2-3 Trees

COS 326

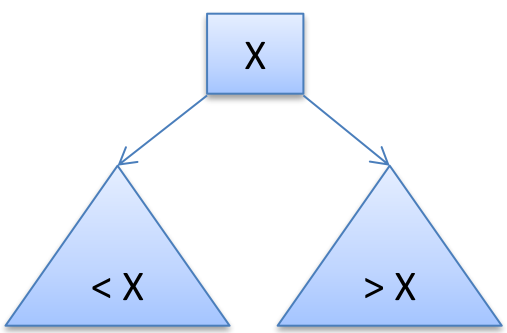
Assignment #5

Princeton University

2-3 Trees

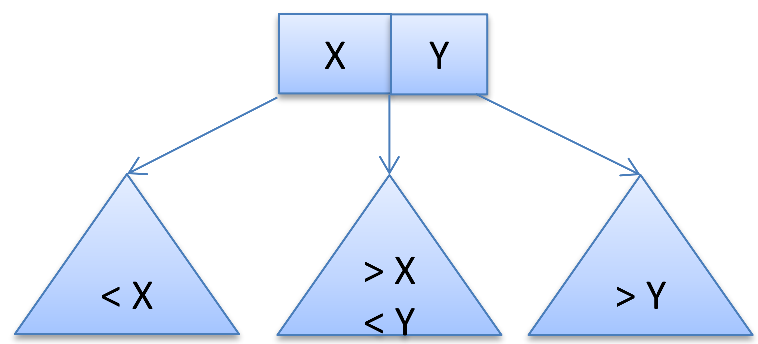
Leaf:

2-node:



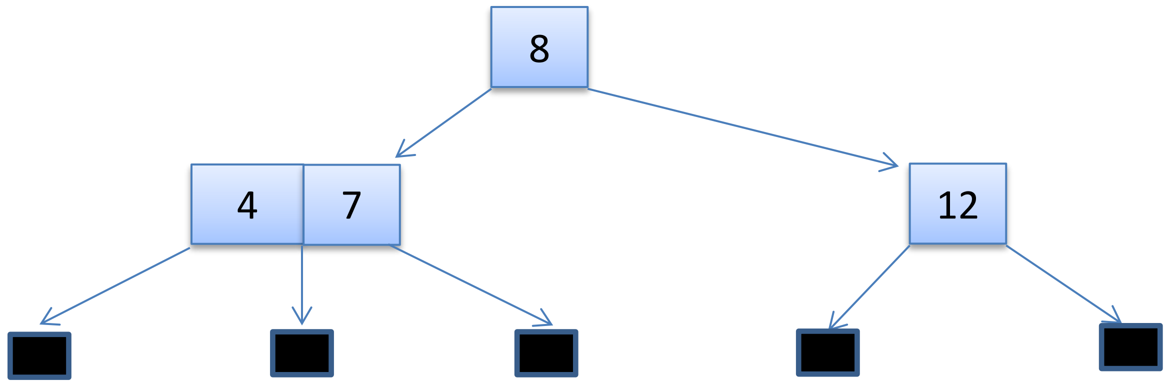
The height of both subtrees must be the same

3-node:

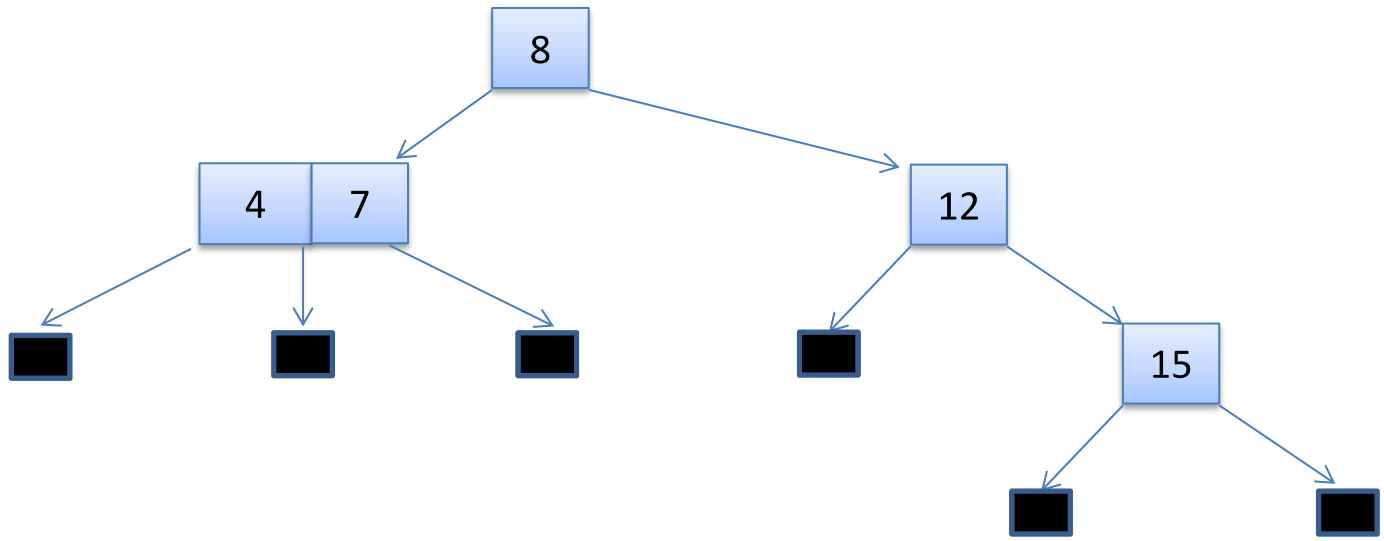


The height of all subtrees must be the same

2-3 Tree Example

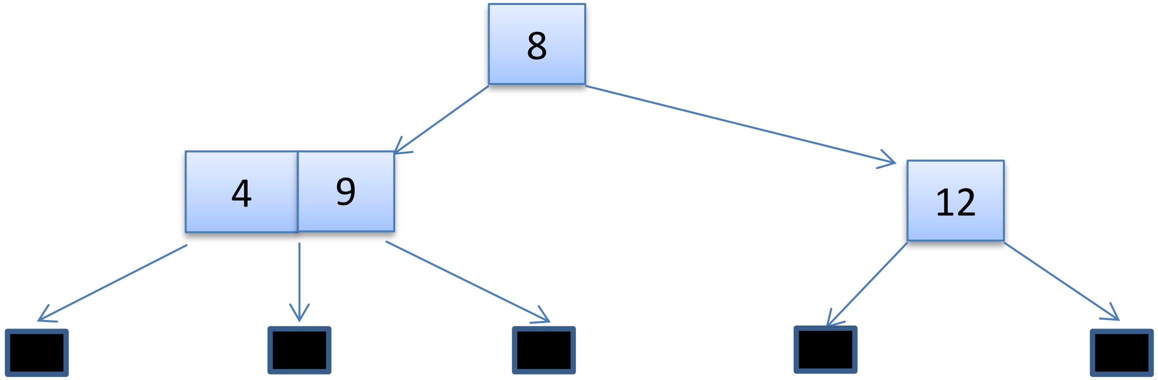


2-3 Tree *Non*-Example



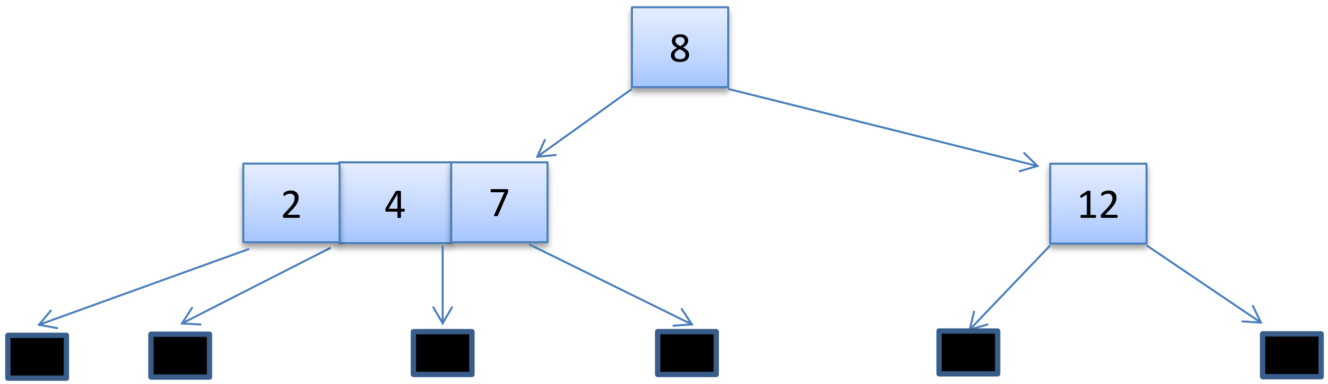
unequal subtree height!

2-3 Tree *Non*-Example



out of order keys!

2-3 Tree *Non*-Examples

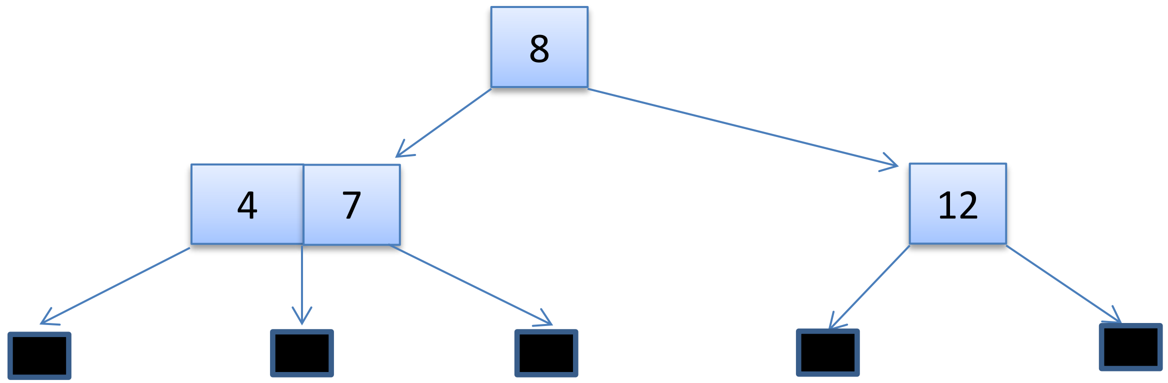


1-4-7 has too many keys – not a 3-node!

