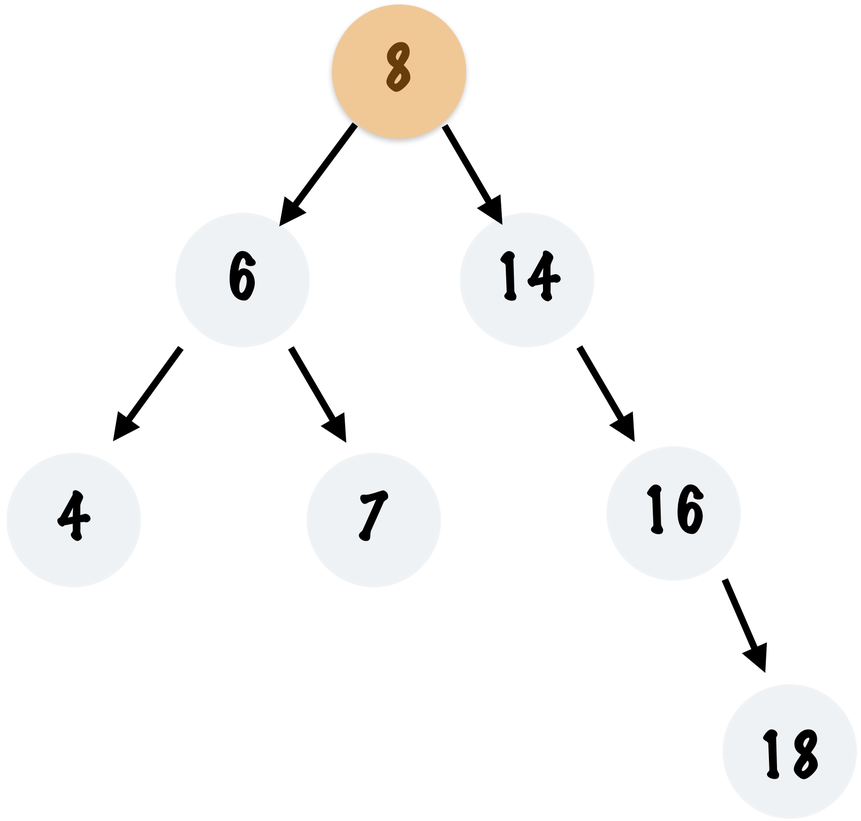
INSERTION INTO A BINARY SEARCH TREE

INSERTION

INSERT THE NODE 2
INTO THIS TREE

2

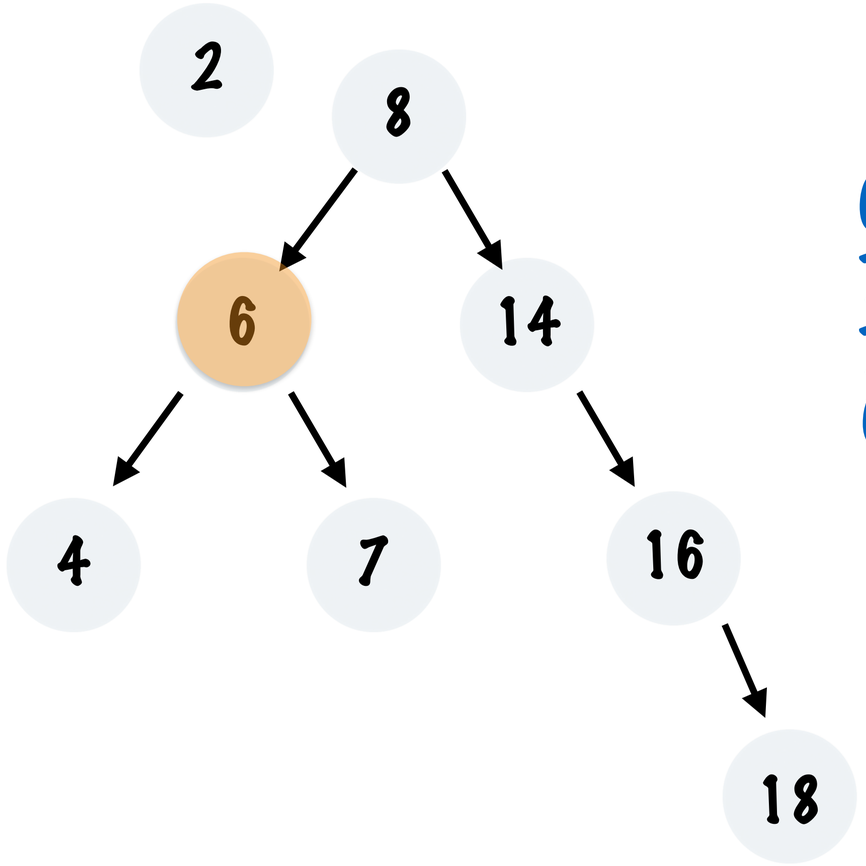


COMPARE THE NODE
TO BE INSERTED WITH
THE ROOT OF THE
TREE

INSERTION

2 < 8 SO WE MOVE DOWN THE LEFT SUB TREE

8 HAS A LEFT CHILD SO WE CONTINUE COMPARING NODE VALUES

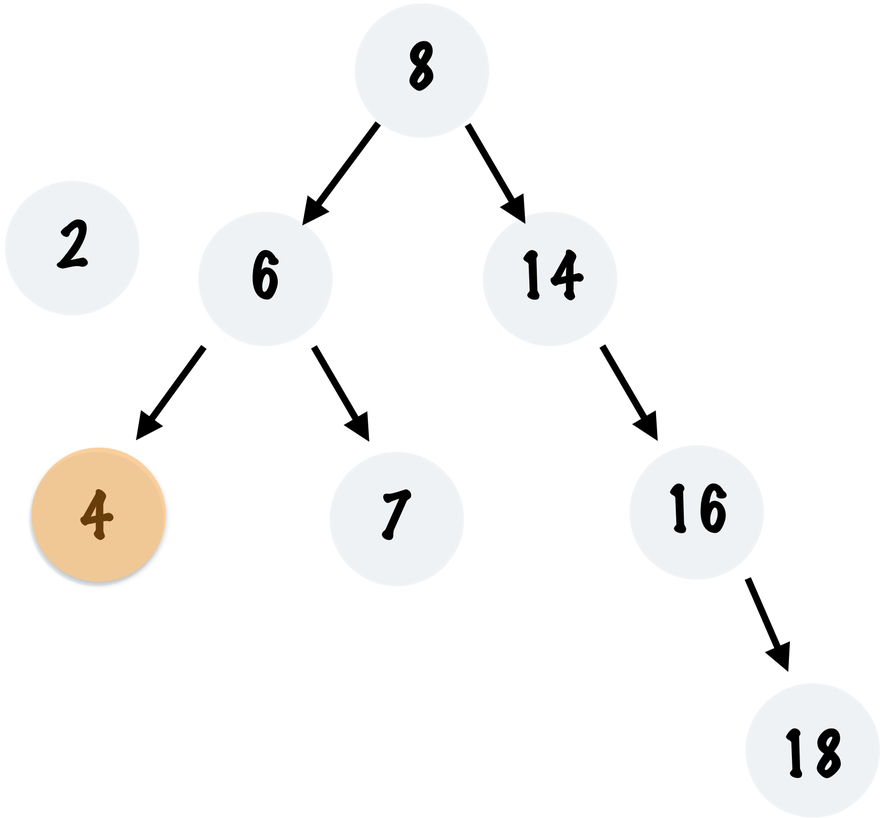


