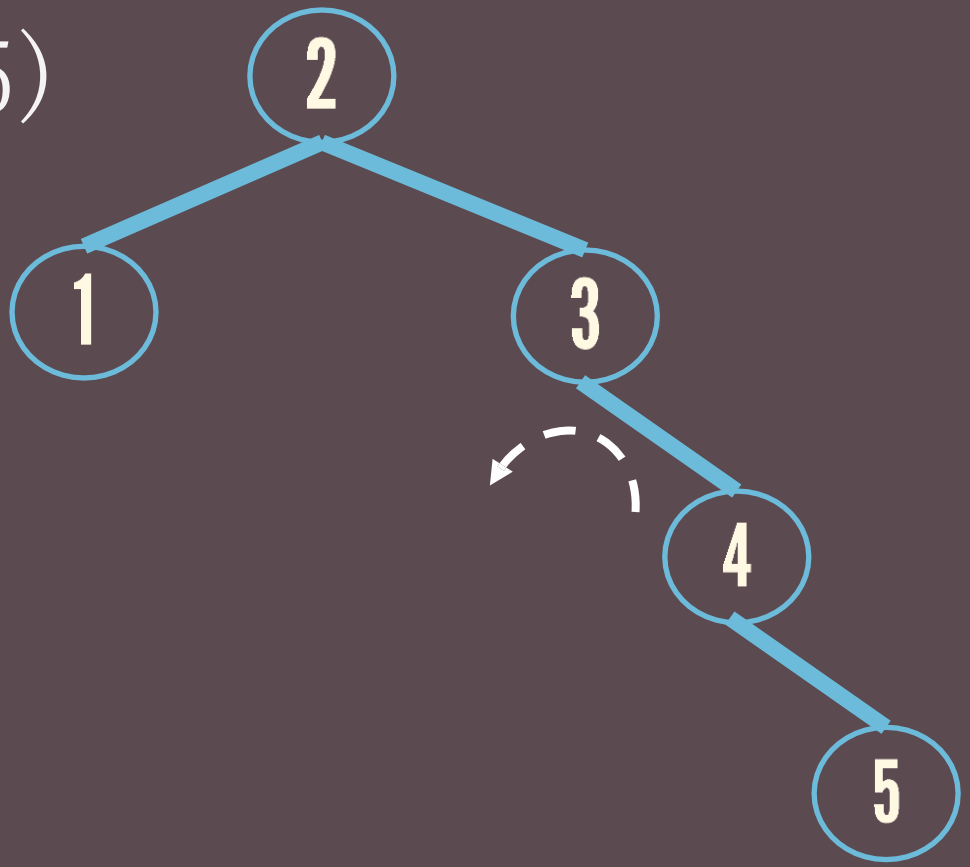
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(5)



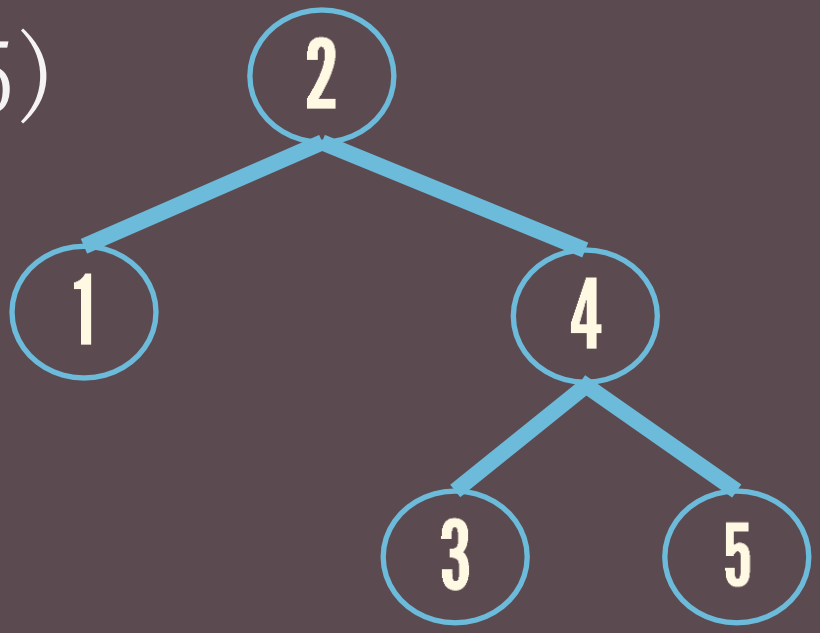
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(5)



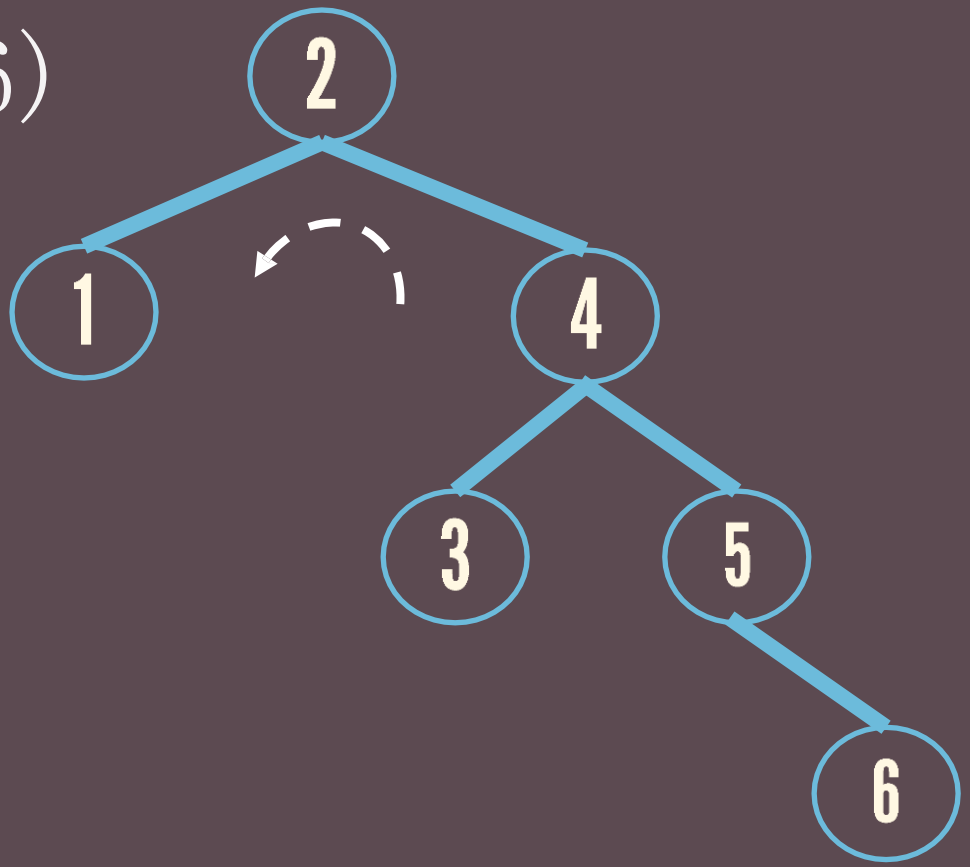
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(5)



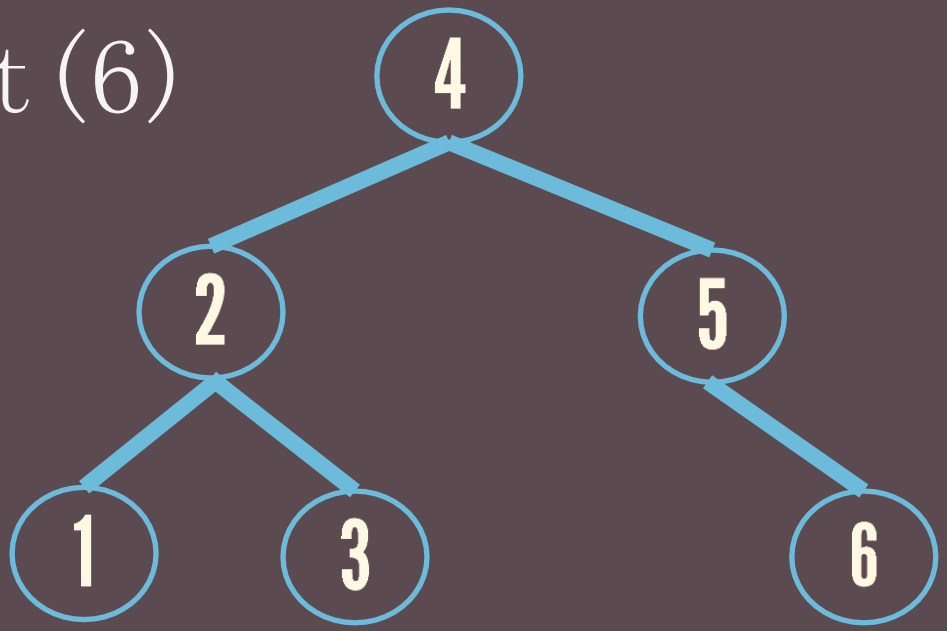
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(6)



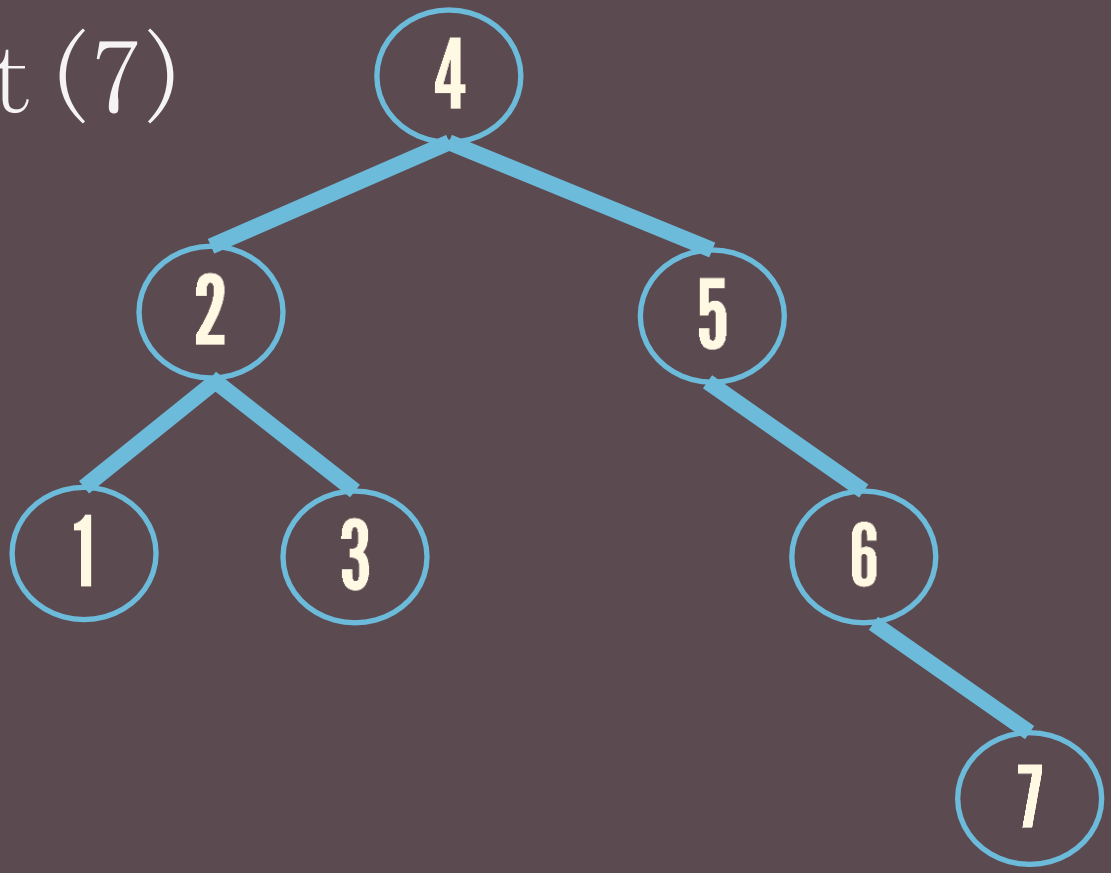
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(6)