INSERT

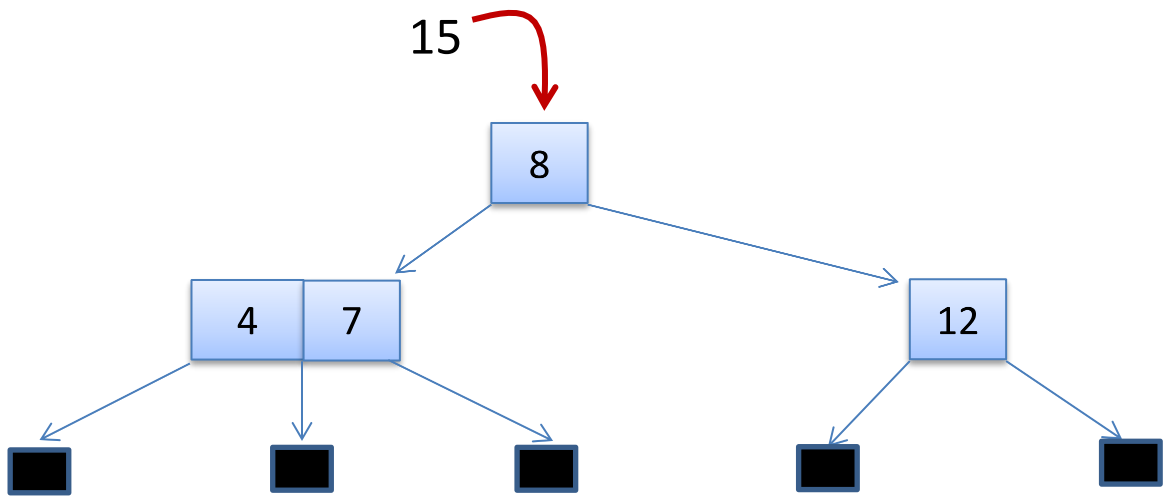
How to Insert

insert 15 into:



How to Insert

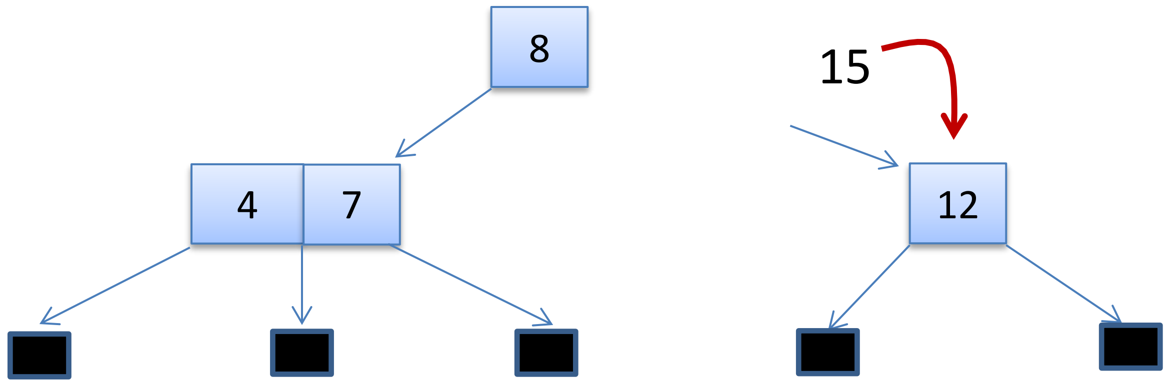
insert 15 into:



compare 15 to the root node

How to Insert

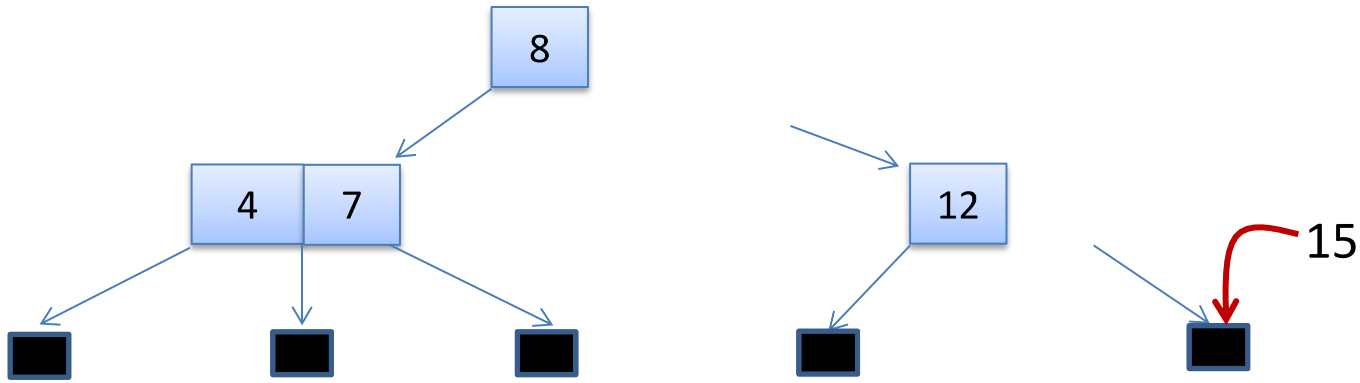
insert 15 into:



recursively insert into the right subtree

How to Insert

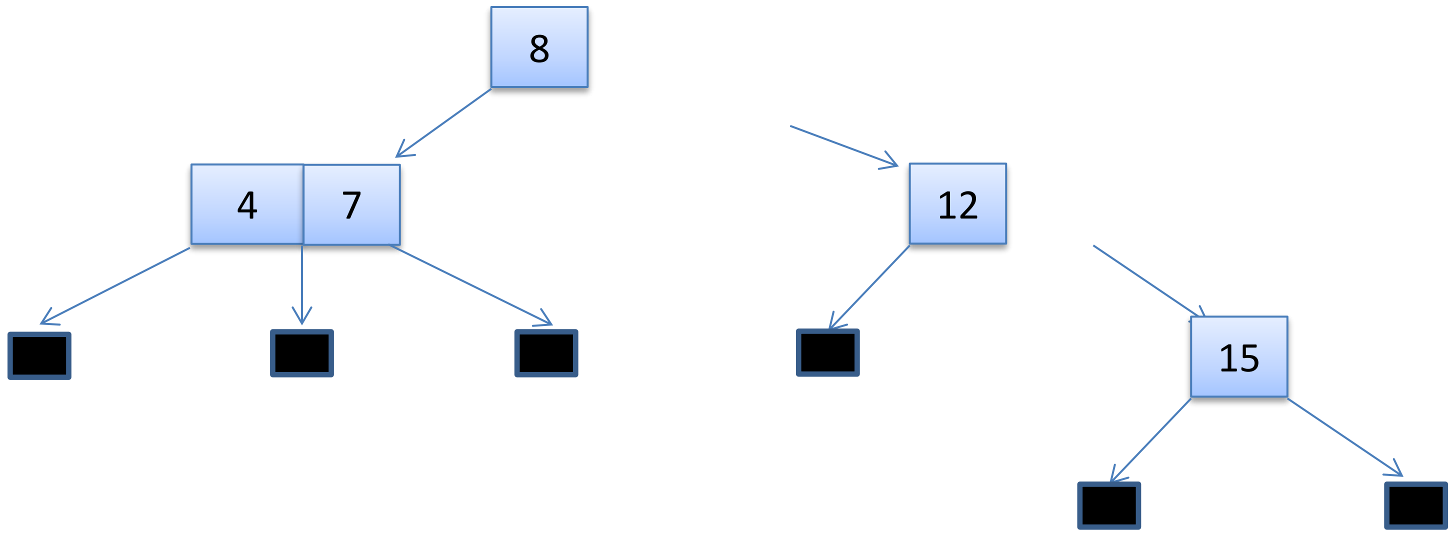
insert 15 into:



reach a leaf node

How to Insert

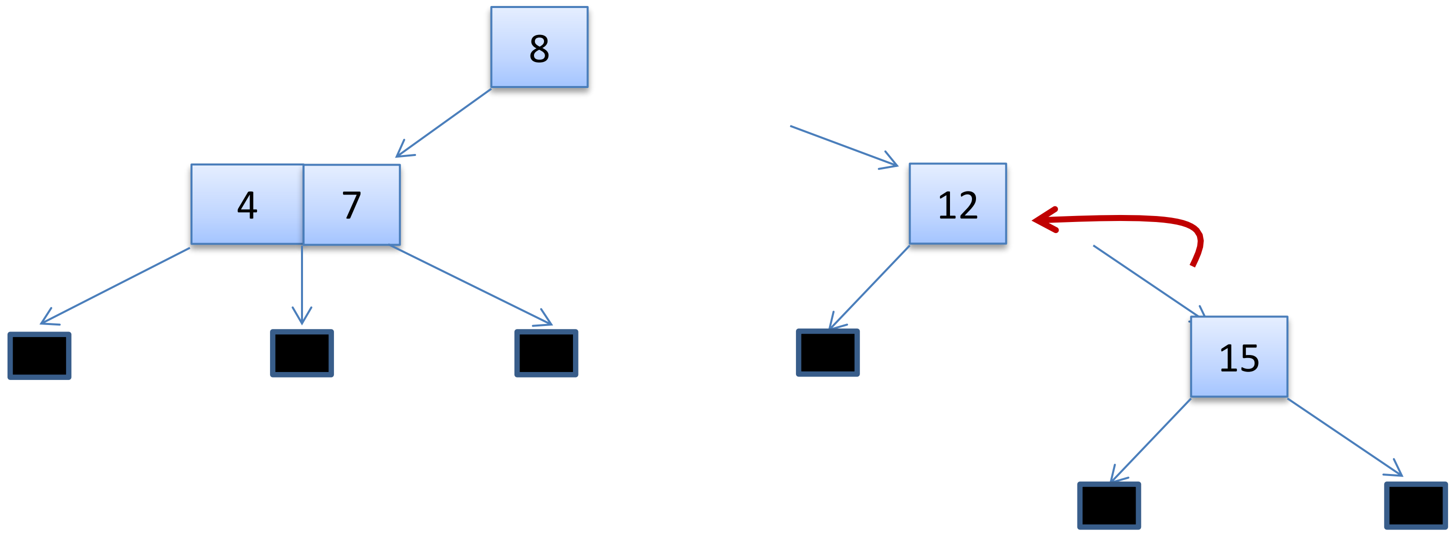
insert 15 into:



create a new subtree with 15

How to Insert

insert 15 into:



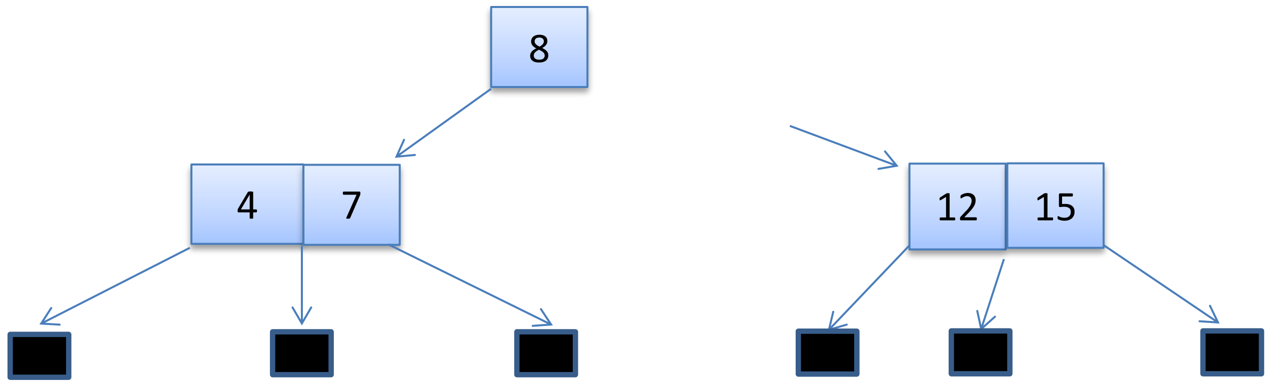
Return from recursive insert

Note:

- The height of the subtree has grown by 1
- It grew from height 0 (a leaf) to height 1 (tree with one node)
- If we include the new subtree in node 12 where the old subtree was then we will have children of uneven height.