COMPARE THE NODE TO BE INSERTED WITH THE NODE WITH VALUE 6

INSERTION

2 < 6 SO WE MOVE DOWN THE LEFT SUB TREE

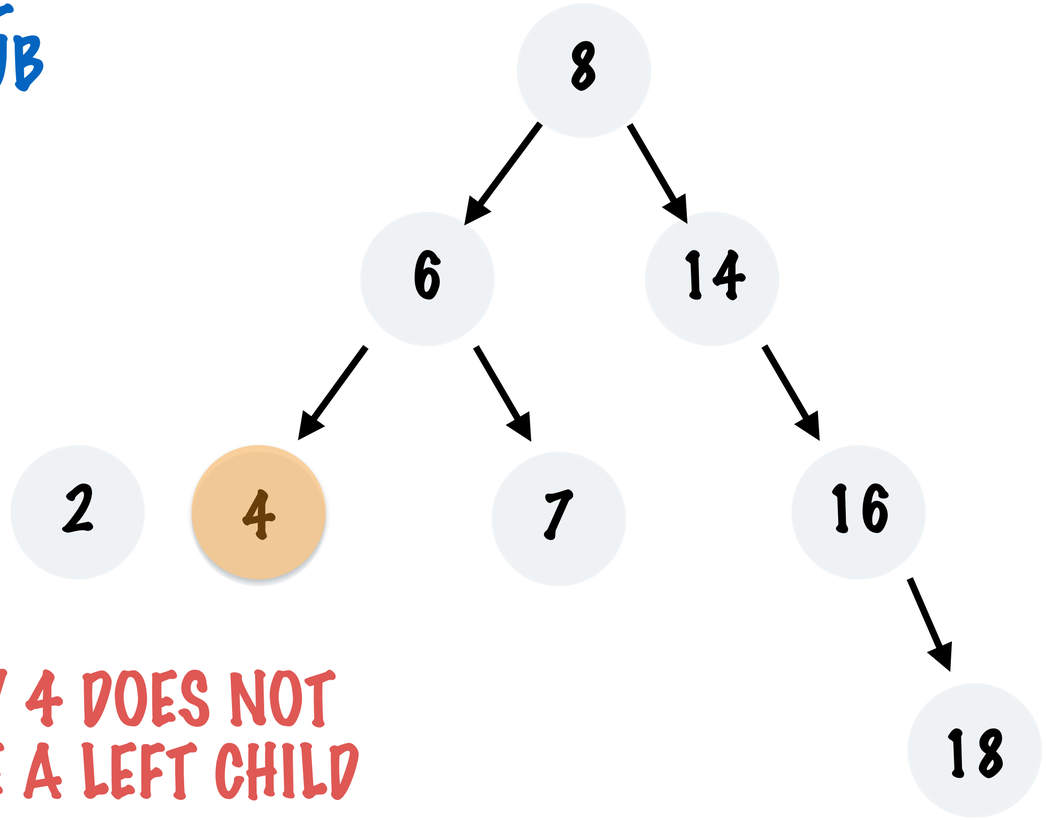
6 ALREADY HAS A LEFT CHILD SO WE CONTINUE COMPARING NODE VALUES



COMPARE THE NODE TO BE INSERTED WITH THE NODE WITH VALUE 4

INSERTION

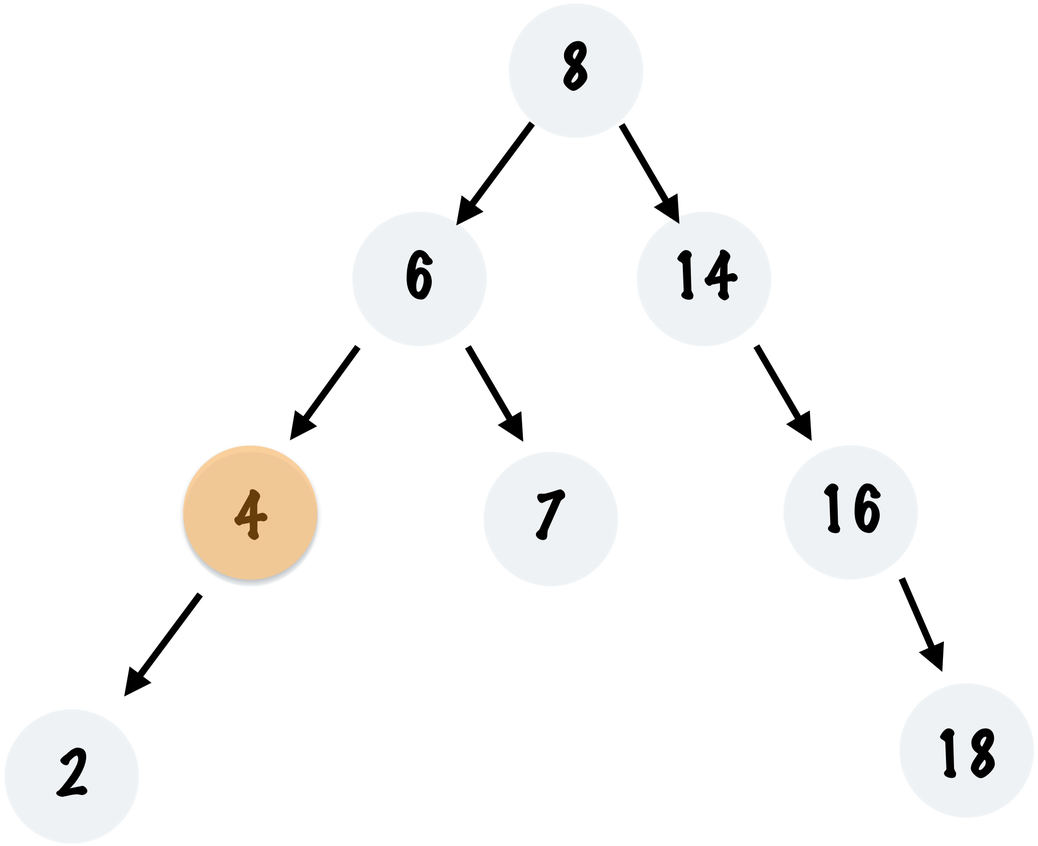
2 < 4 SO WE MOVE DOWN THE LEFT SUB TREE



INSERT 2 AS THE LEFT CHILD OF 4

NOW 4 DOES NOT HAVE A LEFT CHILD