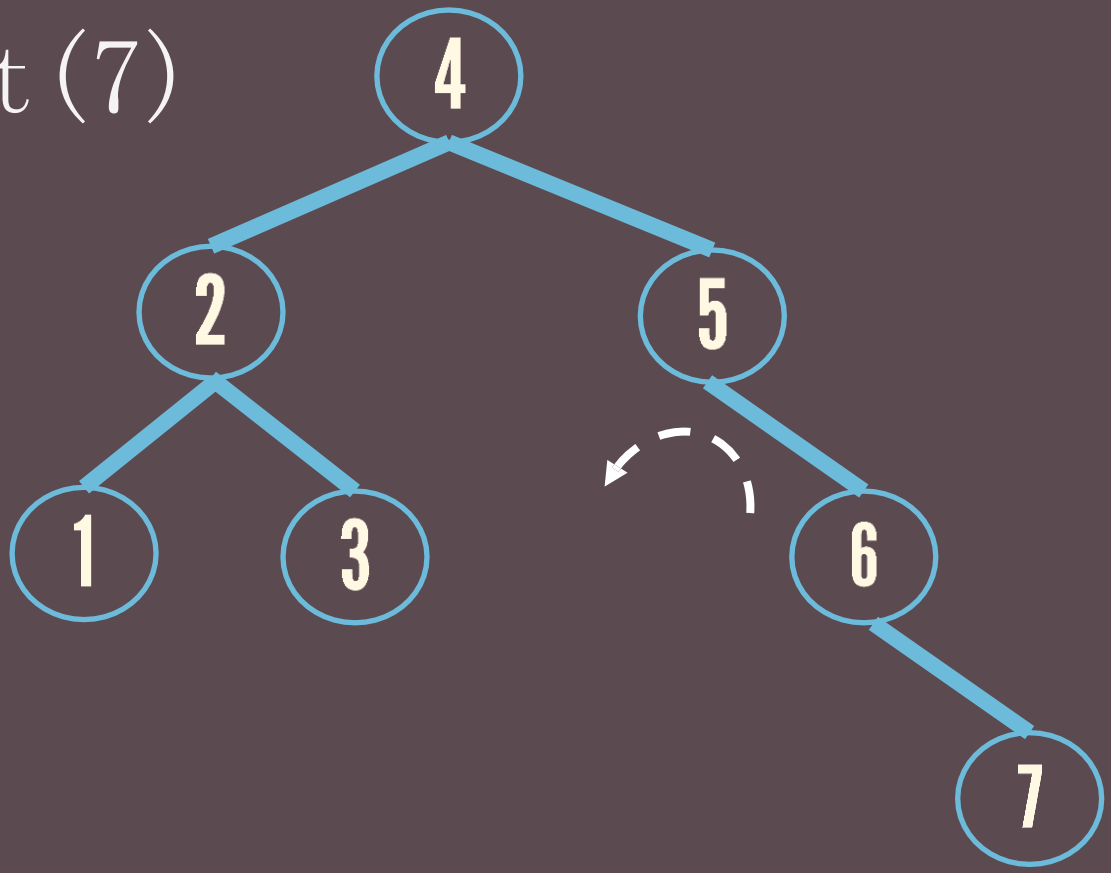
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(7)



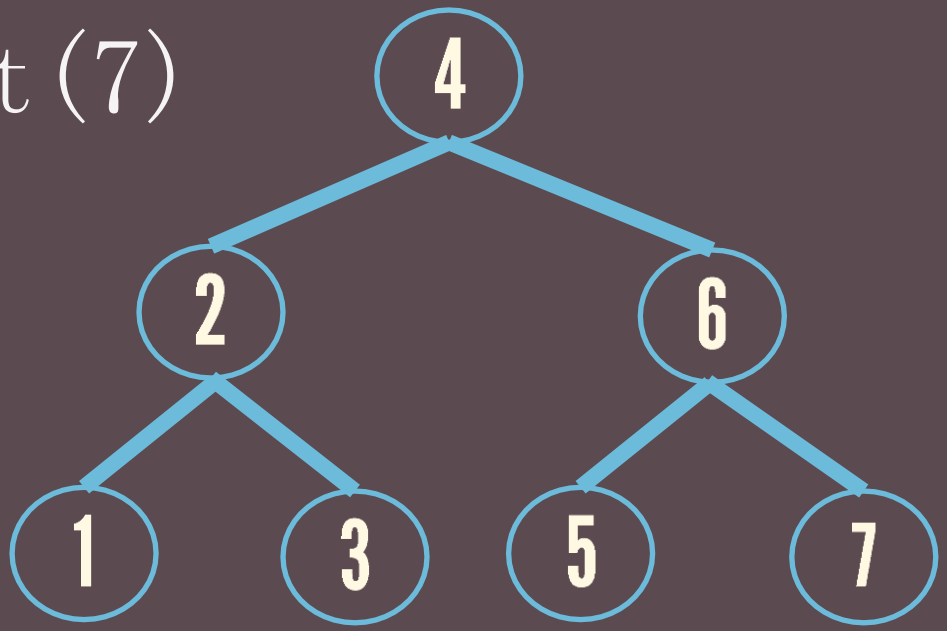
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(7)



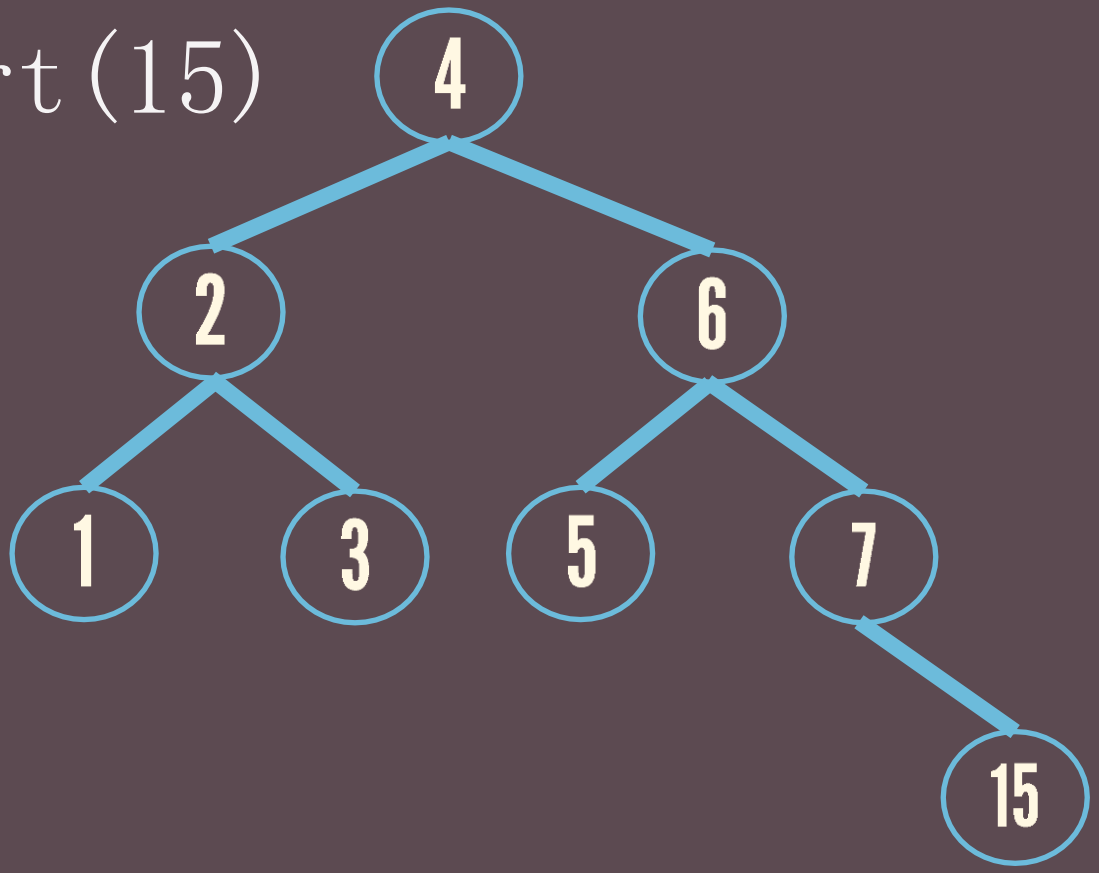
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(7)



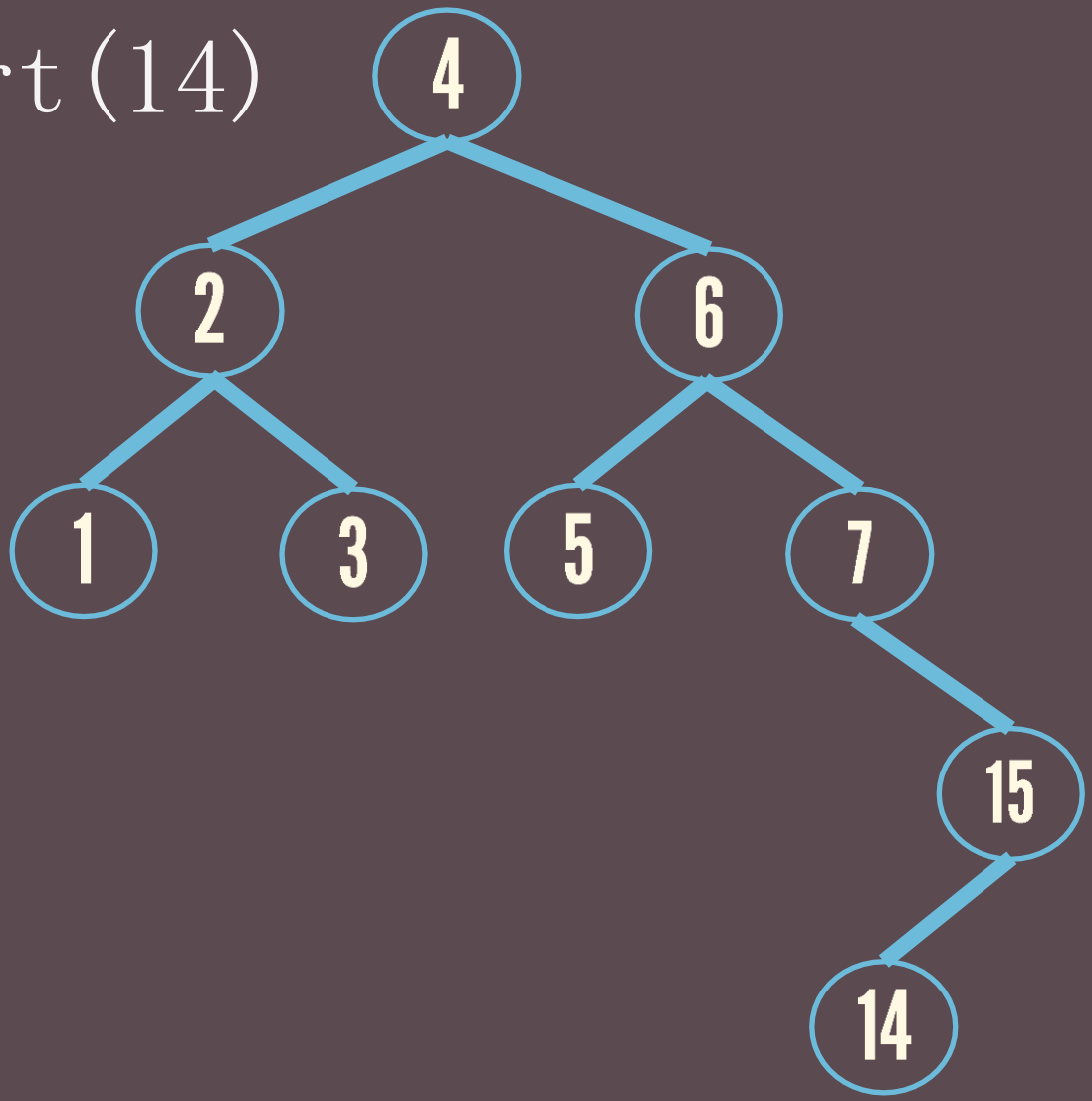
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(15)