How to Insert

insert 15 into:



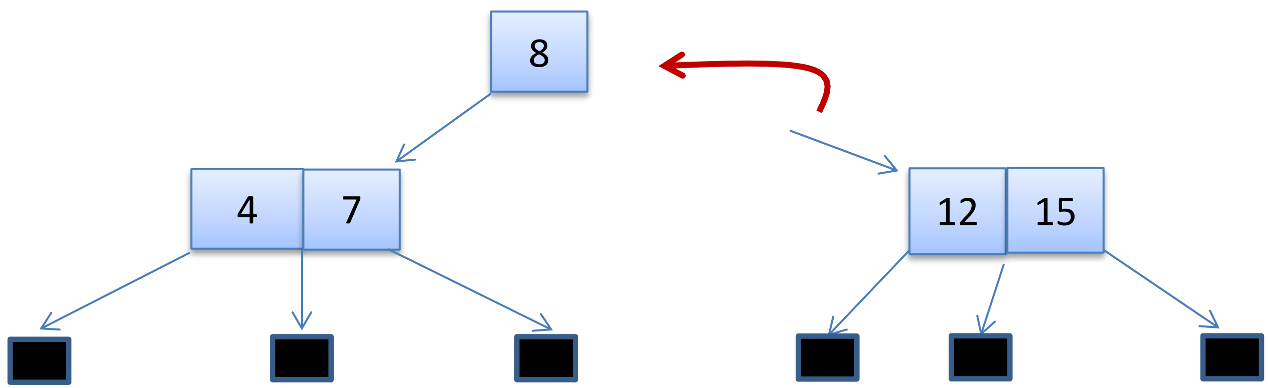
Solution: Turn a 2-node into a 3-node

Note:

- The height of the new subtree with root 12-15 is *the same* as the height of the original subtree that just contained 12

How to Insert

insert 15 into:



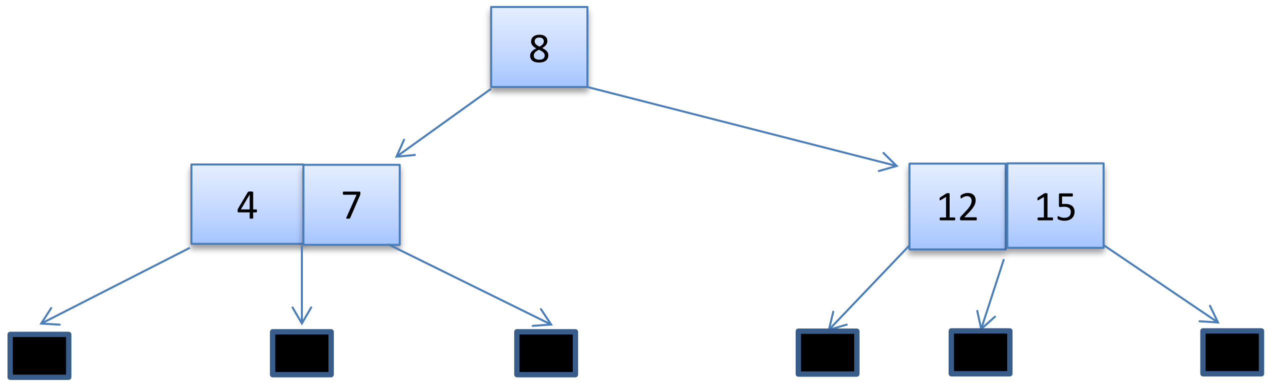
Return from recursive call to insert from 8-node

Note:

- the height of the 12-15 node is *the same* as the height of the original subtree of 8
- that means the new node also has the same height as the 4-7 child of 8
- since the heights of the two children are the same, the 12-15 node may be included directly as a child of the 8-node

How to Insert

insert 15 into:



We are done!

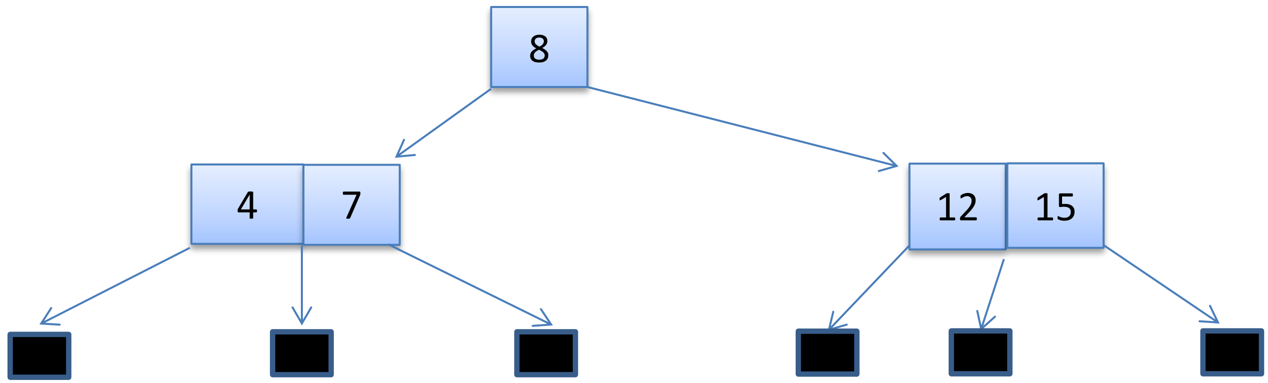
Key idea: When returning from a call to insert, return a boolean "grow."

Invariant:

- if grow is true, the height of the tree increased by 1
- if grow is false, the height of the tree stayed the same

How to Insert

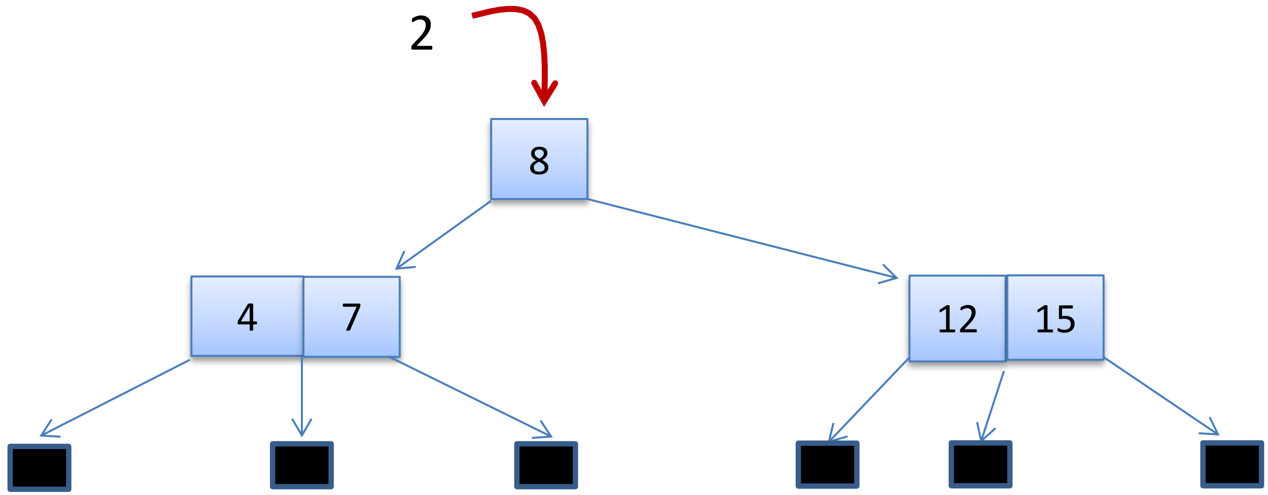
insert 2 into:



We are done!

How to Insert

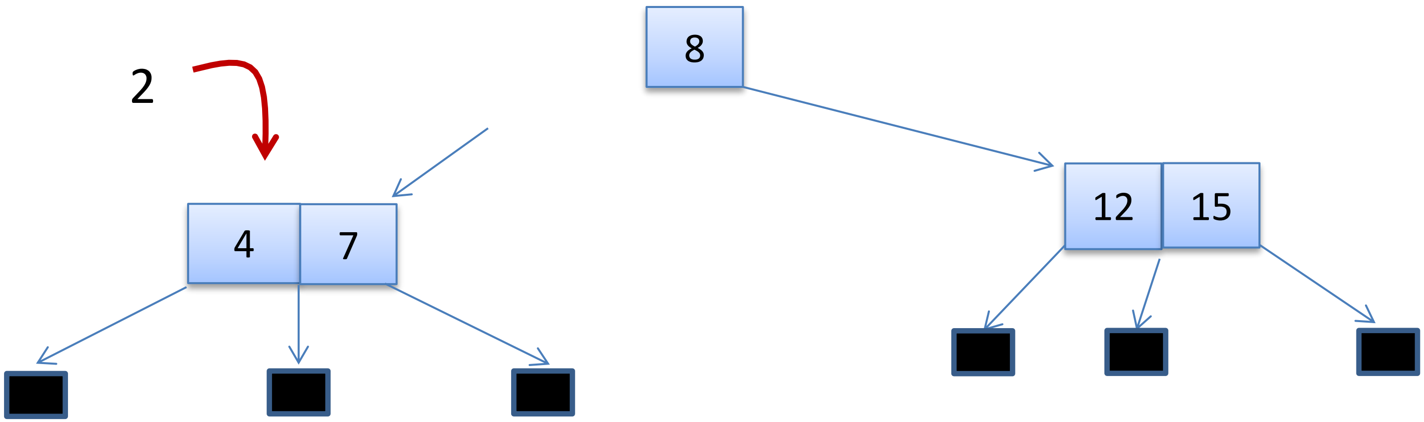
insert 2 into:



Compare 2 with the root

How to Insert

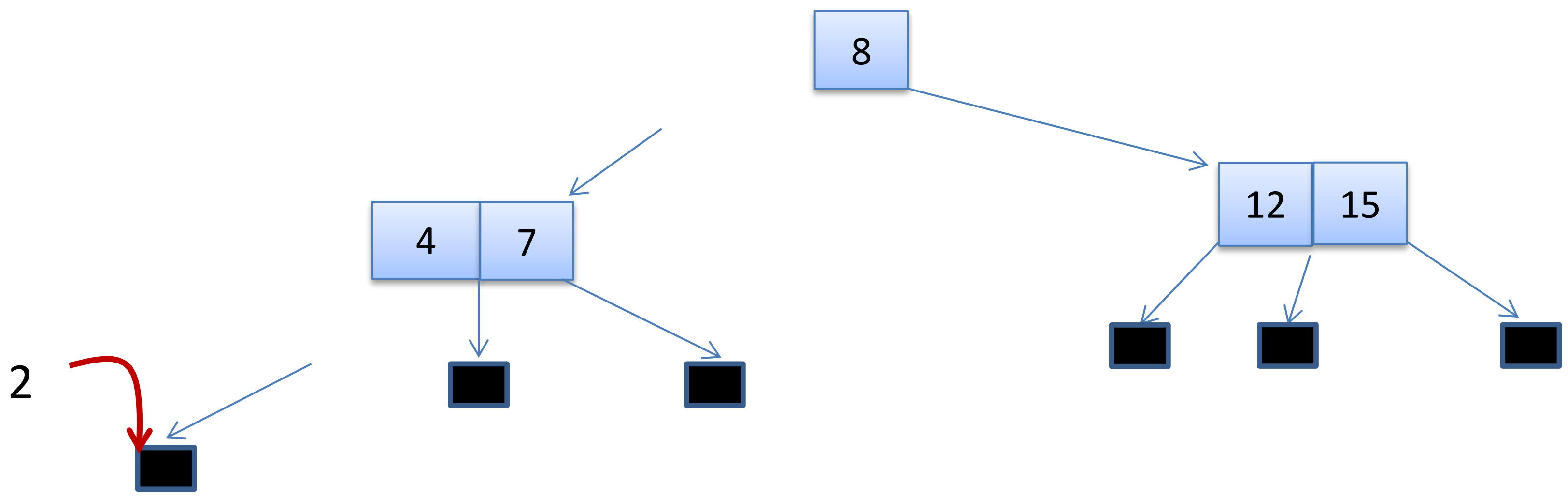
insert 2 into:



Recursively insert 2 into the 4-7 subtree

How to Insert

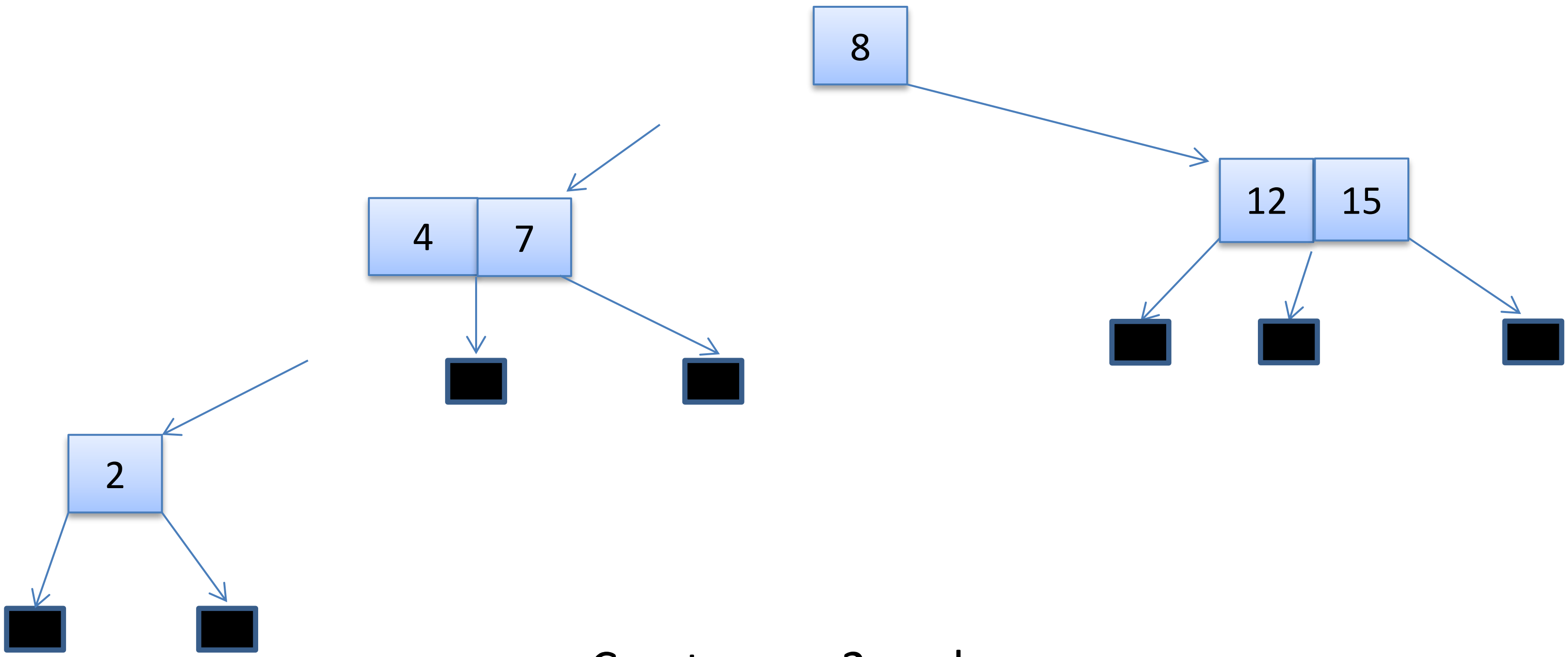
insert 2 into:



Recursively insert 2 into the Leaf

How to Insert

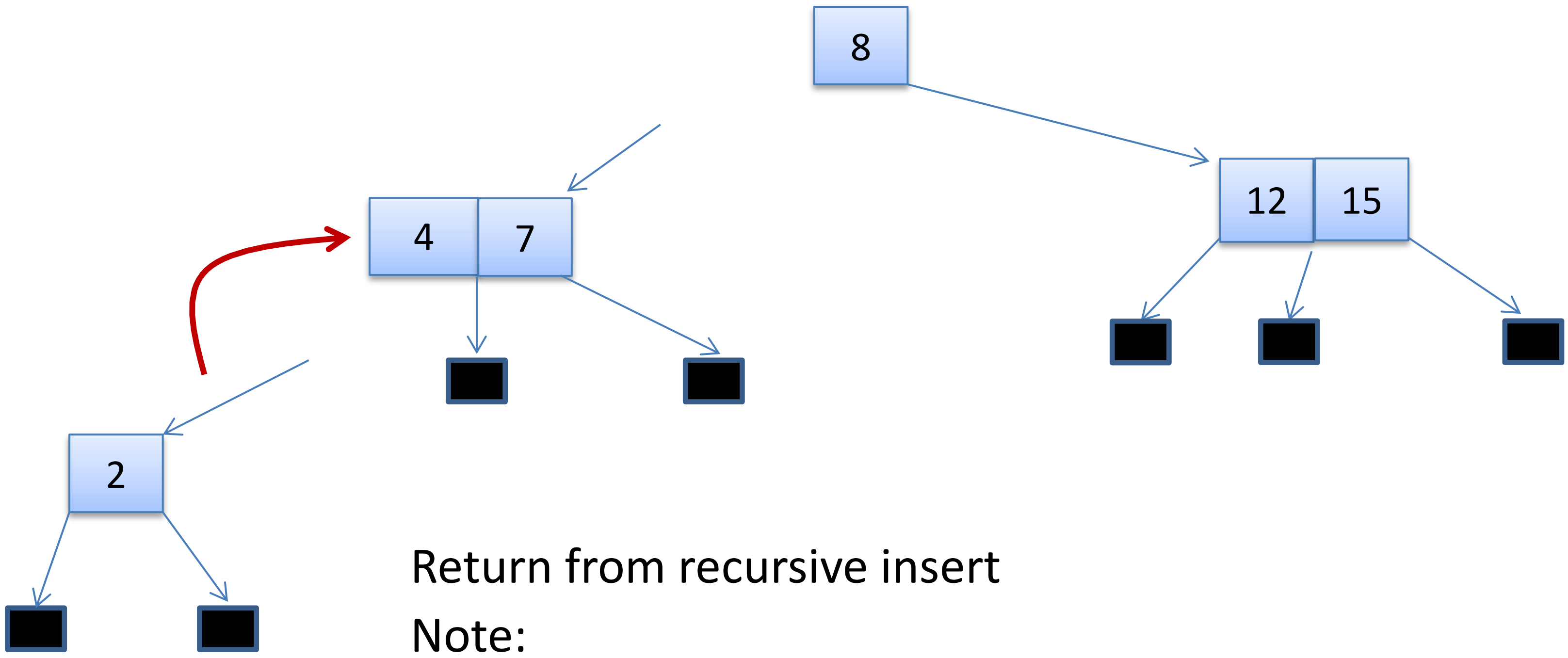
insert 2 into:



Create new 2-node

How to Insert

insert 2 into:



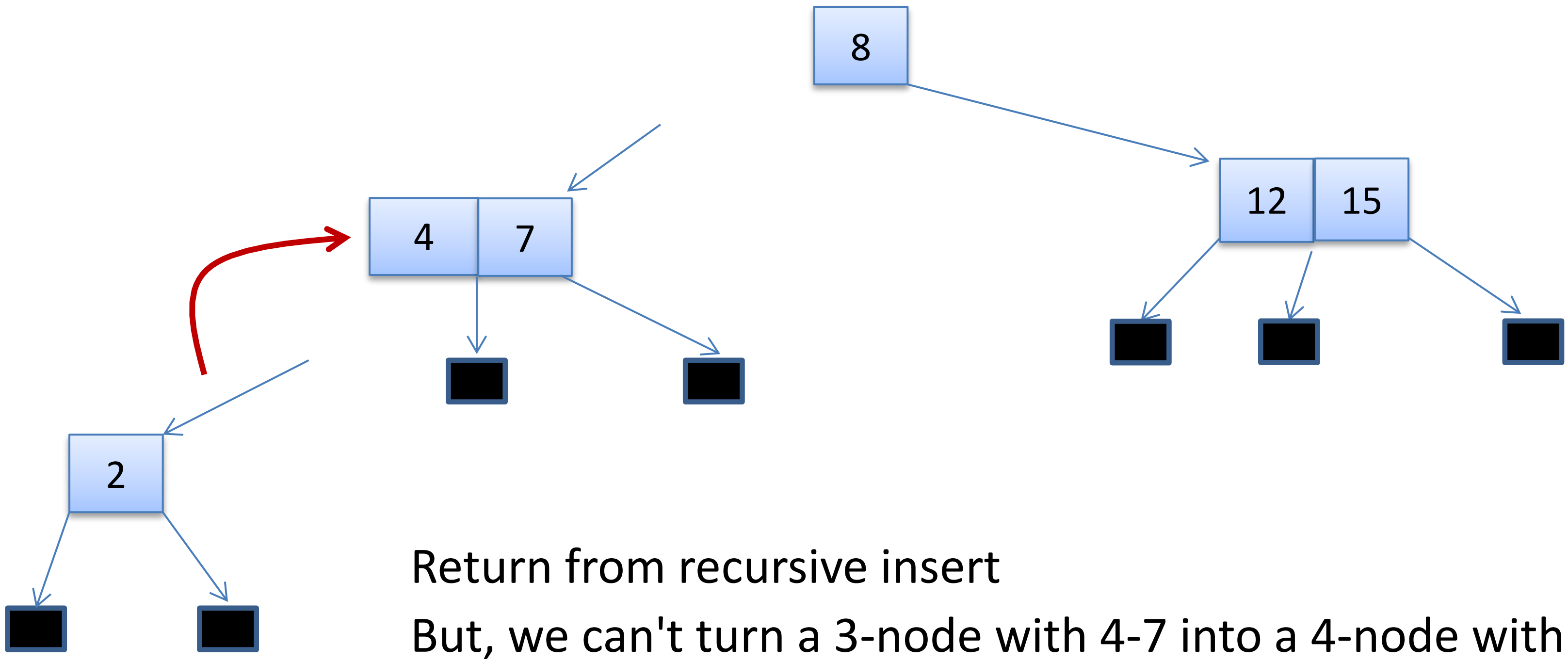
Return from recursive insert

Note:

- The height of the subtree has grown by 1
- It grew from height 0 (a leaf) to height 1 (tree with one node)
- If we include the new subtree in node 12 where the old subtree was then we will have children of uneven height.

How to Insert

insert 2 into:



Return from recursive insert

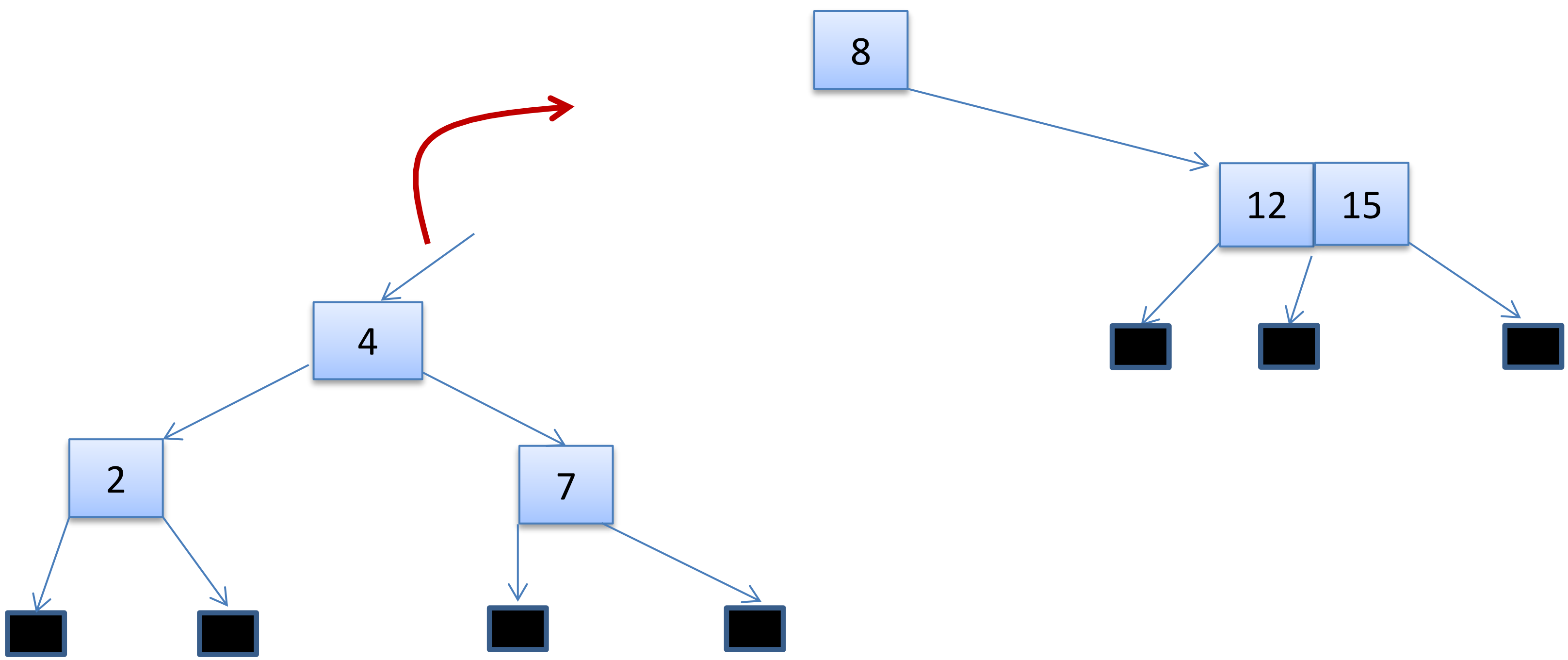
But, we can't turn a 3-node with 4-7 into a 4-node with 2-4-7!

<u>Solution:</u>

- turn the 3-node into a 2-node, with 2 2-node children

How to Insert

insert 2 into:



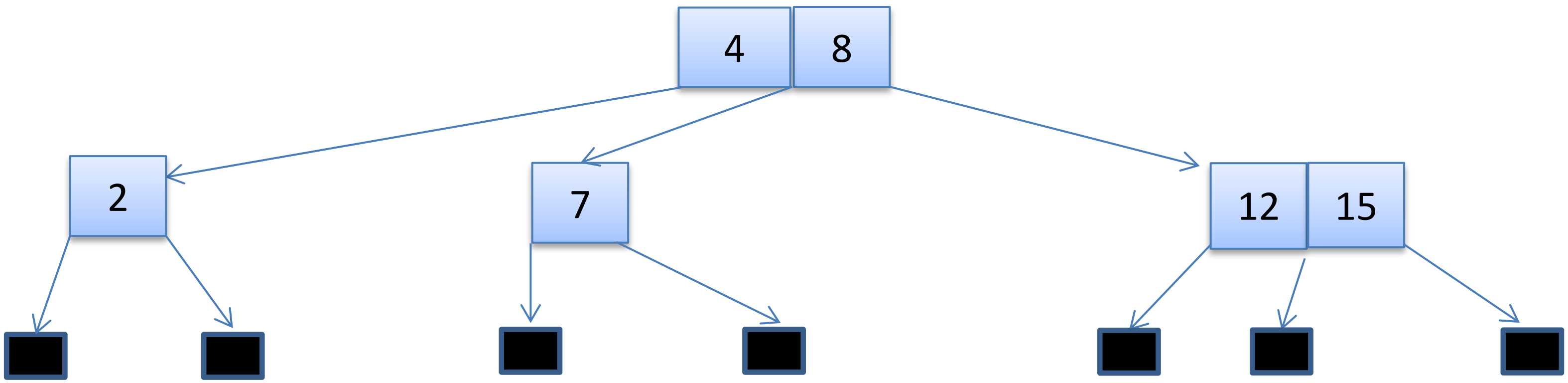
new subtree created. return from recursive call

<u>note:</u>

- this new subtree has grown by 1
- report that when returning from the recursive call

How to Insert

insert 2 into:



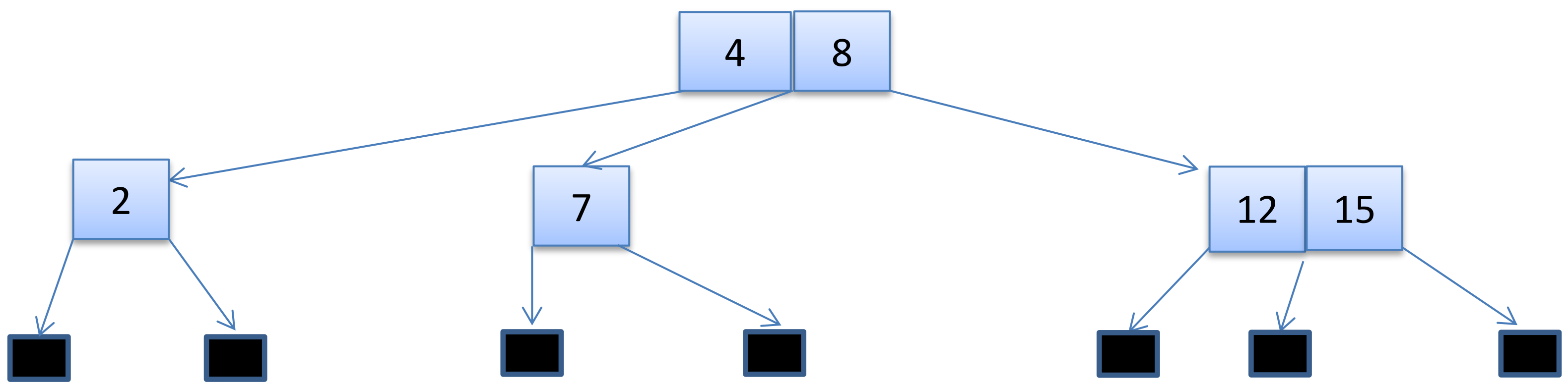
new subtree has grown by one, but we can include it in the root because the root is a 2-node.

all paths from the root to the leaves now have the same length.