INSERTION

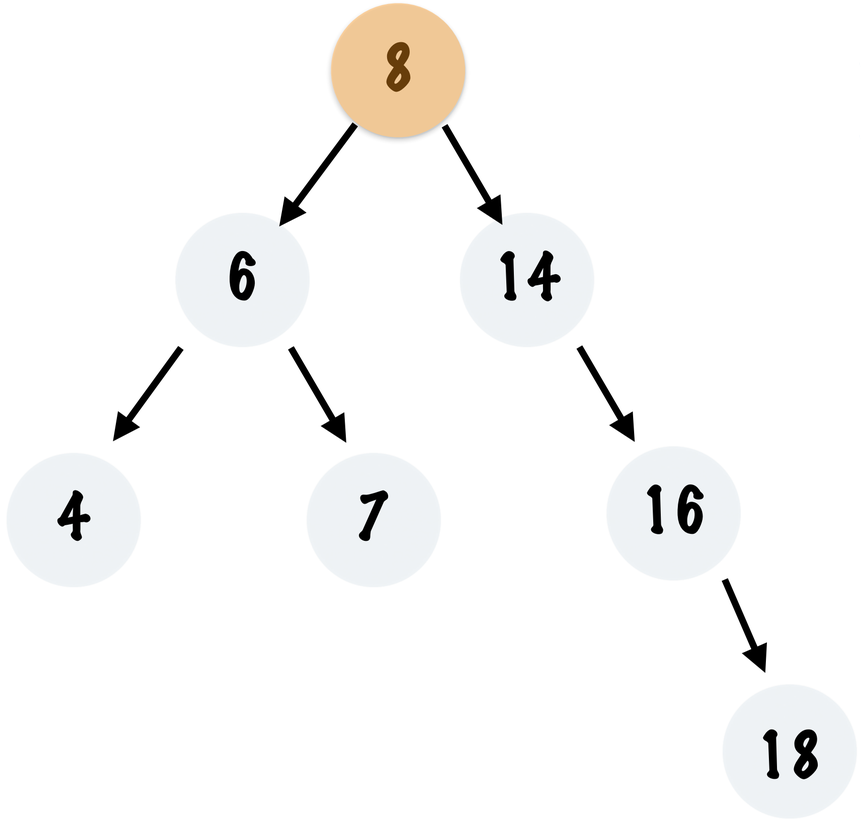


2 HAS BEEN INSERTED IN THE CORRECT POSITION

INSERTION

INSERT THE NODE 15
INTO THIS TREE

15

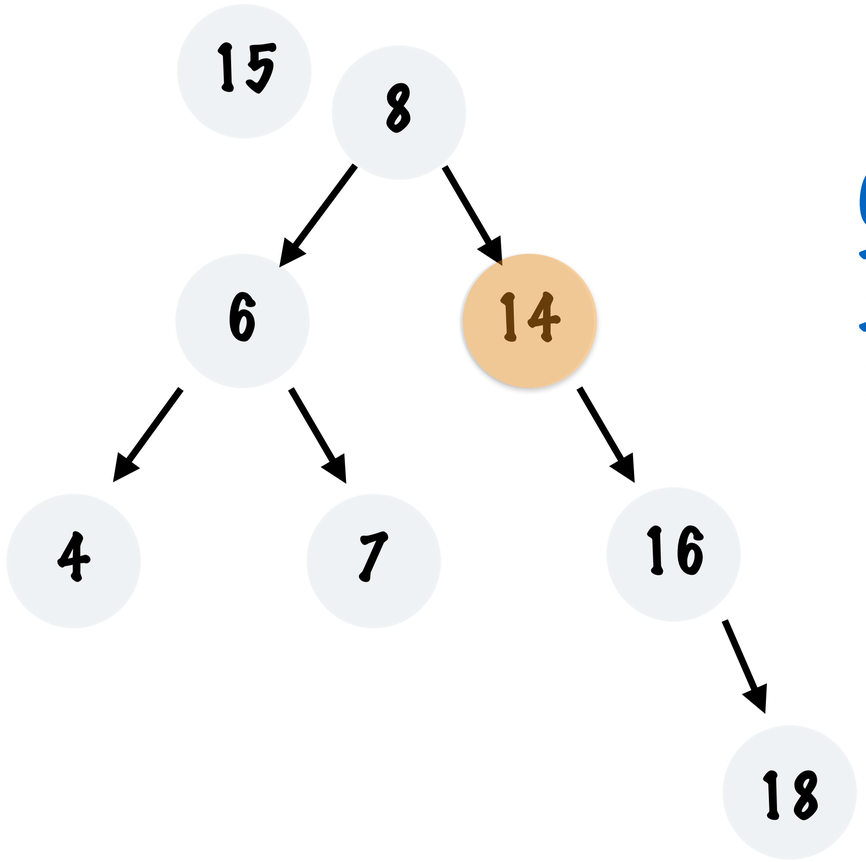


COMPARE THE NODE
TO BE INSERTED WITH
THE ROOT OF THE
TREE

INSERTION

15 > 8 SO WE MOVE DOWN THE RIGHT SUB TREE

8 HAS A RIGHT CHILD SO WE CONTINUE COMPARING NODE VALUES

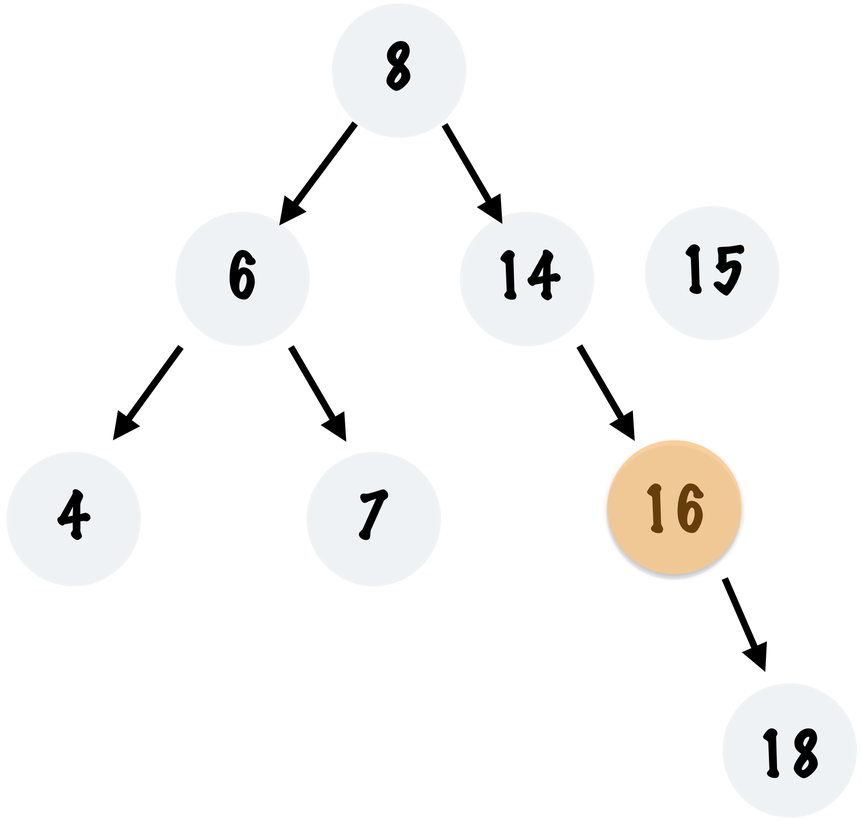