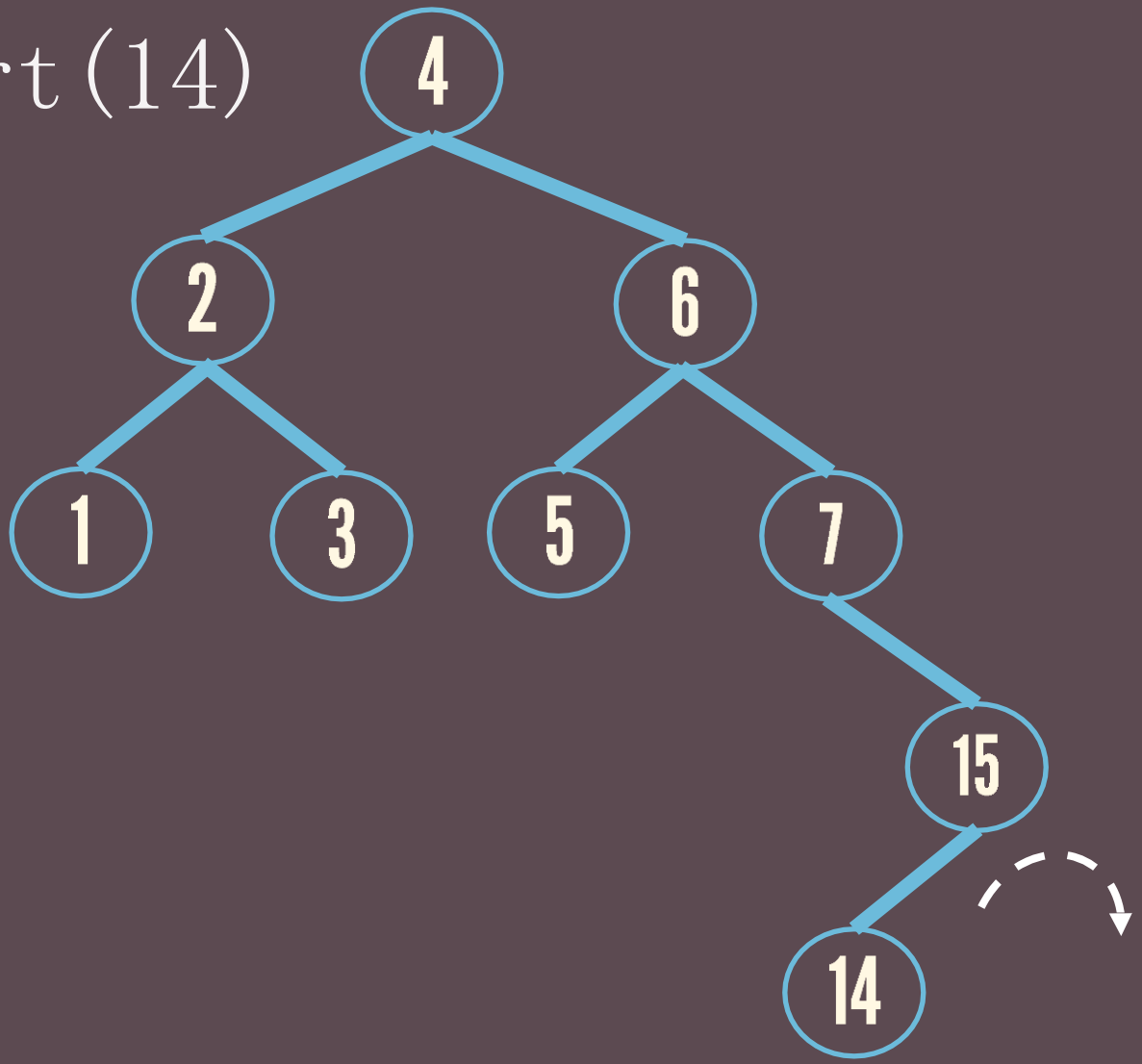
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(14)



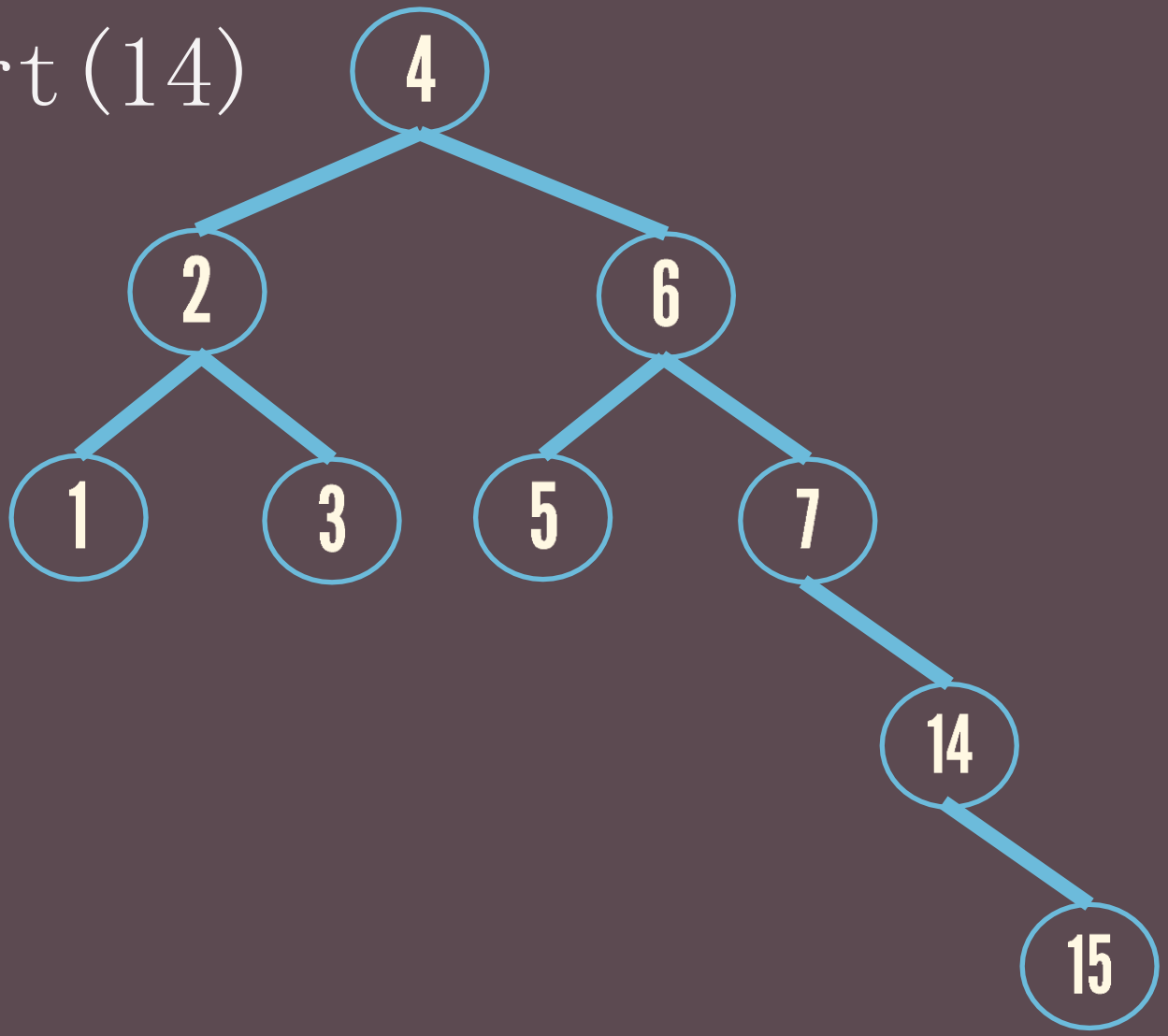
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(14)



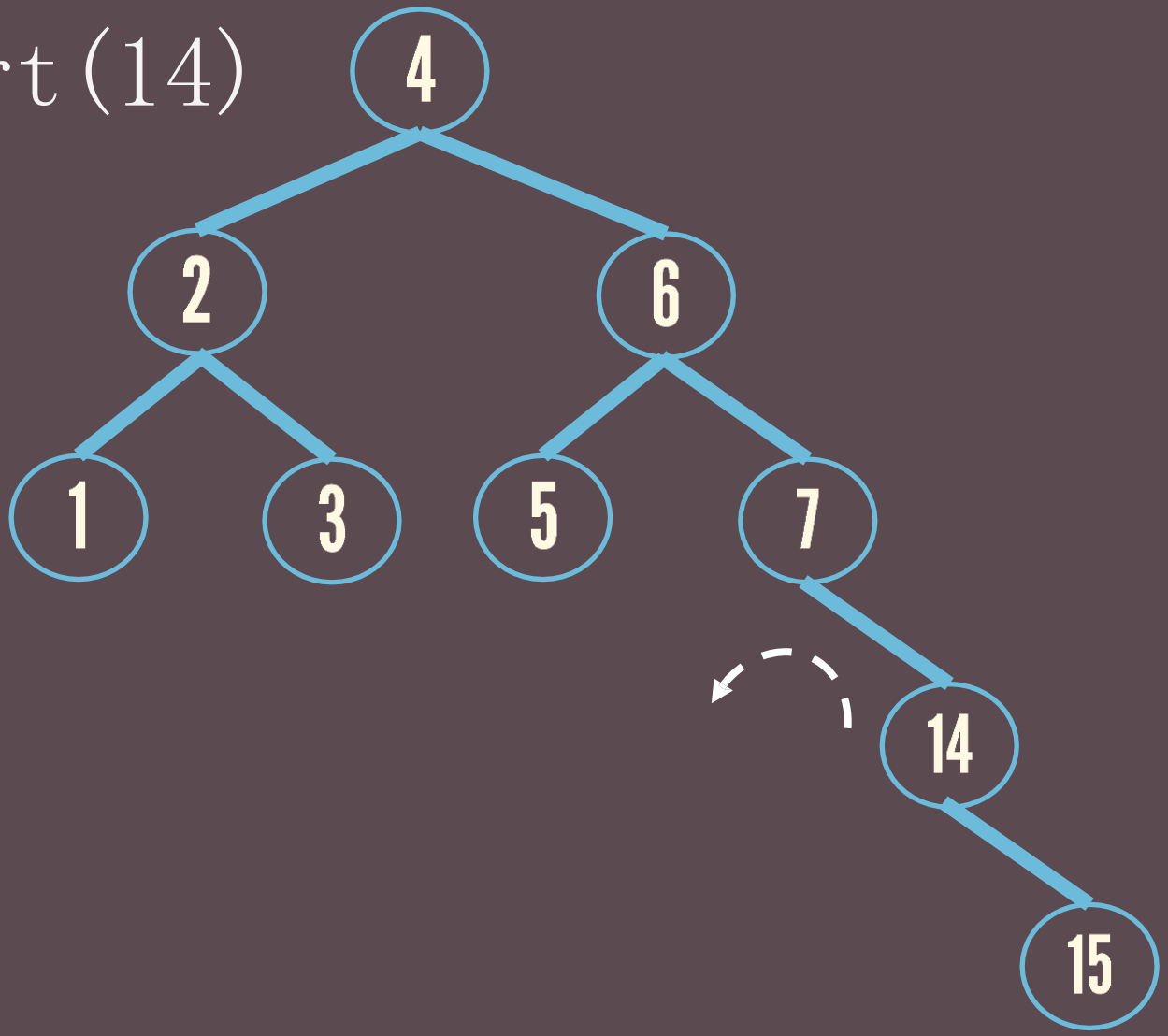
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(14)