COMPARE THE NODE TO BE INSERTED WITH THE NODE WITH VALUE 14

INSERTION

15 > 14 SO WE MOVE DOWN THE RIGHT SUB TREE

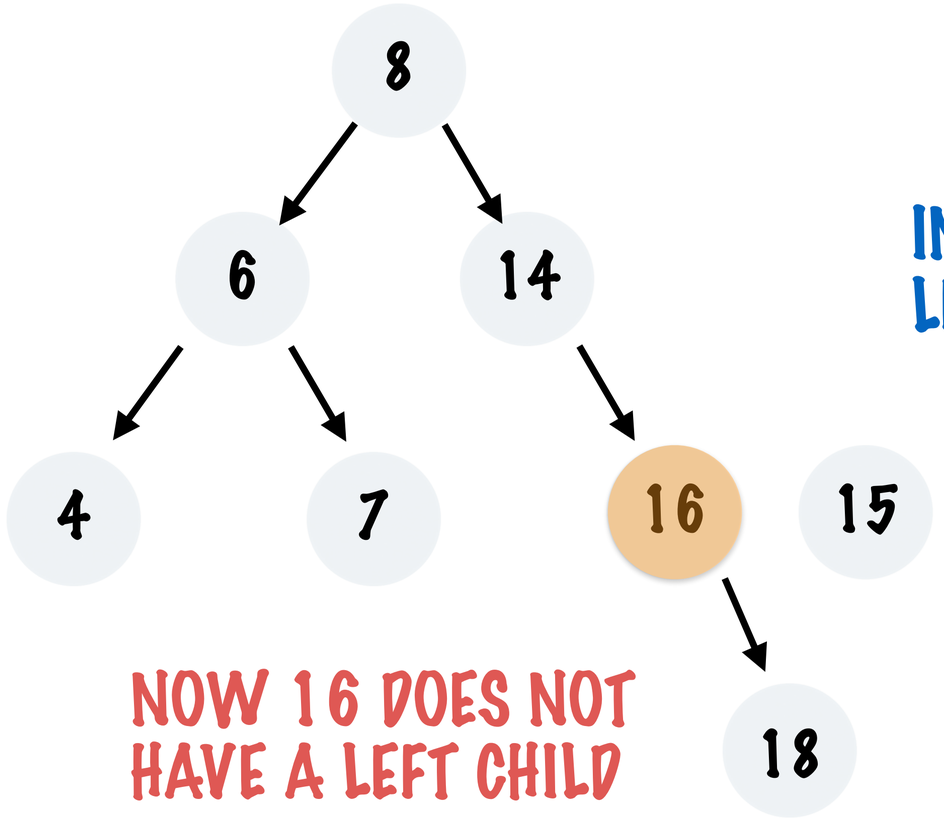
14 ALREADY HAS A RIGHT CHILD SO WE CONTINUE COMPARING NODE VALUES



COMPARE THE NODE TO BE INSERTED WITH THE NODE WITH VALUE 16

INSERTION

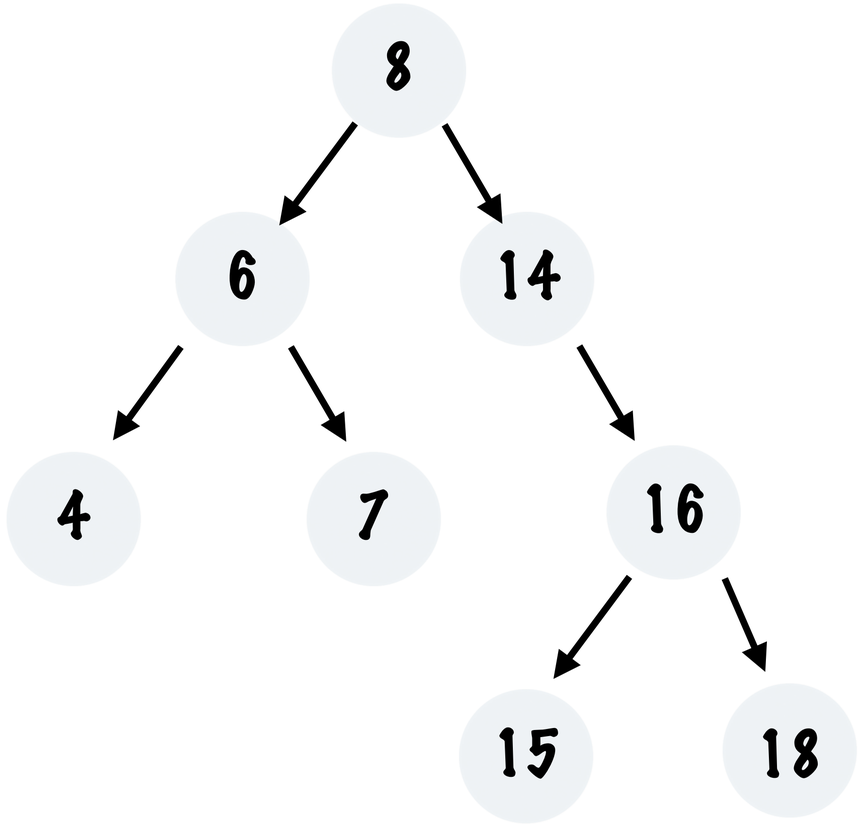
15 < 16 SO WE MOVE DOWN THE LEFT SUB TREE



INSERT 15 AS THE LEFT CHILD OF 16

NOW 16 DOES NOT HAVE A LEFT CHILD

INSERTION



15 HAS BEEN INSERTED IN THE CORRECT POSITION

INSERTION CODE

```
public static Node<Integer> insert(Node<Integer> head, Node<Integer> node) {
    if (head == null) {
        return node;
    }

    if (node.getData() <= head.getData()) {
        head.setLeftChild(insert(head.getLeftChild(), node));
    } else {
        head.setRightChild(insert(head.getRightChild(), node));
    }

    return head;
}
```

BASE CASE, IF THE HEAD IS NULL THEN THE NODE ITSELF IS THE HEAD

IF THE NODE VALUES IS SMALLER THAN THE HEAD THEN IT'S CORRECT PLACE IS SOMEWHERE IN THE LEFT SUB-TREE - WE INSERT THE NODE INTO THE LEFT SUB TREE

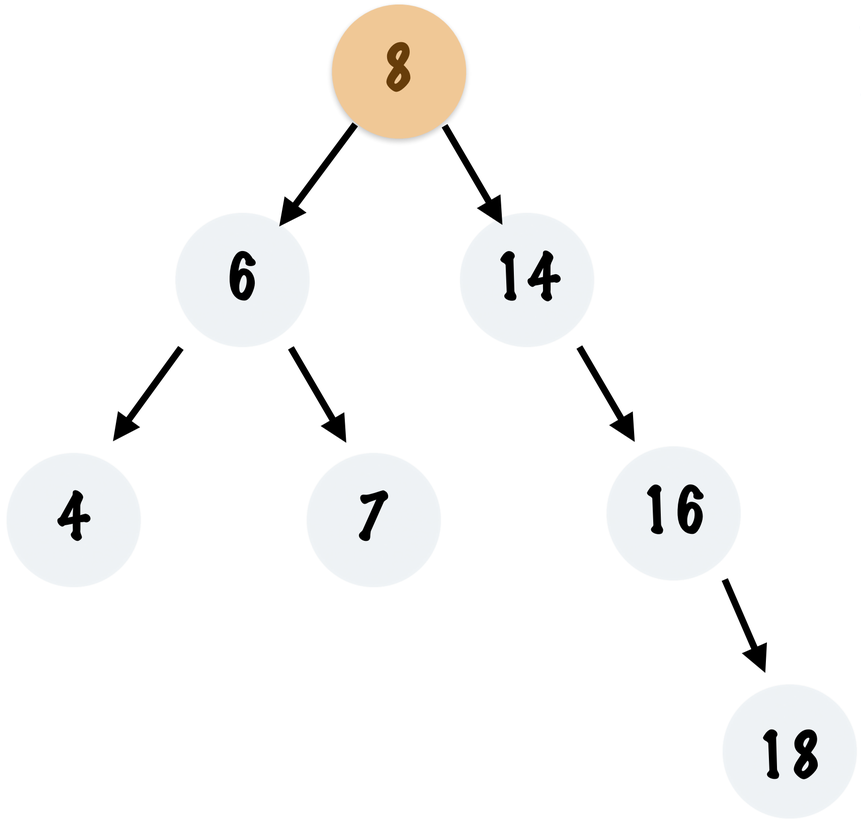
IF THE NODE VALUE IS GREATER THAN THE HEAD THEN IT'S CORRECT PLACE IS SOMEWHERE IN THE RIGHT SUB-TREE - WE INSERT THE NODE INTO THE RIGHT SUB TREE