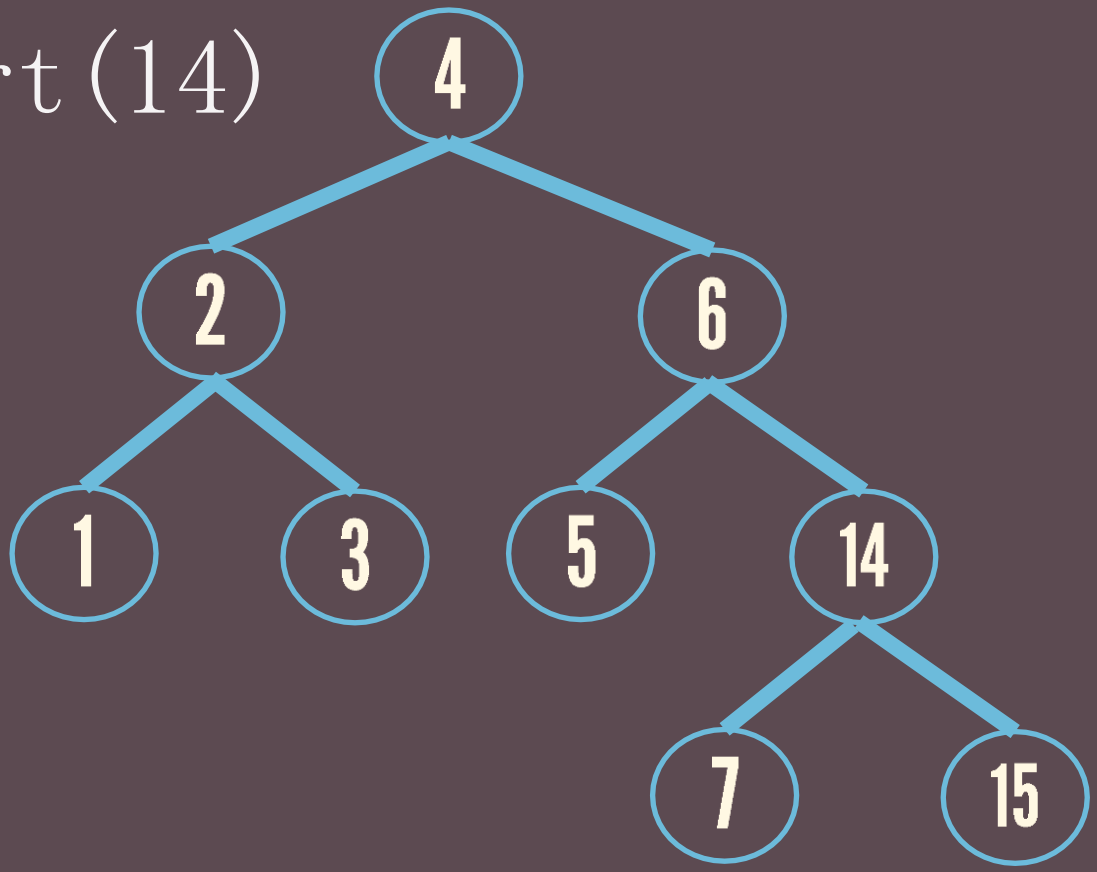
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(14)



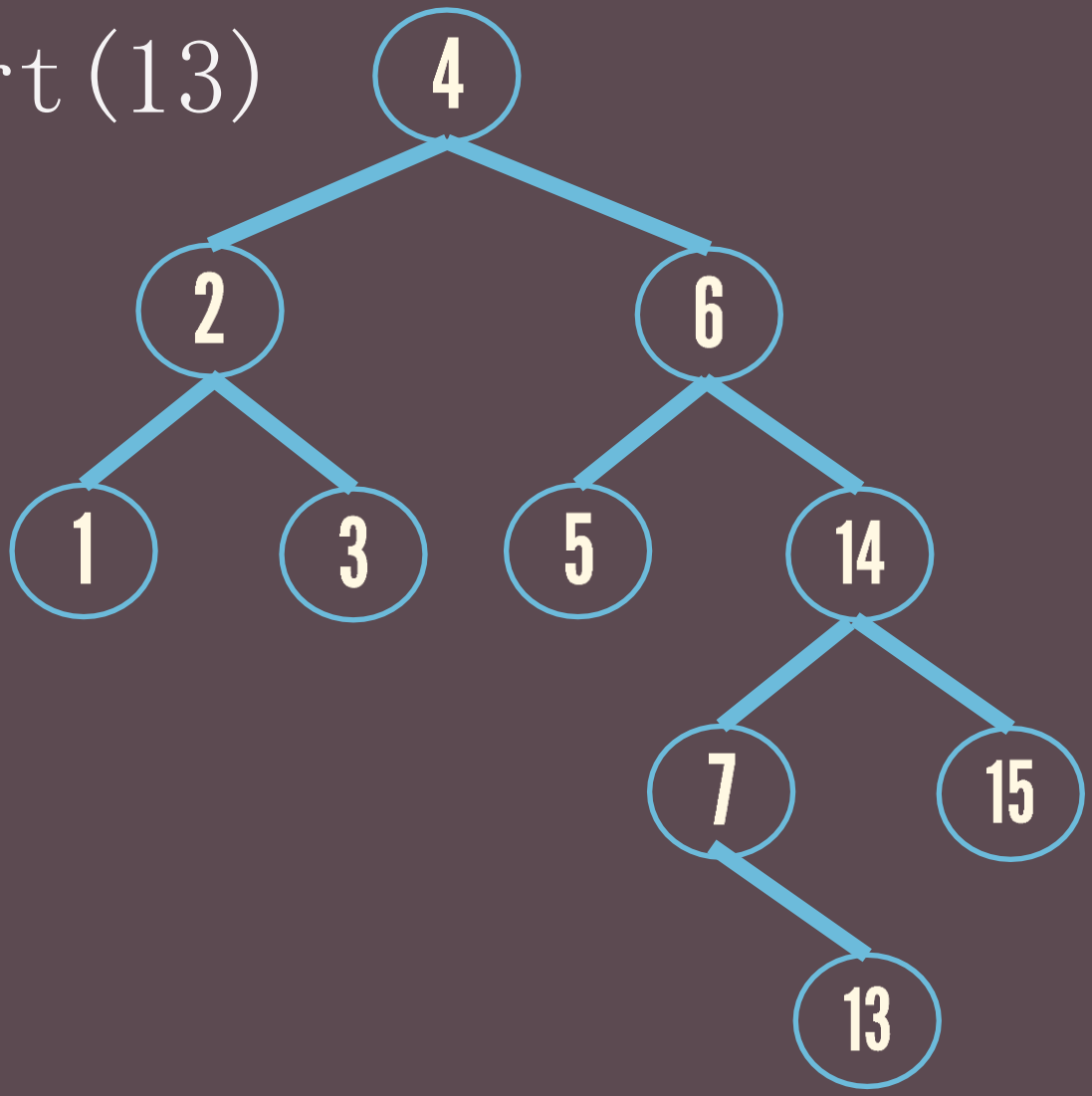
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(14)



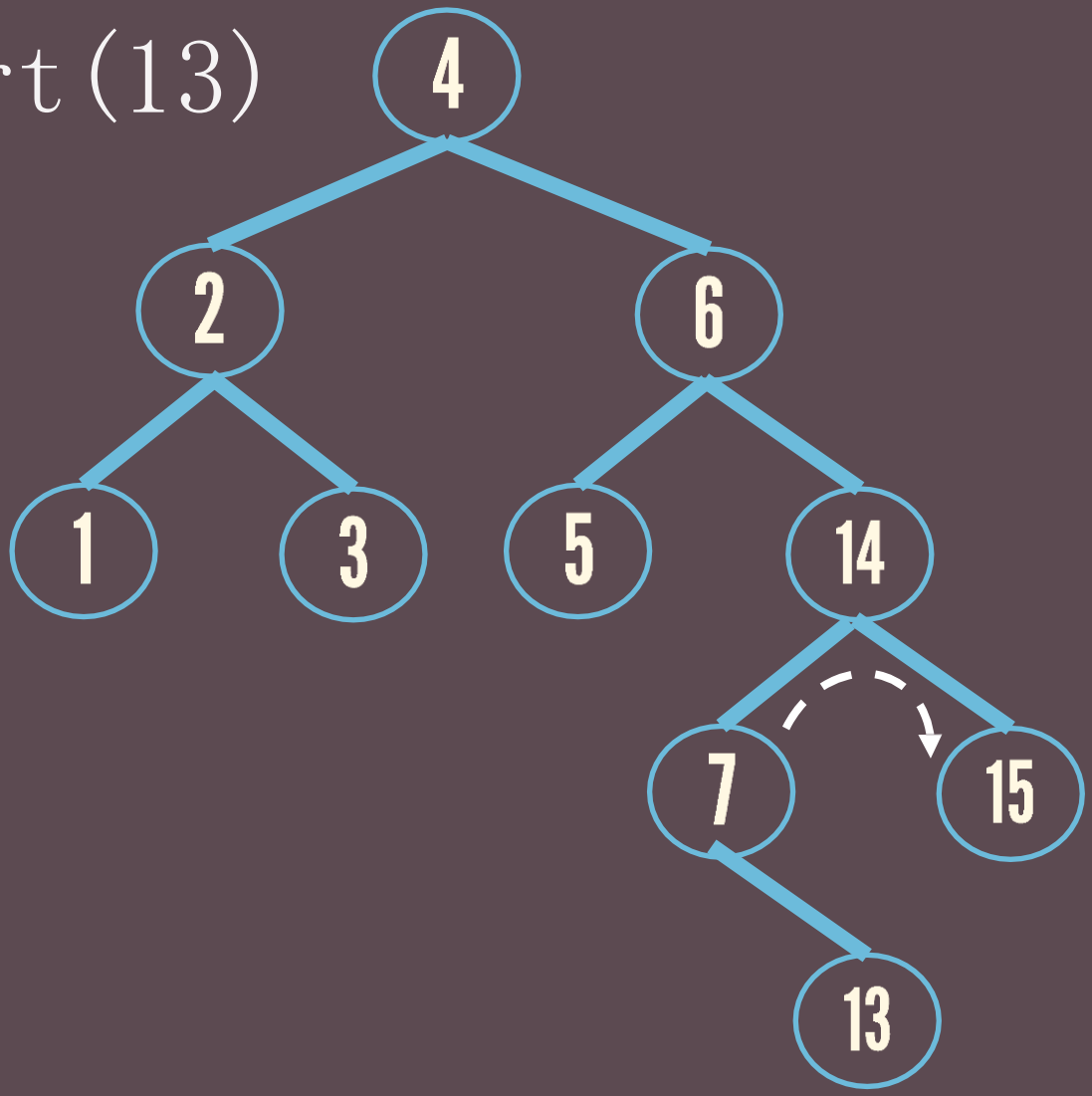
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(13)



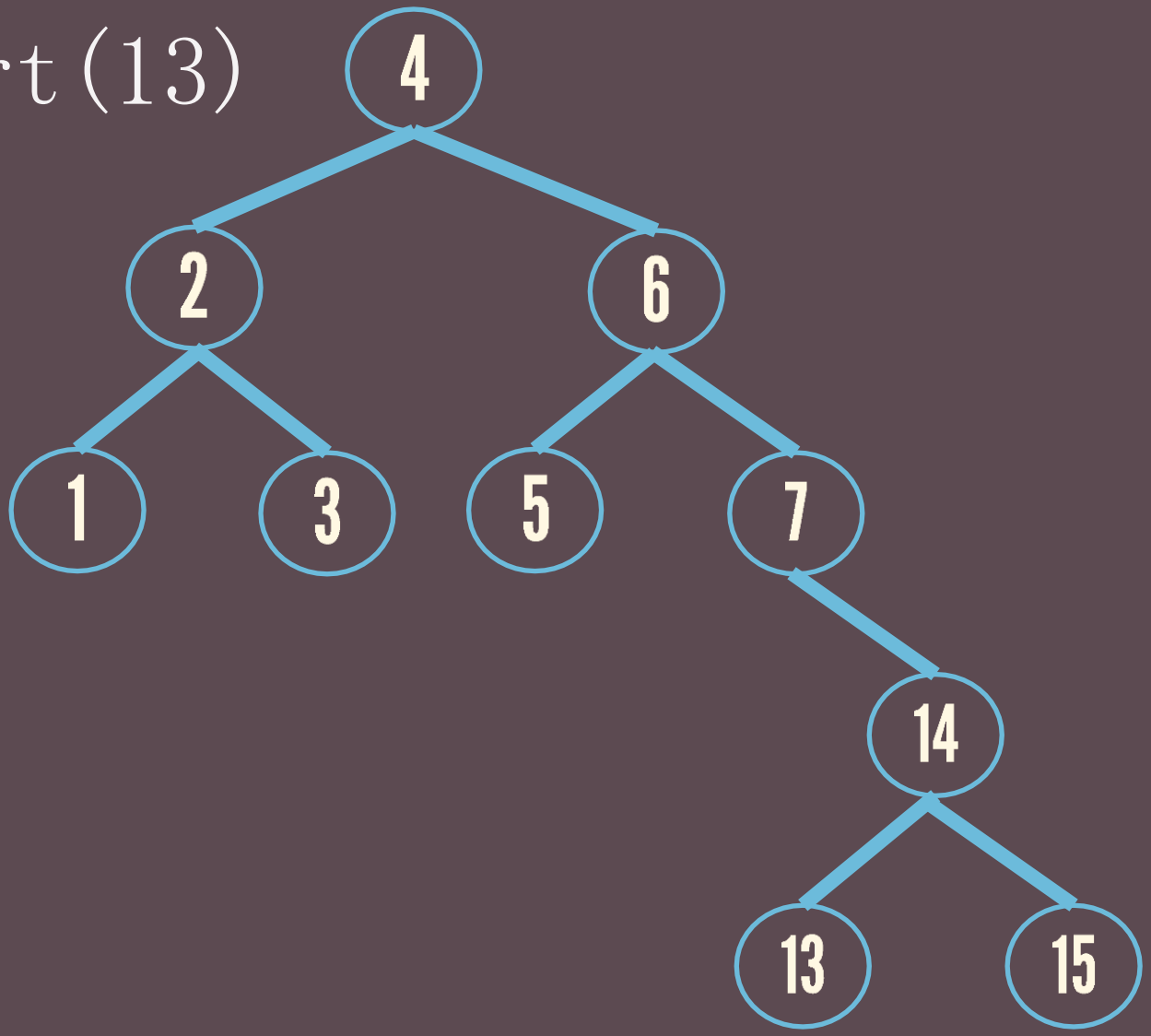
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(13)



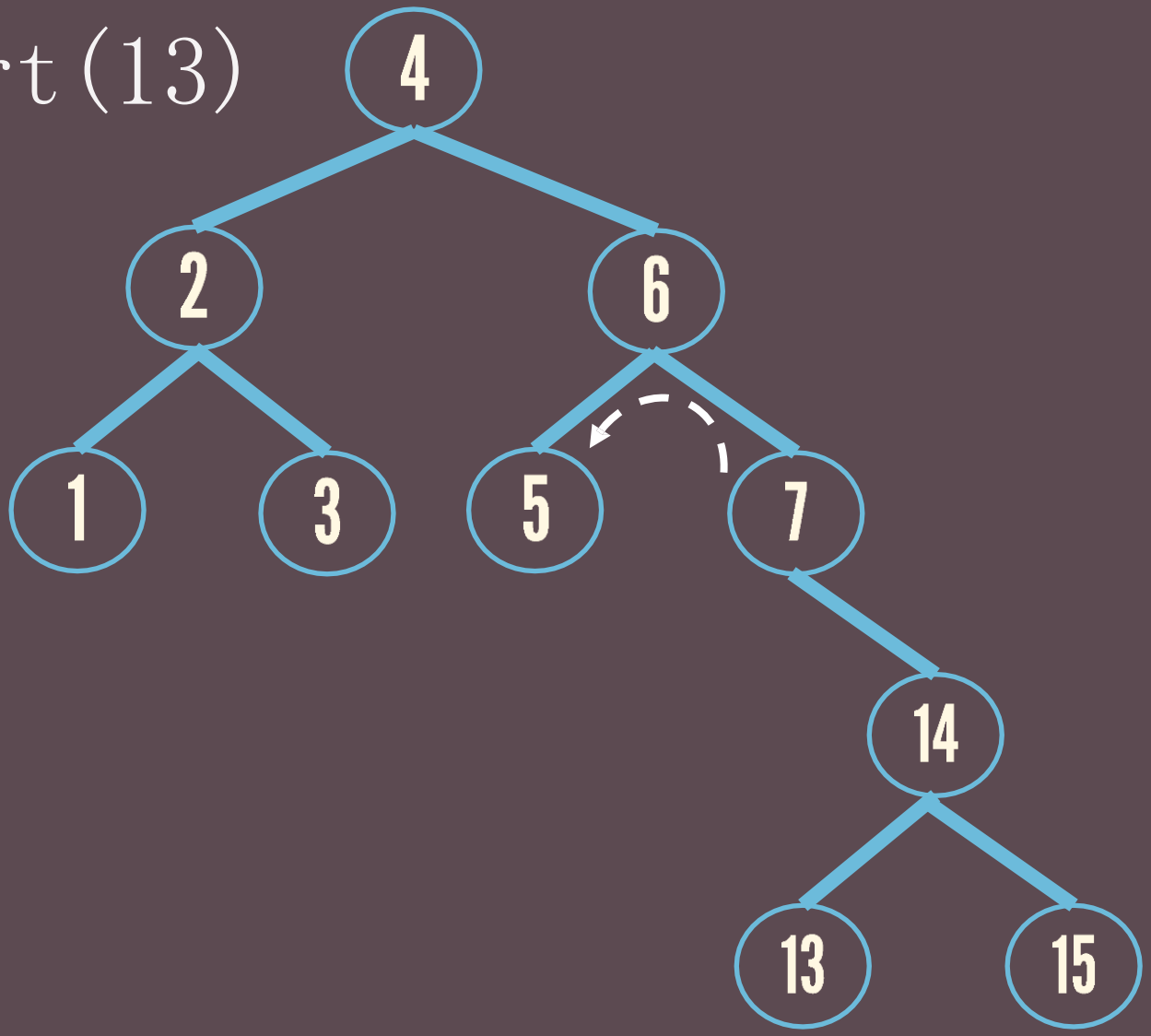
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(13)



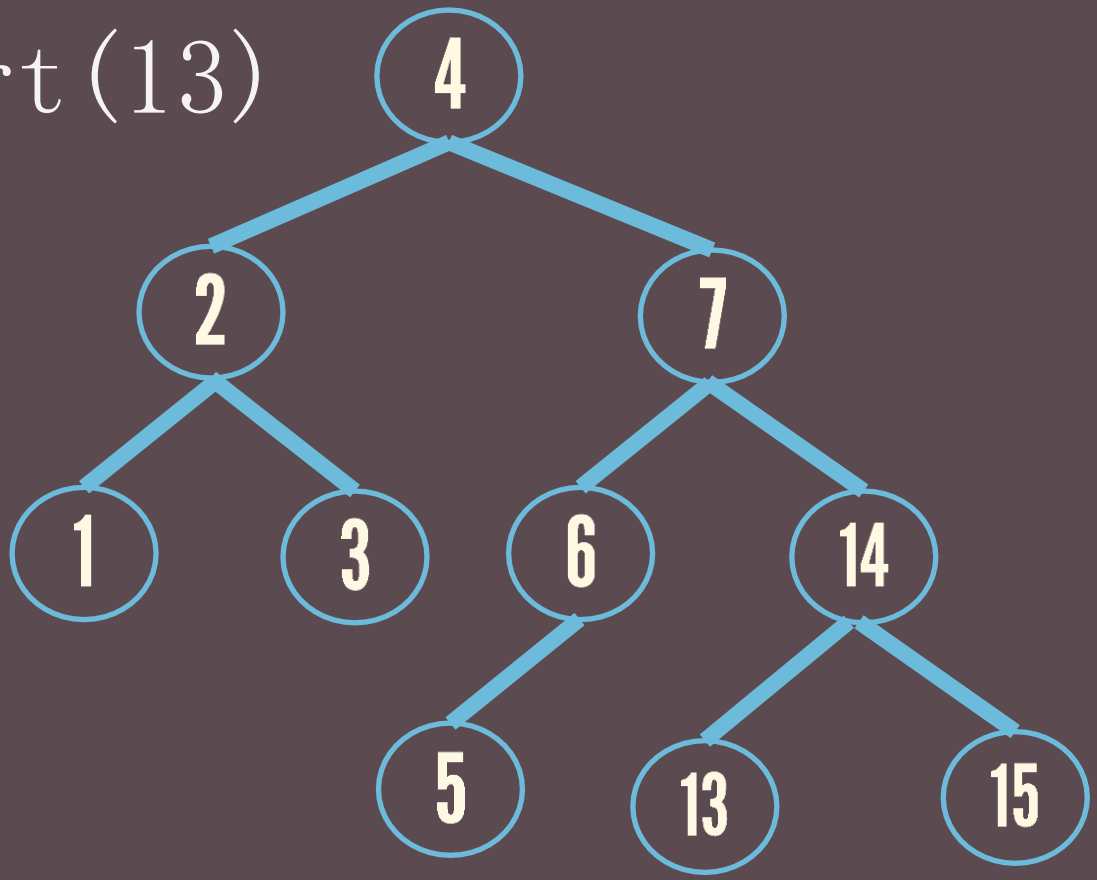
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(13)



insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

insert(13)



insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
insert(15)
insert(14)
insert(13)