LOOKUP IN A BINARY SEARCH TREE

LOOKUP

LOOKUP THE VALUE 7
IN THIS TREE

7

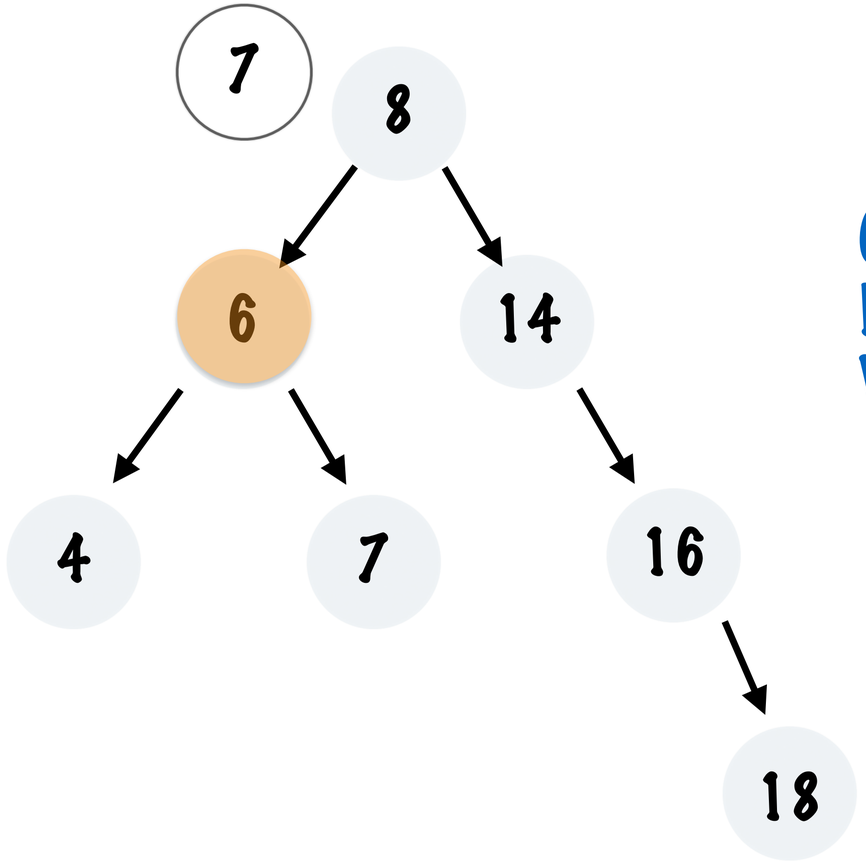


COMPARE THE VALUE
TO LOOK UP WITH THE
ROOT OF THE TREE

LOOKUP

7 < 8 SO WE MOVE DOWN THE LEFT SUB TREE

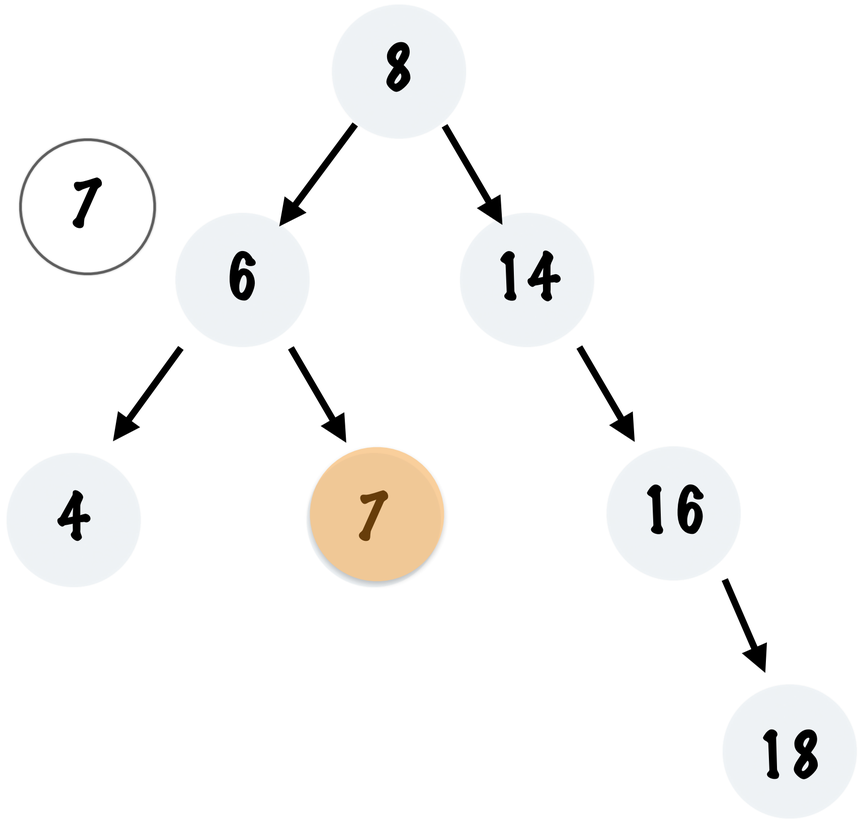
8 HAS A LEFT CHILD SO WE CONTINUE COMPARING NODE VALUES



COMPARE THE LOOKUP NODE WITH THE NODE WITH VALUE 6

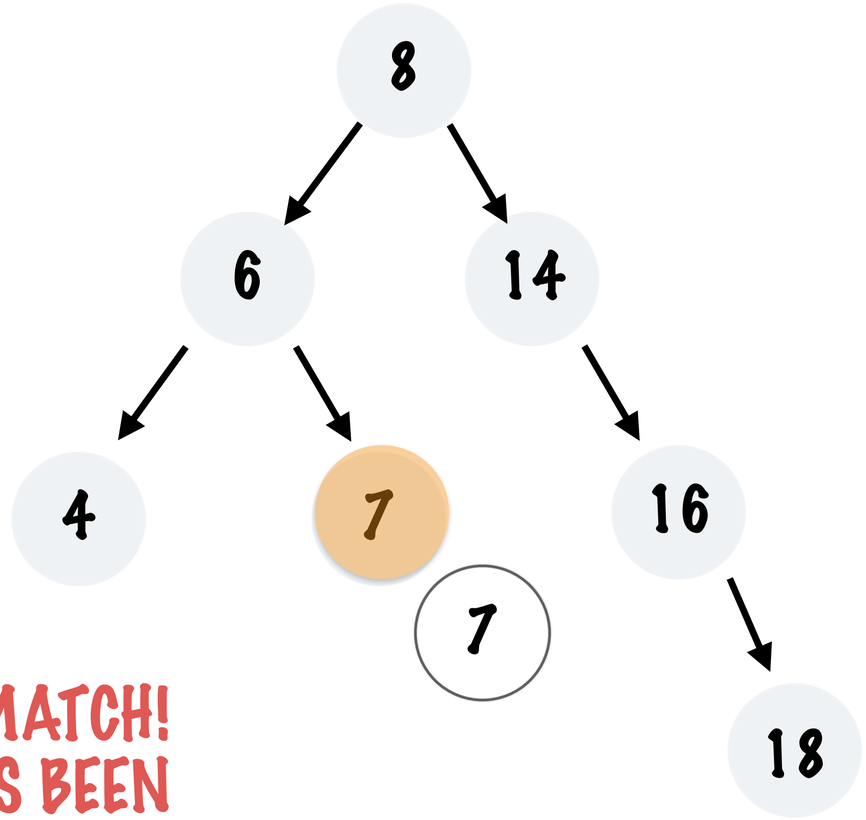
LOOKUP

7 > 6 SO WE MOVE DOWN THE RIGHT SUB TREE



COMPARE THE LOOKUP NODE WITH THE NODE WITH VALUE 7

LOOKUP

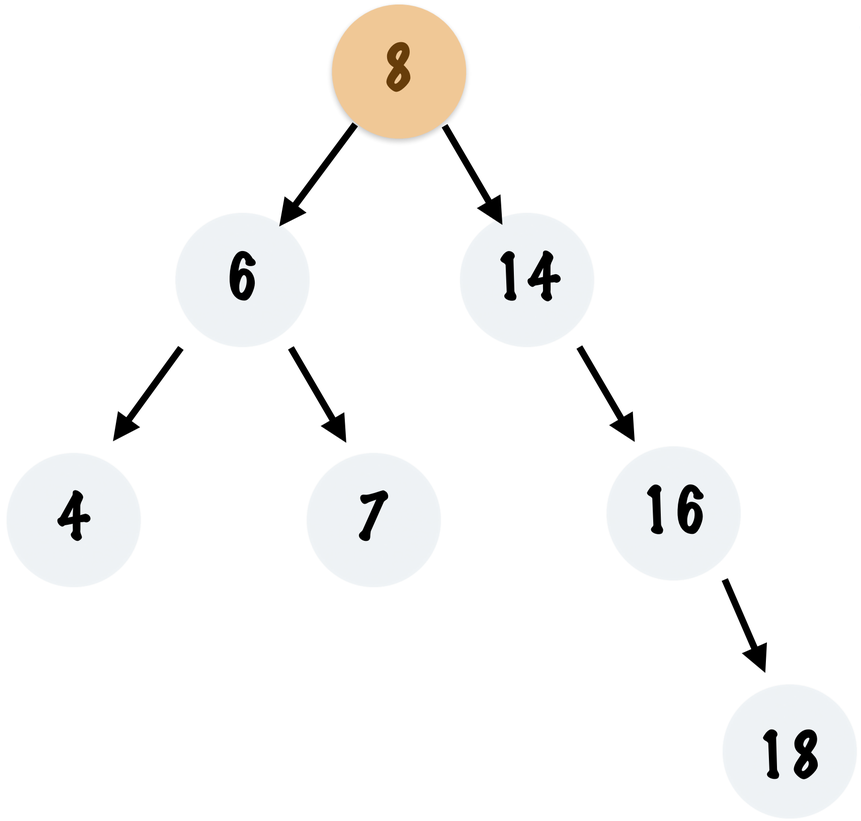


THERE IS A MATCH!
THE NODE HAS BEEN
FOUND

LOOKUP

LOOKUP THE VALUE 20
IN THIS TREE

20

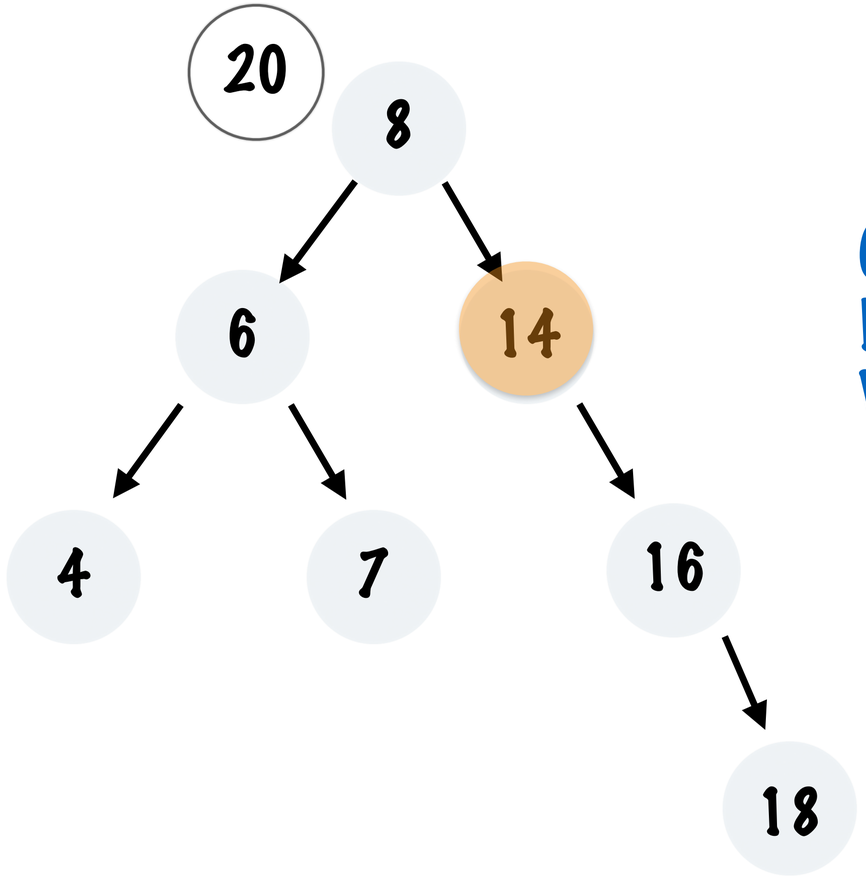