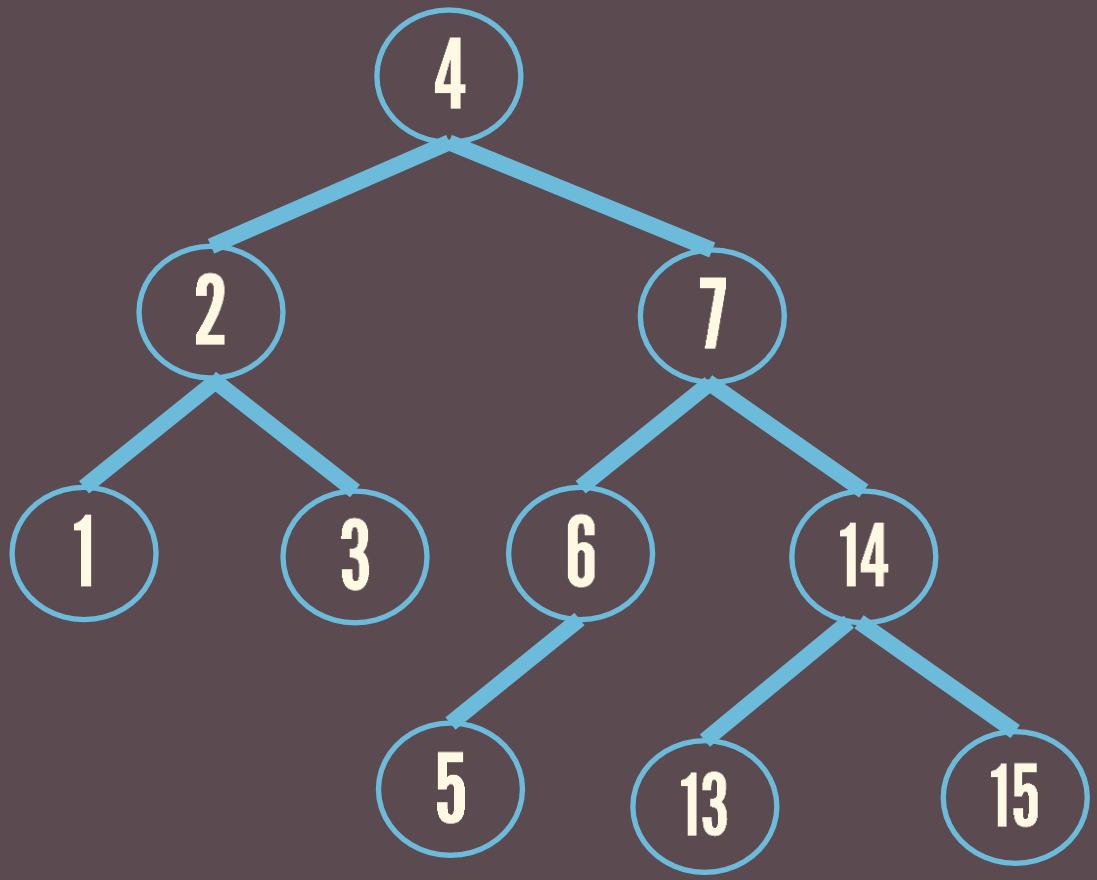
AVL

OPERATIONS

find
insert (with rotations)
delete (with rotations)
minimum
maximum
successor
predecessor

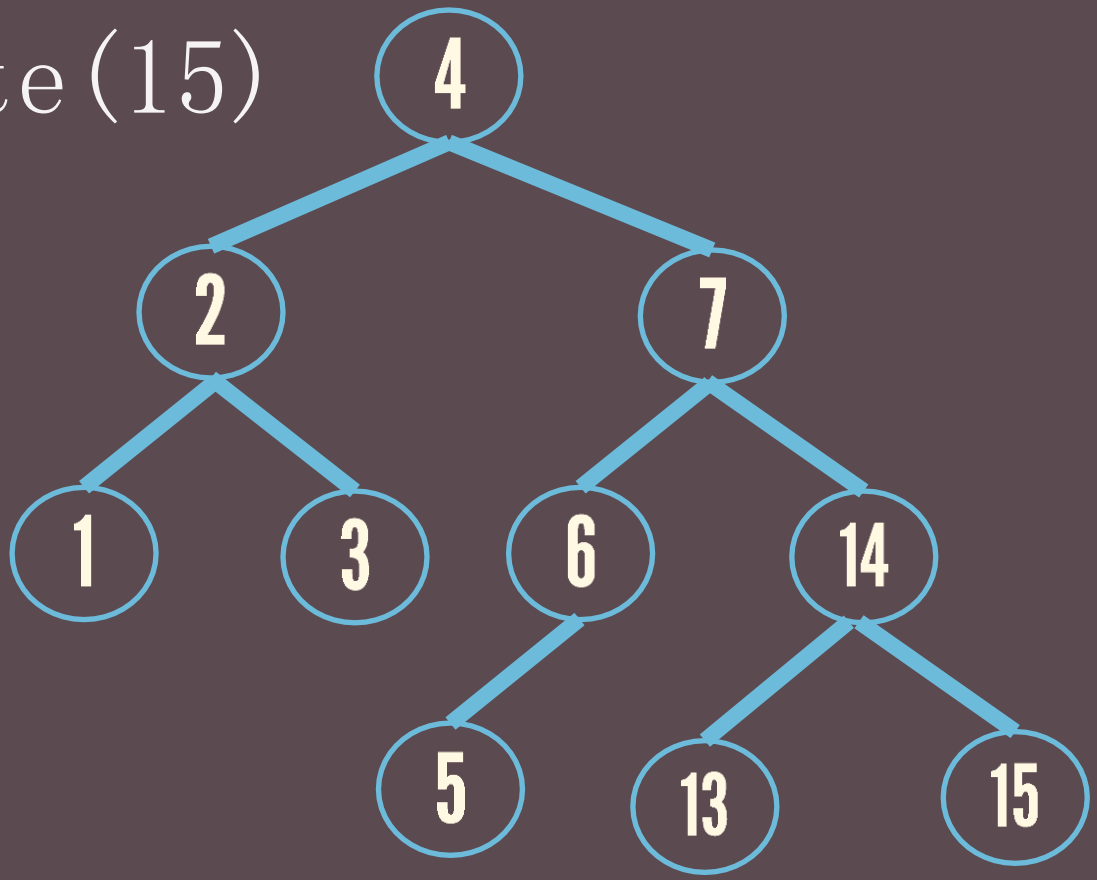
```
deleteNode(){

   if the node is a leaf,
     simply remove it.
   if the node is not a leaf,
     replace it with either its
     predecessor  or its successor and
     recursively  delete that node.
   perform fixup() (the insert fixup) on
   the parent of the repacement up to the
   root.

}
```

```
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)
```



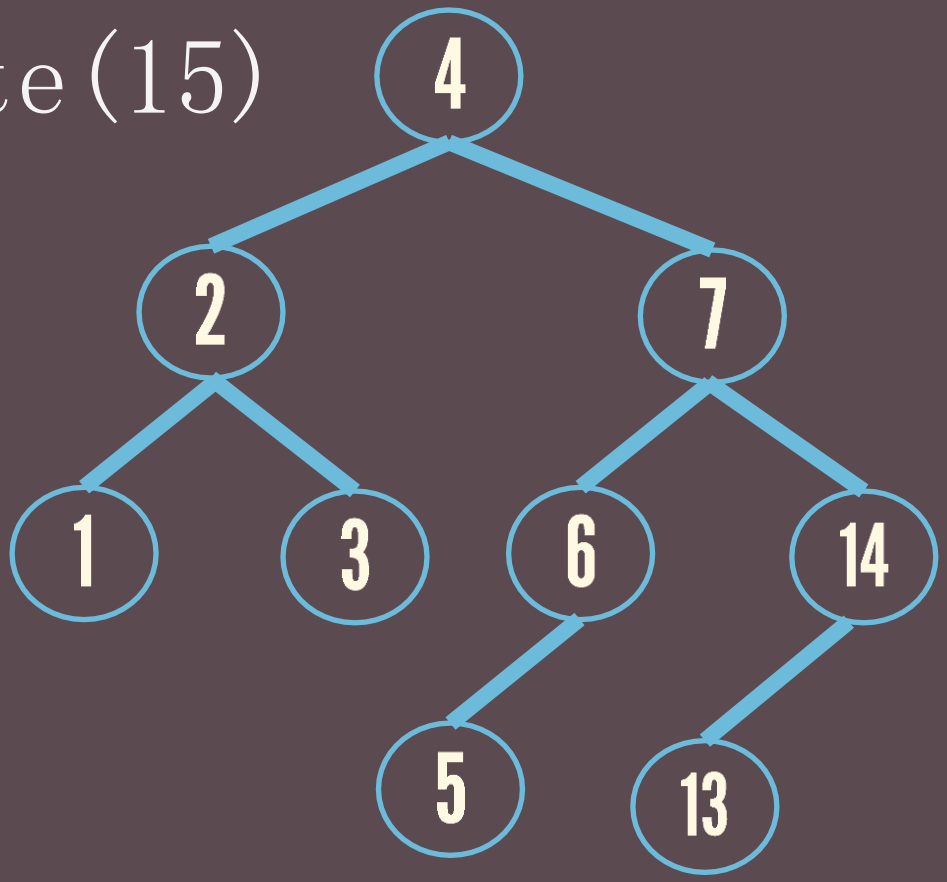
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(15)



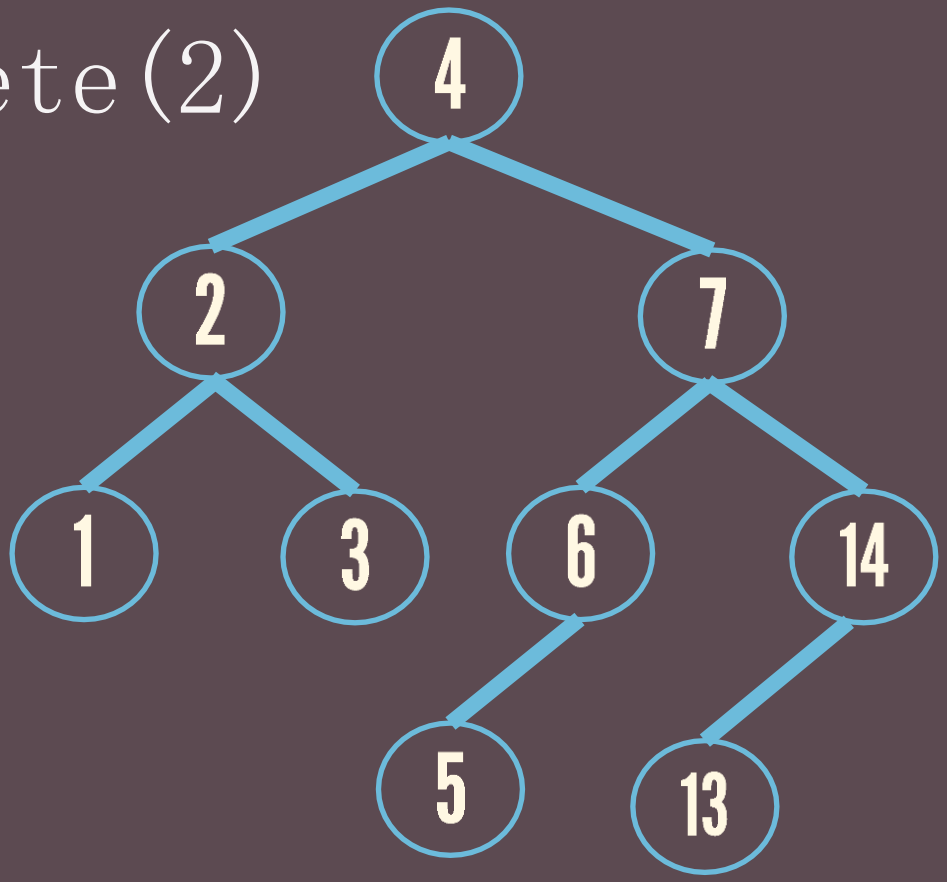
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(15)



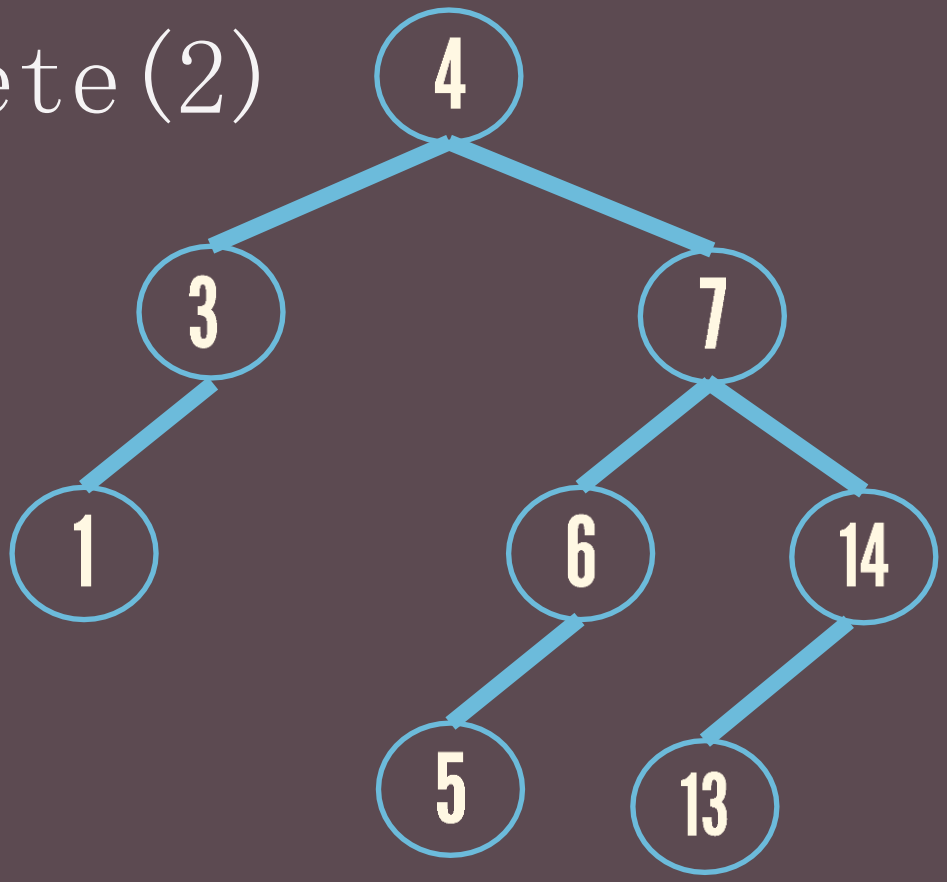
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(2)



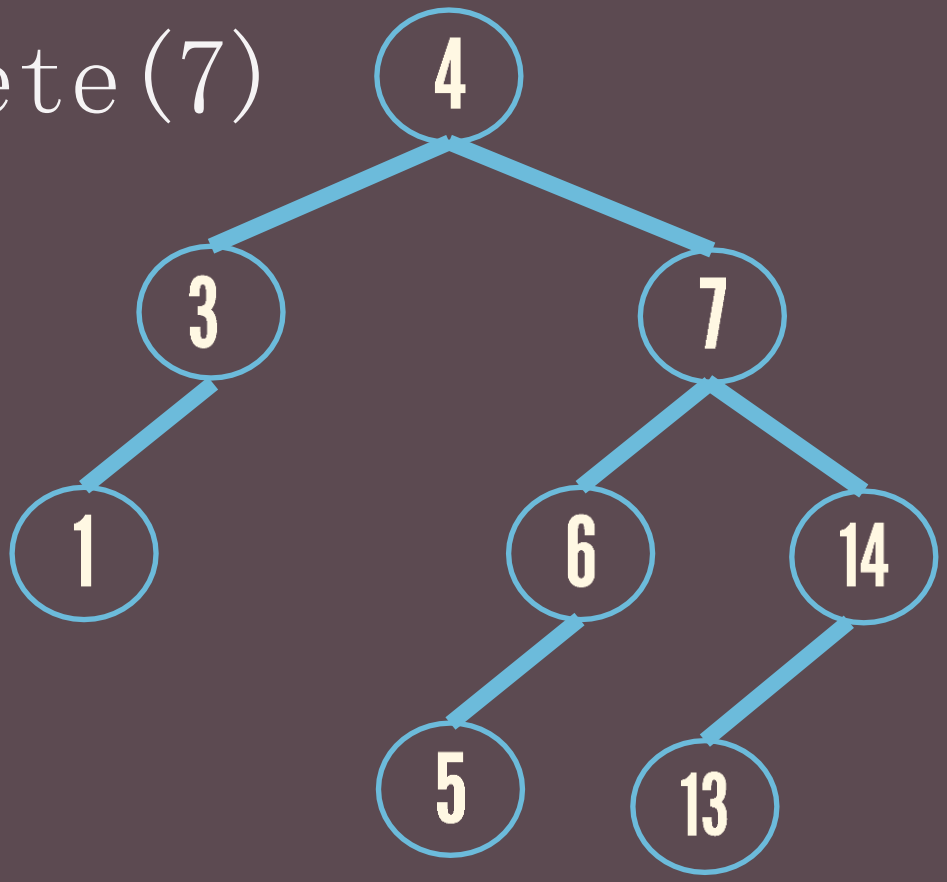
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(2)



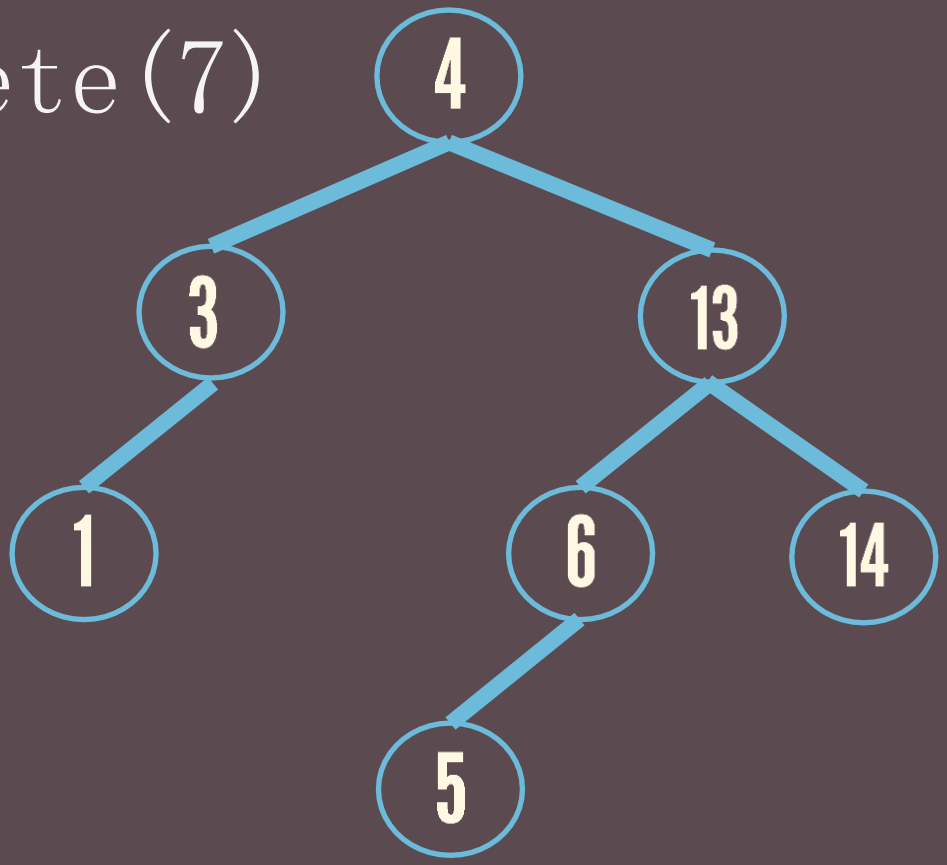
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(7)



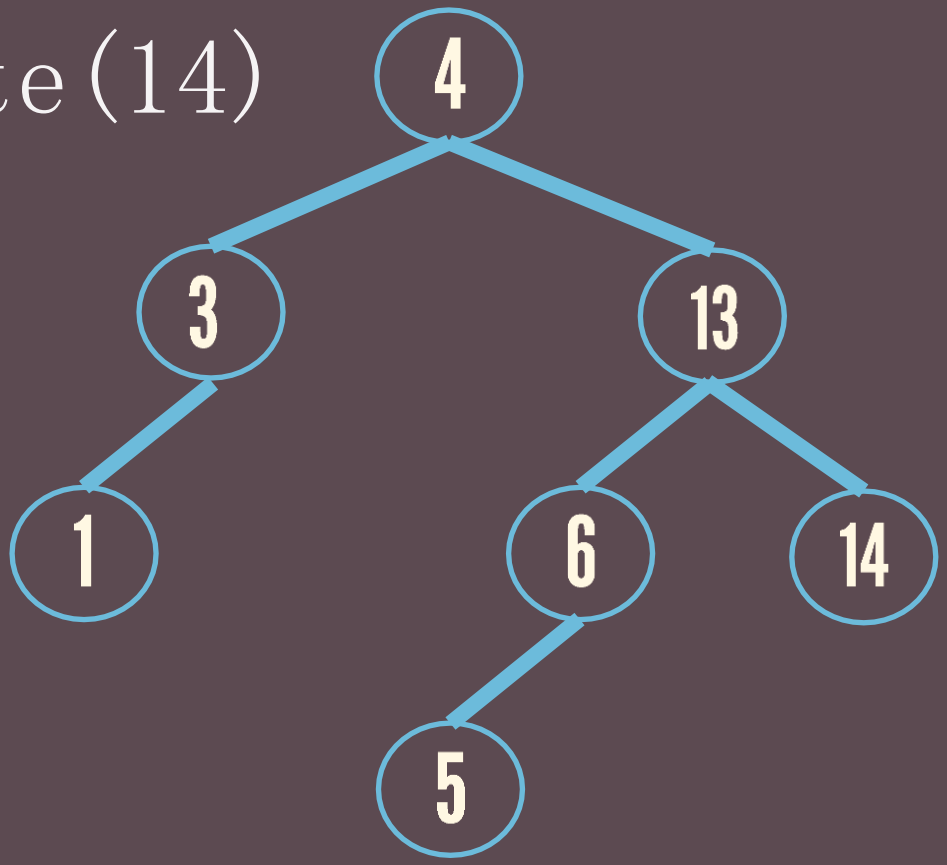
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(7)