COMPARE THE VALUE
TO LOOK UP WITH THE
ROOT OF THE TREE

LOOKUP

20 > 8 SO WE MOVE DOWN THE RIGHT SUB TREE

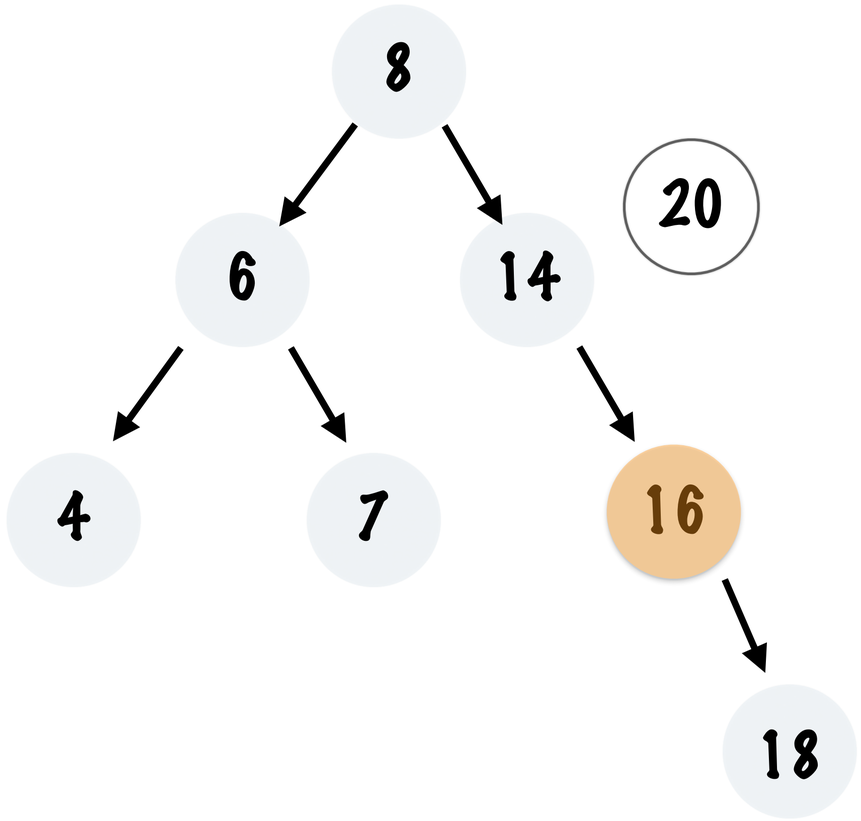
8 HAS A RIGHT CHILD SO WE CONTINUE COMPARING NODE VALUES



COMPARE THE LOOKUP NODE WITH THE NODE WITH VALUE 14

LOOKUP

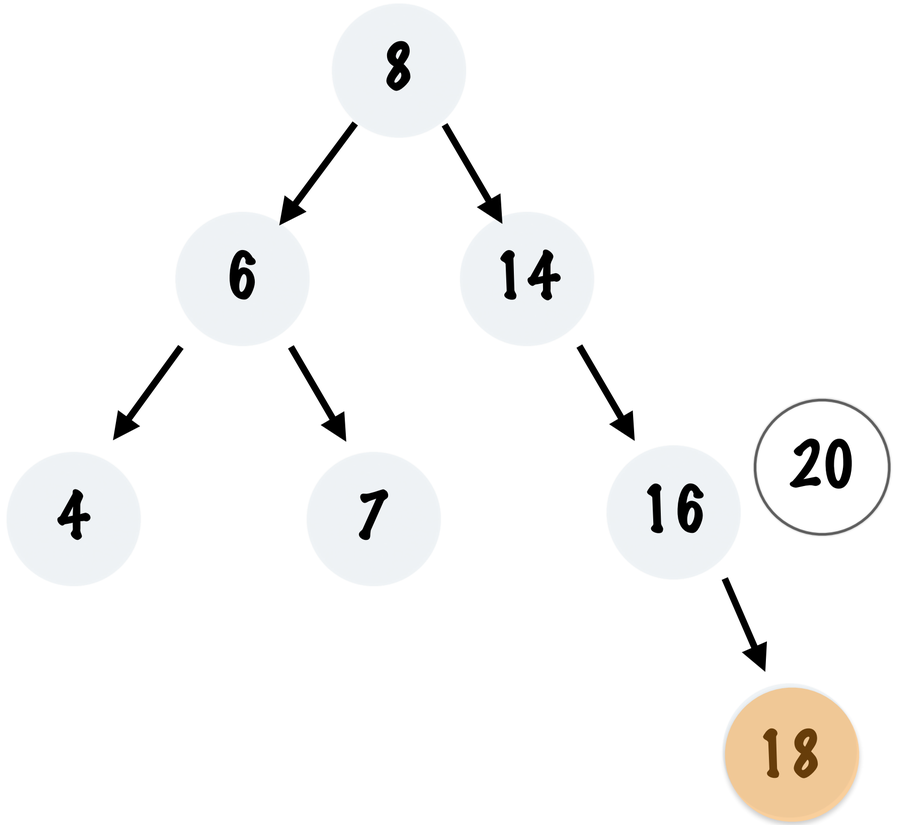
20 > 14 SO WE MOVE DOWN THE RIGHT SUB TREE



COMPARE THE LOOKUP NODE WITH THE NODE WITH VALUE 16

LOOKUP

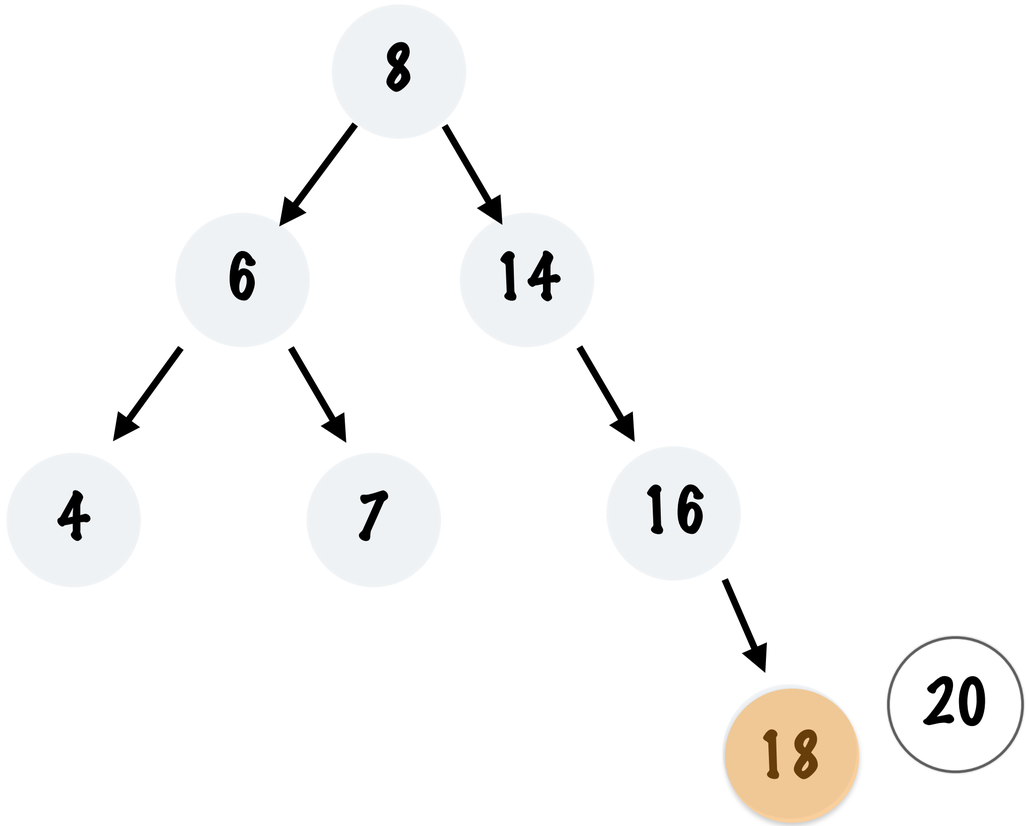
20 > 16 SO WE MOVE DOWN THE RIGHT SUB TREE



COMPARE THE LOOKUP NODE WITH THE NODE WITH VALUE 18

LOOKUP

20 > 18 SO WE NEED TO MOVE DOWN THE RIGHT SUB TREE



WE CANNOT MOVE FURTHER DOWN THE RIGHT SUBTREE - THE VALUE 20 IS NOT PRESENT IN THE TREE!

LOOKUP CODE

```
public static Node<Integer> lookup(Node<Integer> head, int data) {
    if (head == null) {
        return null;
    }
    if (head.getData() == data) {
        return head;
    }

    if (data <= head.getData()) {
        return lookup(head.getLeftChild(), data);
    } else {
        return lookup(head.getRightChild(), data);
    }
}
```

BASE CASE, IF THE HEAD IS NULL THEN THE NODE HAS NOT BEEN FOUND, RETURN NULL

CHECK IF THE VALUE OF THE HEAD MATCHES THE VALUE WE'RE LOOKING UP, IF YES THEN WE'VE FOUND A MATCH!

IF THE LOOKUP VALUE IS SMALLER THAN OR EQUAL TO THE HEAD THEN LOOKUP THE LEFT SUBTREE - OTHERWISE LOOKUP THE RIGHT SUBTREE