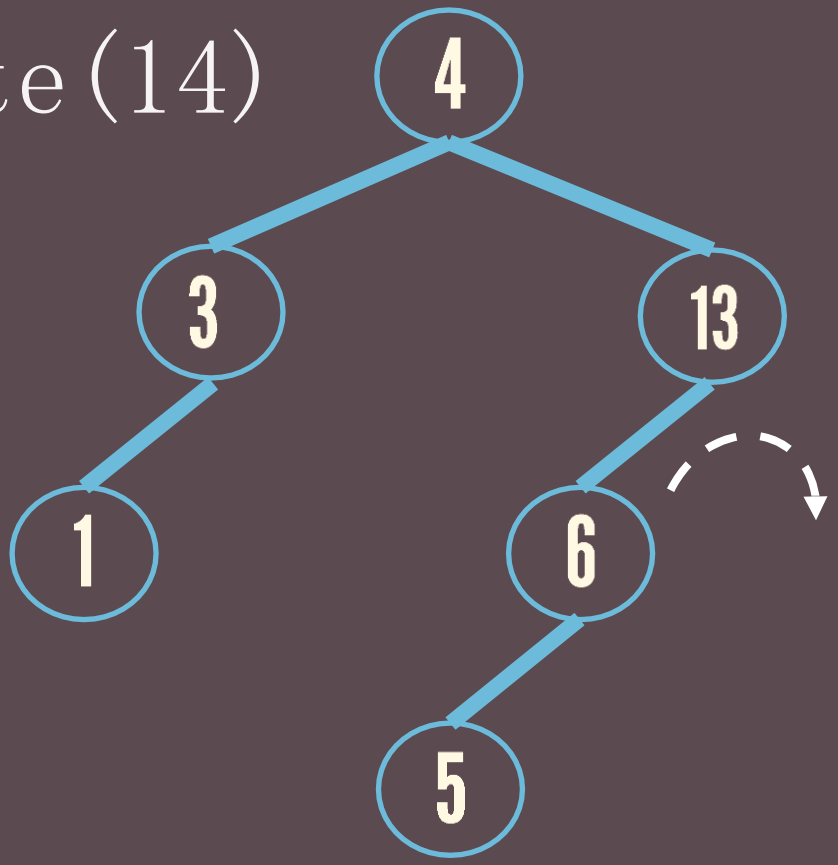
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(14)



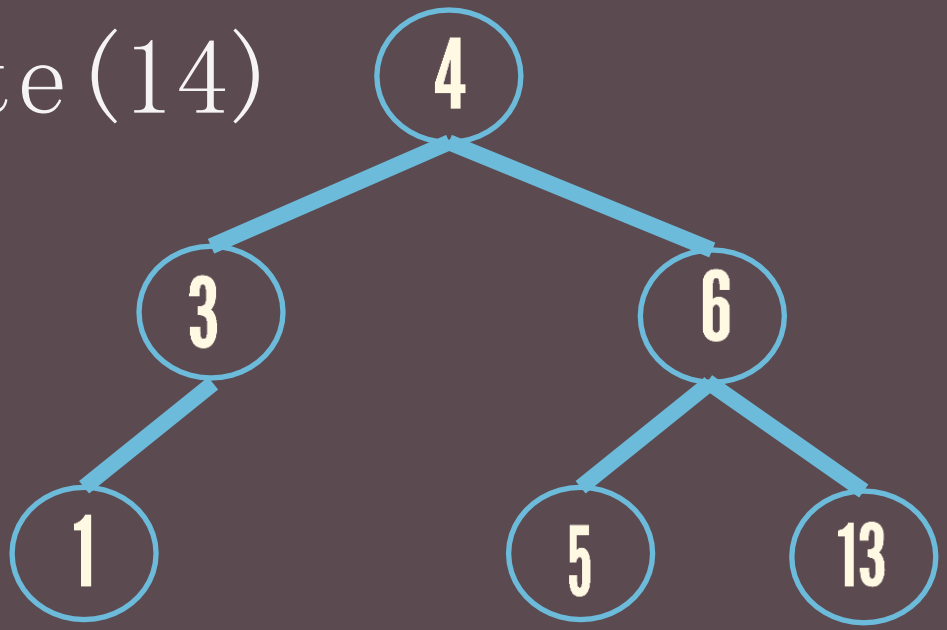
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(14)



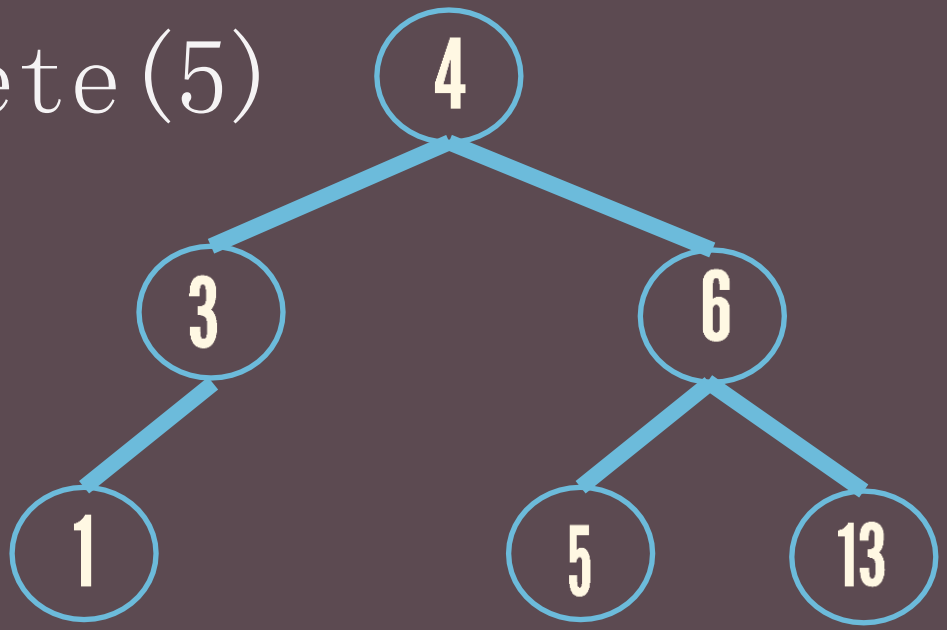
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(14)



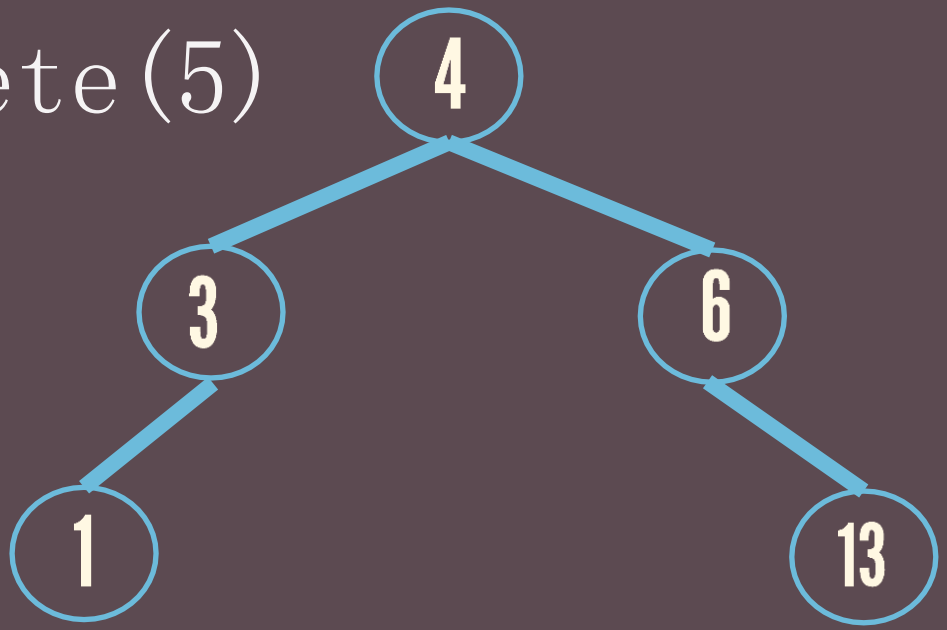
delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(5)

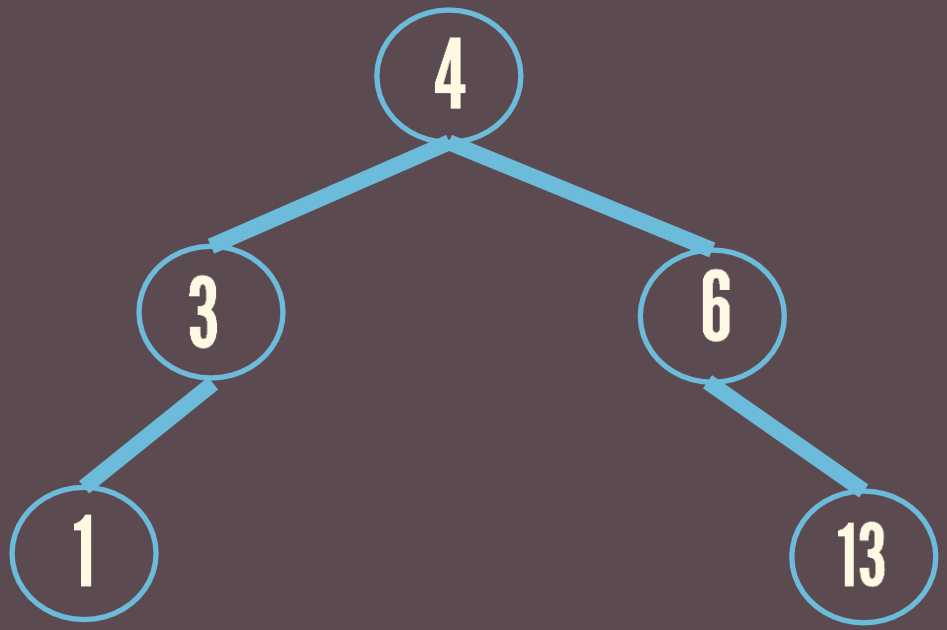


delete(15)
delete(2)
delete(7)
delete(14)
delete(5)

delete(5)



delete(15)
delete(2)
delete(7)
delete(14)
delete(5)



delete(15)
delete(2)
delete(7)
delete(14)
delete(5)