DELETE

How to Delete

delete 12 in:

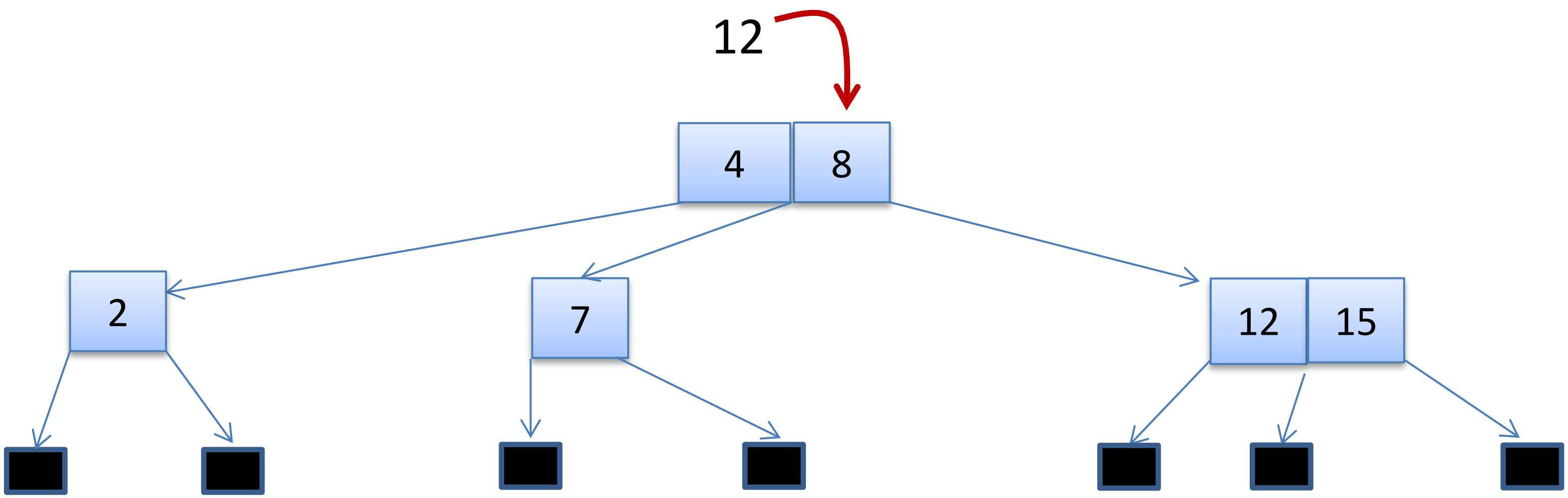


new subtree has grown by one, but we can include it in the root because the root is a 2-node.

all paths from the root to the leaves now have the same length.

How to Delete

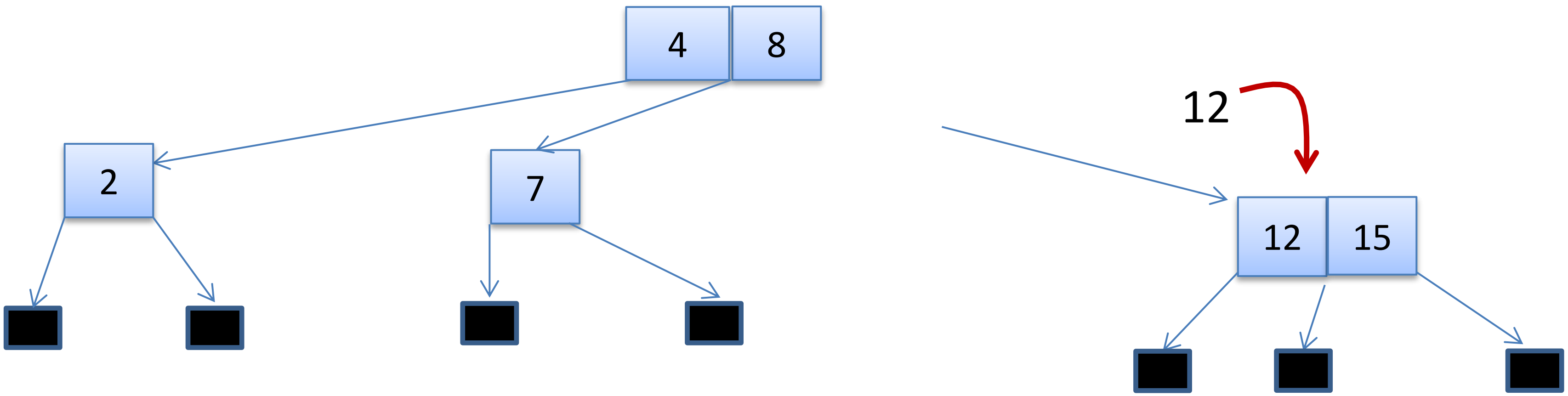
delete 12 in:



compare with root

How to Delete

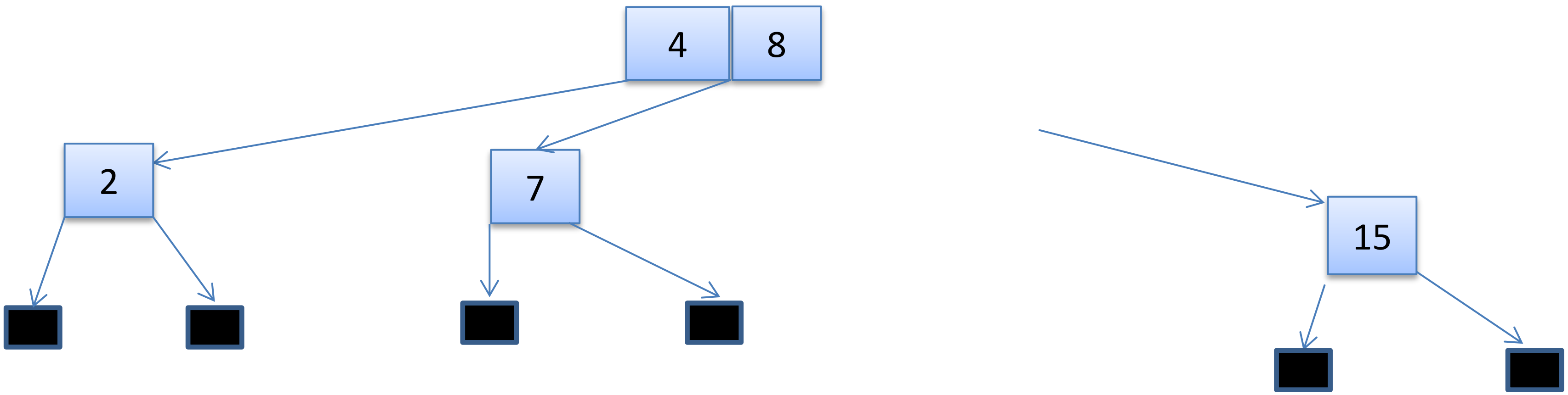
delete 12 in:



found 12 in terminal 3-node

How to Delete

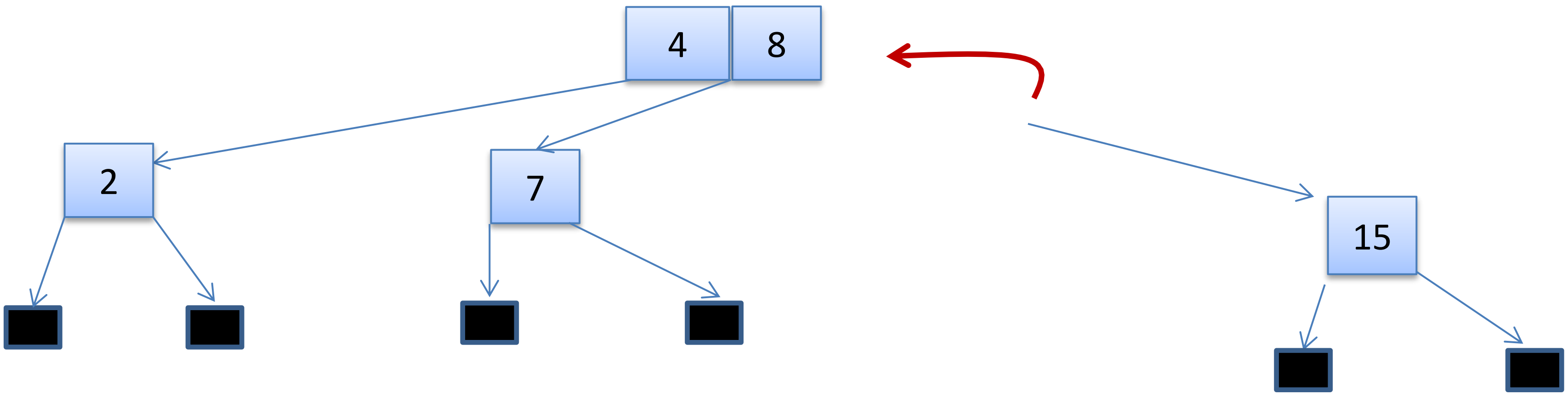
delete 12 in:



convert 3-node to 2-node

How to Delete

delete 12 in:

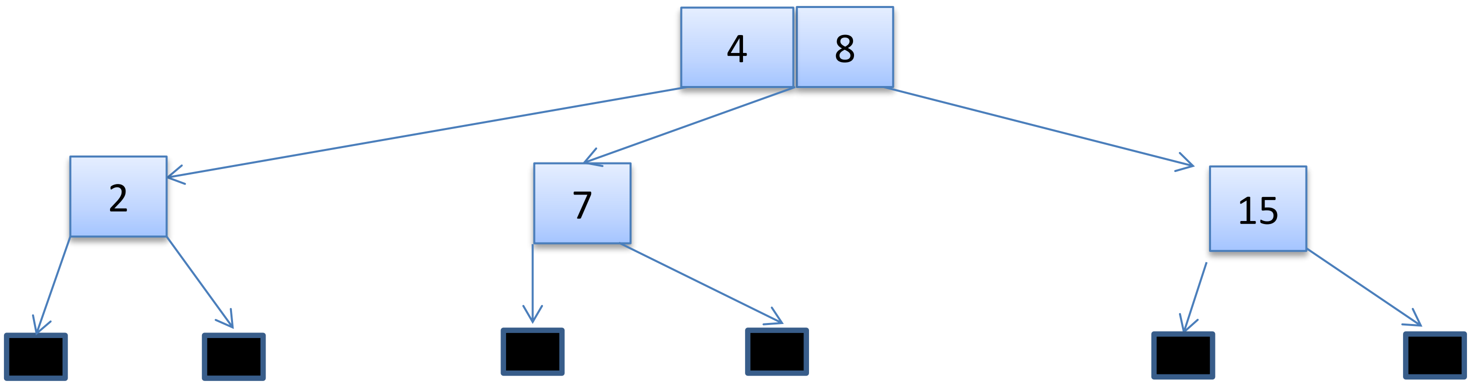


return from recursive call

- report that height of new subtree is the same height as old subtree

How to Delete

delete 12 in:



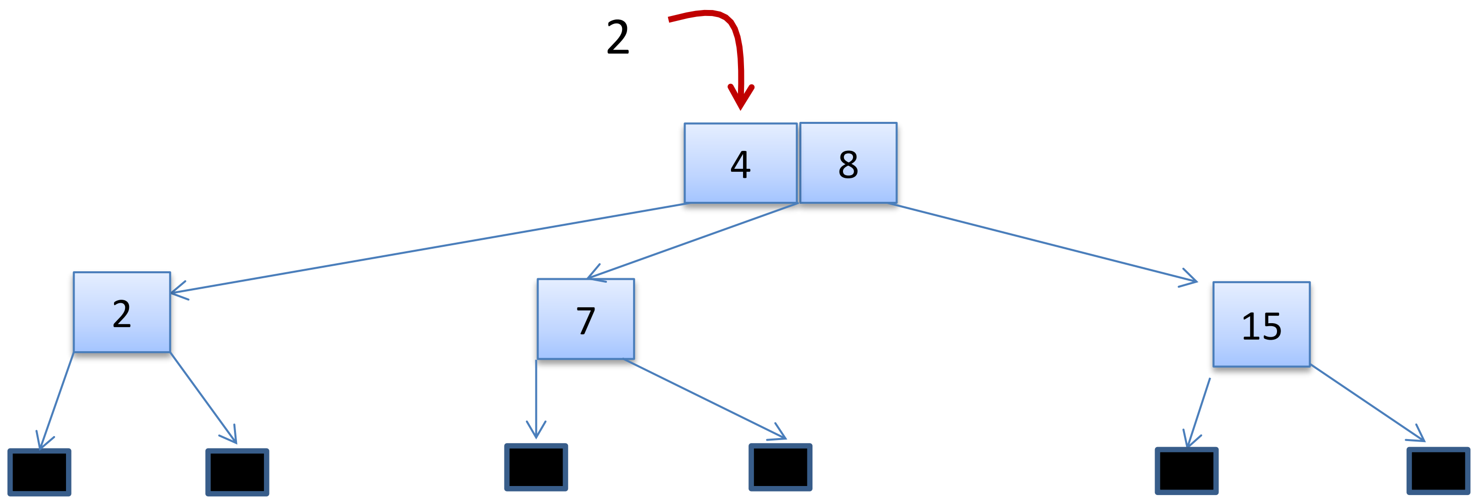
overall tree has 3 children of the same height

we are done

(if we weren't done, recursively return from delete reporting no change in height)

How to Delete

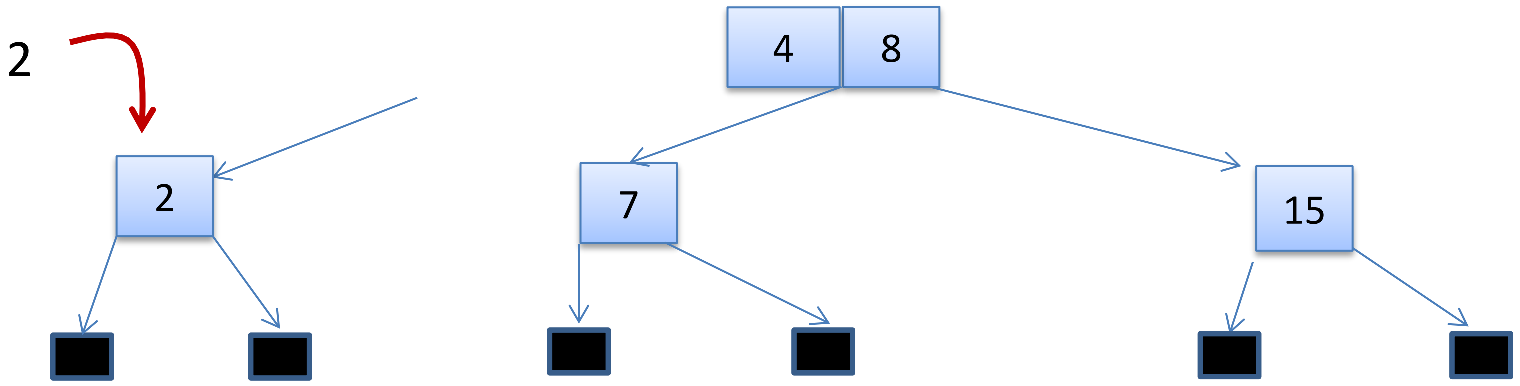
delete 2 in:



overall tree has 3 children of the same height
we are done

How to Delete

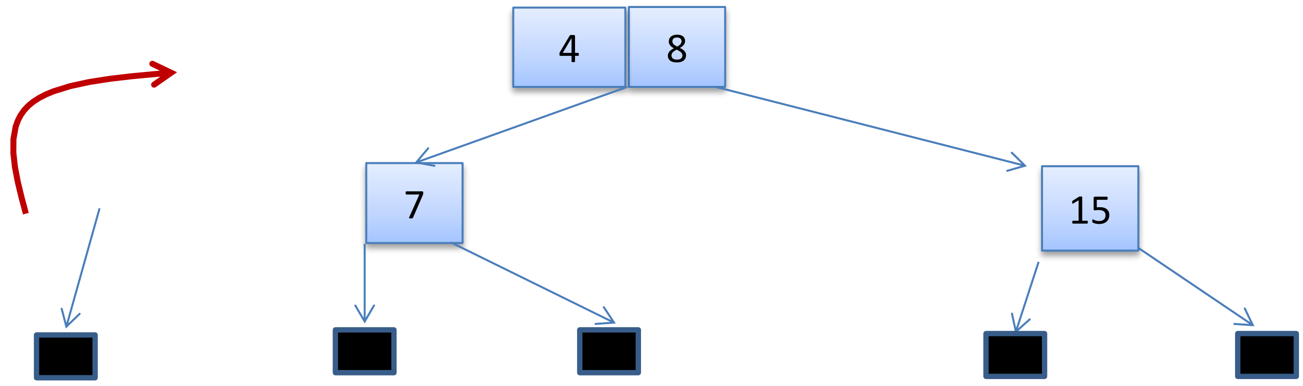
delete 2 in:



found 2 in terminal 2-node

How to Delete

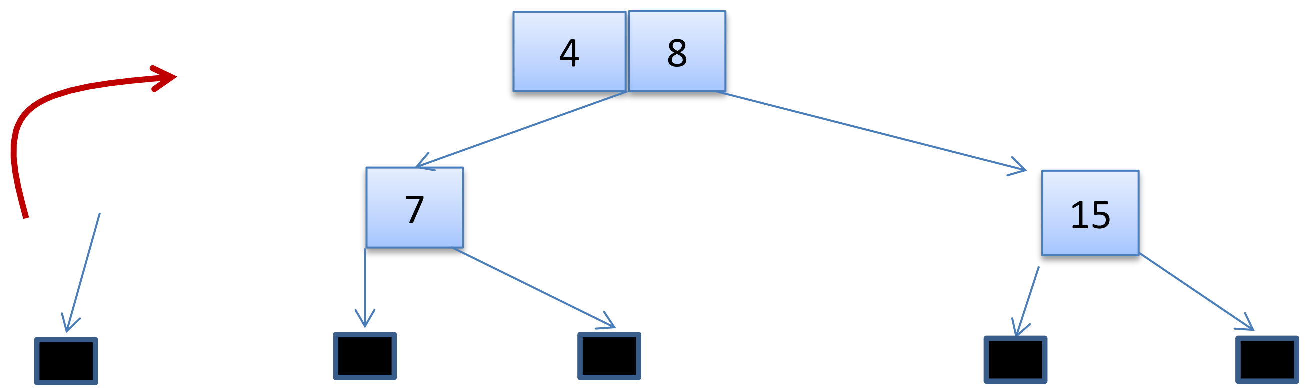
delete 2 in:



Delete element, creating shorter tree

How to Delete

delete 2 in:

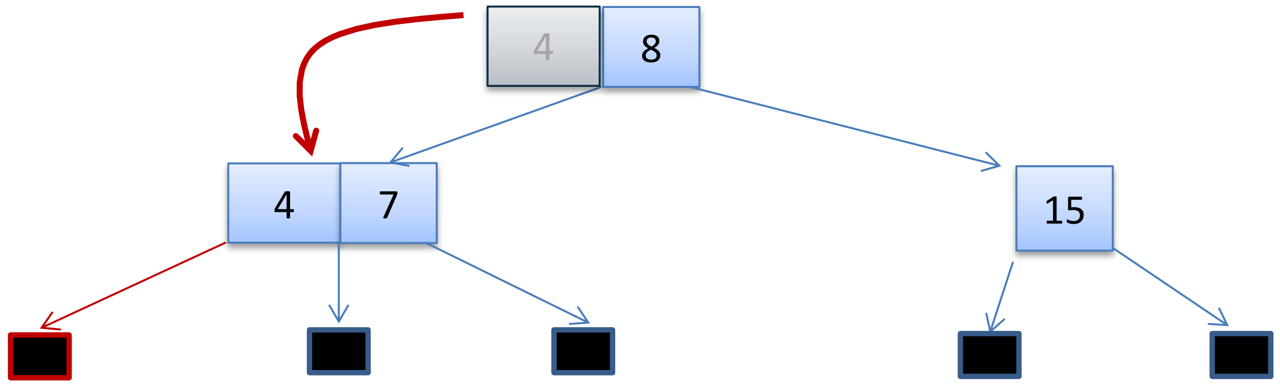


return from recursive call to delete

- report current tree is 1 shorter than height of original tree
- parent is 3-node
- has 2-node as another child

How to Delete

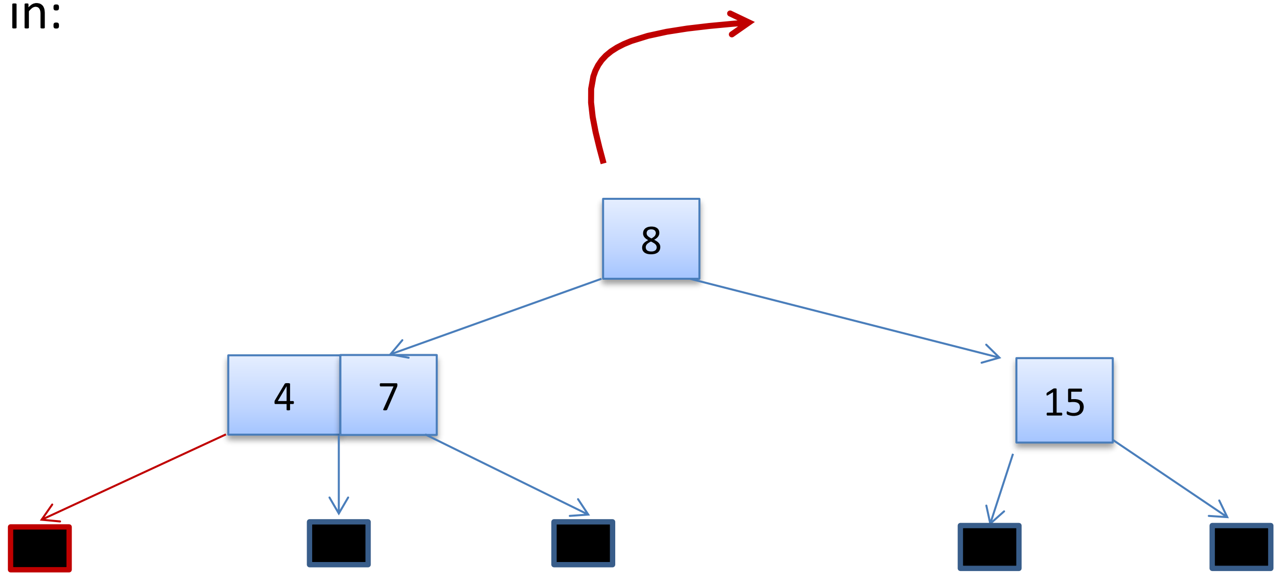
delete 2 in:



rotate element of 3-node from parent to sibling
attach node to sibling

How to Delete

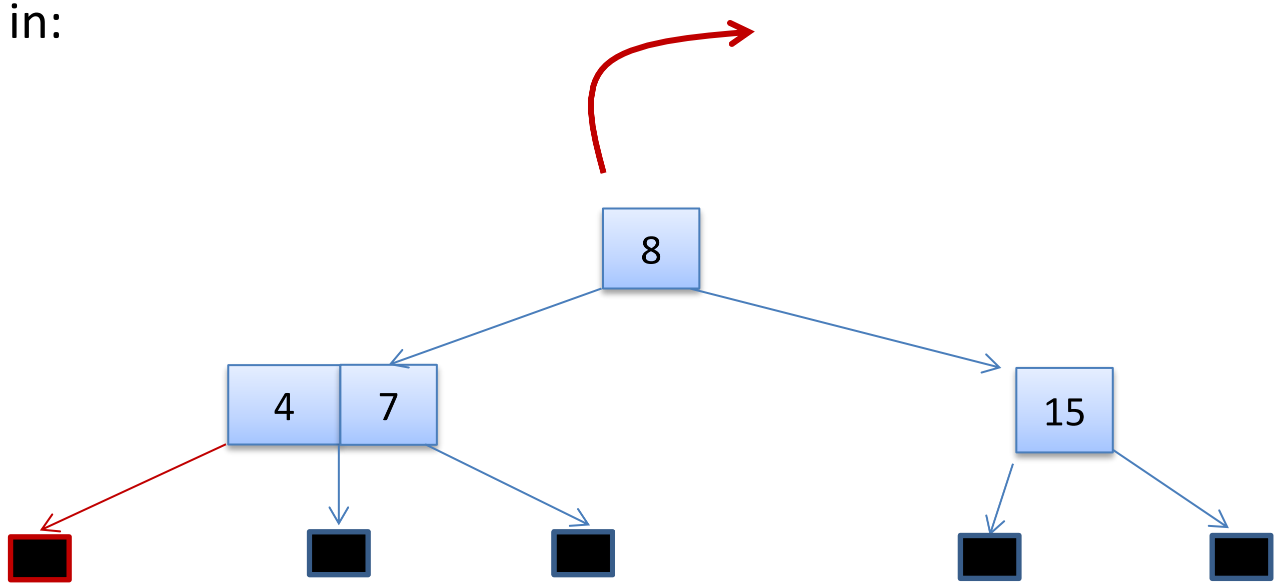
delete 2 in:



return
(done in this case)

How to Delete

delete 2 in:

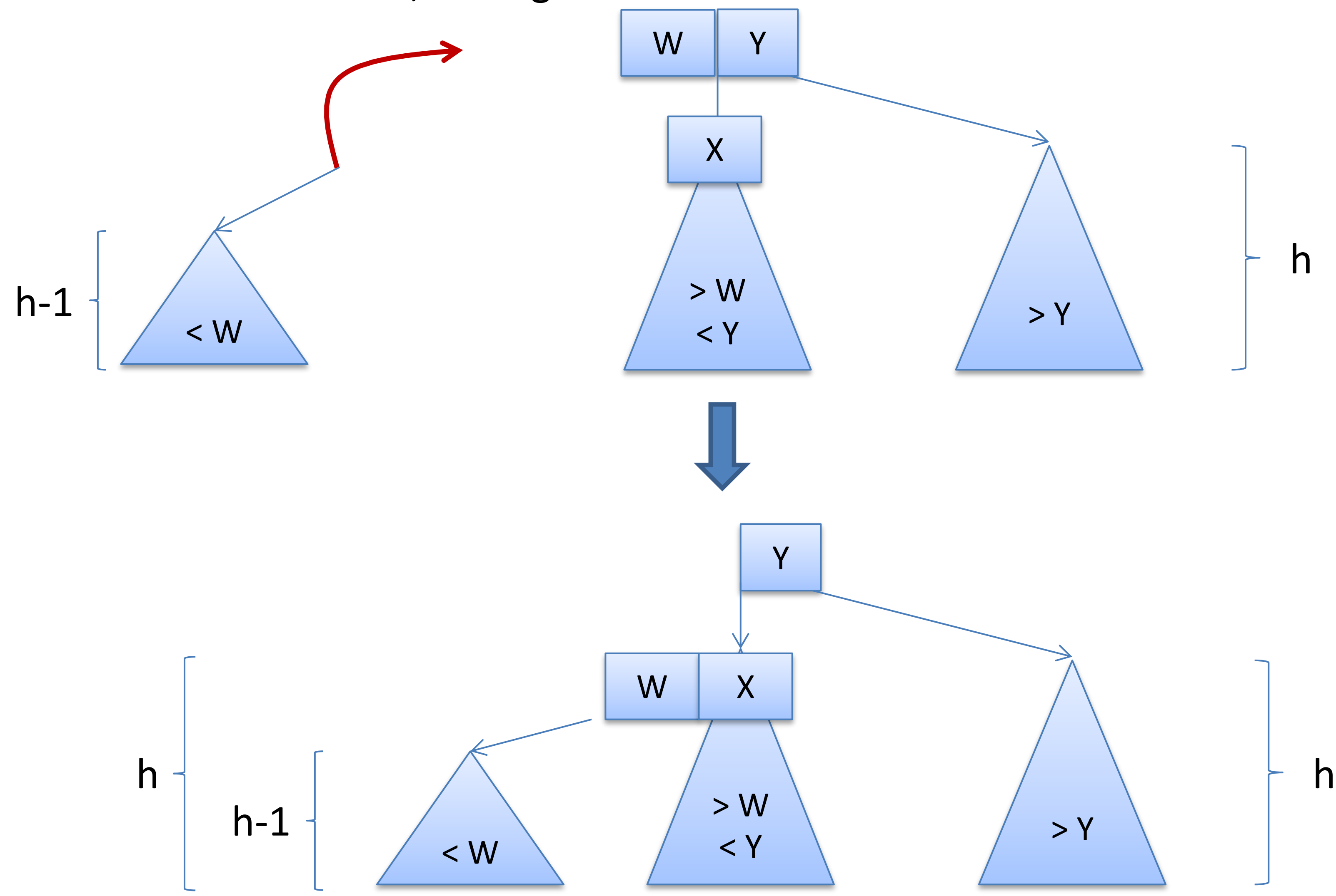


return
(done in this case)

How to Delete

More generally, when returning with a tree of decreased height.

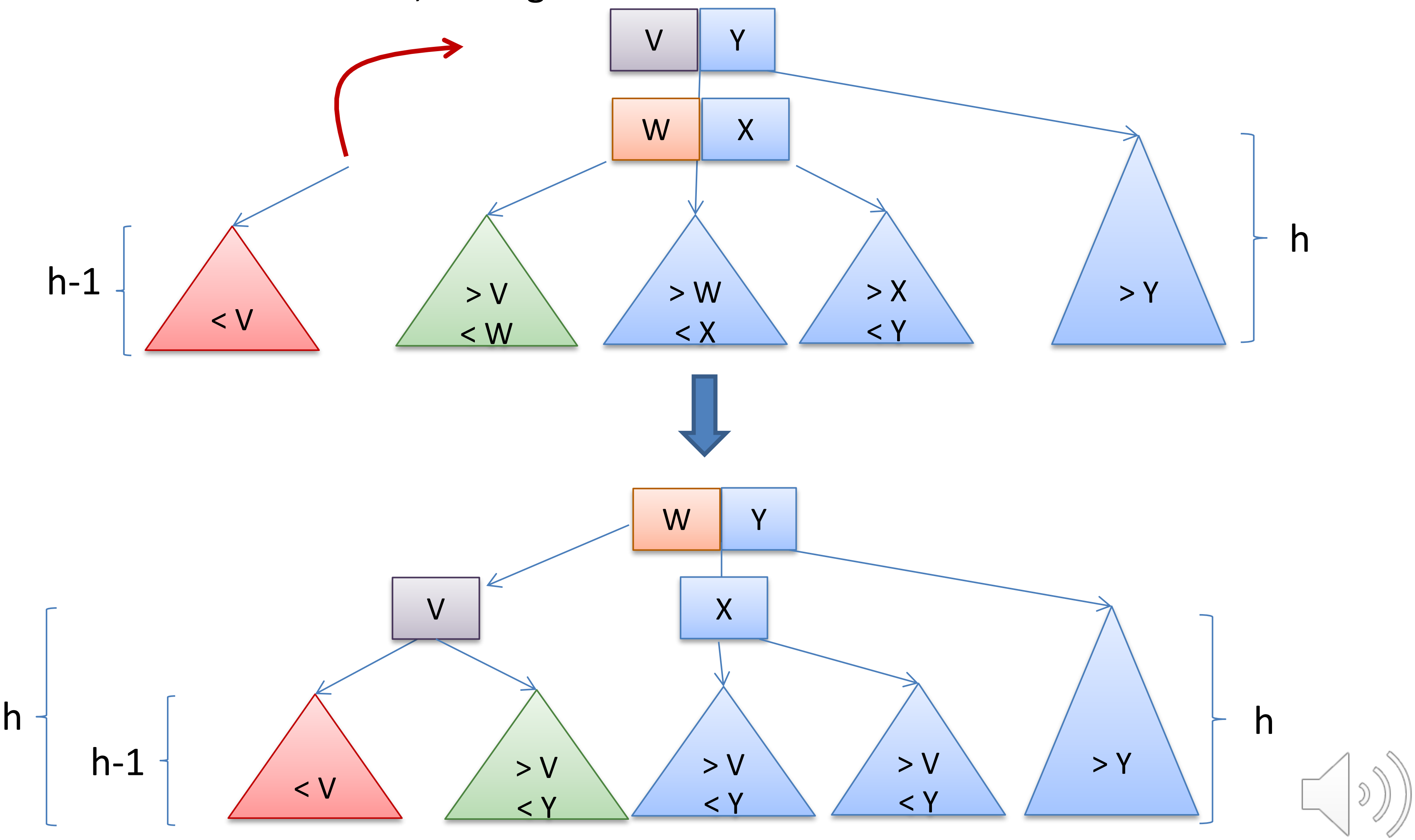
Case: Parent is 3-node; Sibling is 2-node.



How to Delete

More generally, when returning with a tree of decreased height.

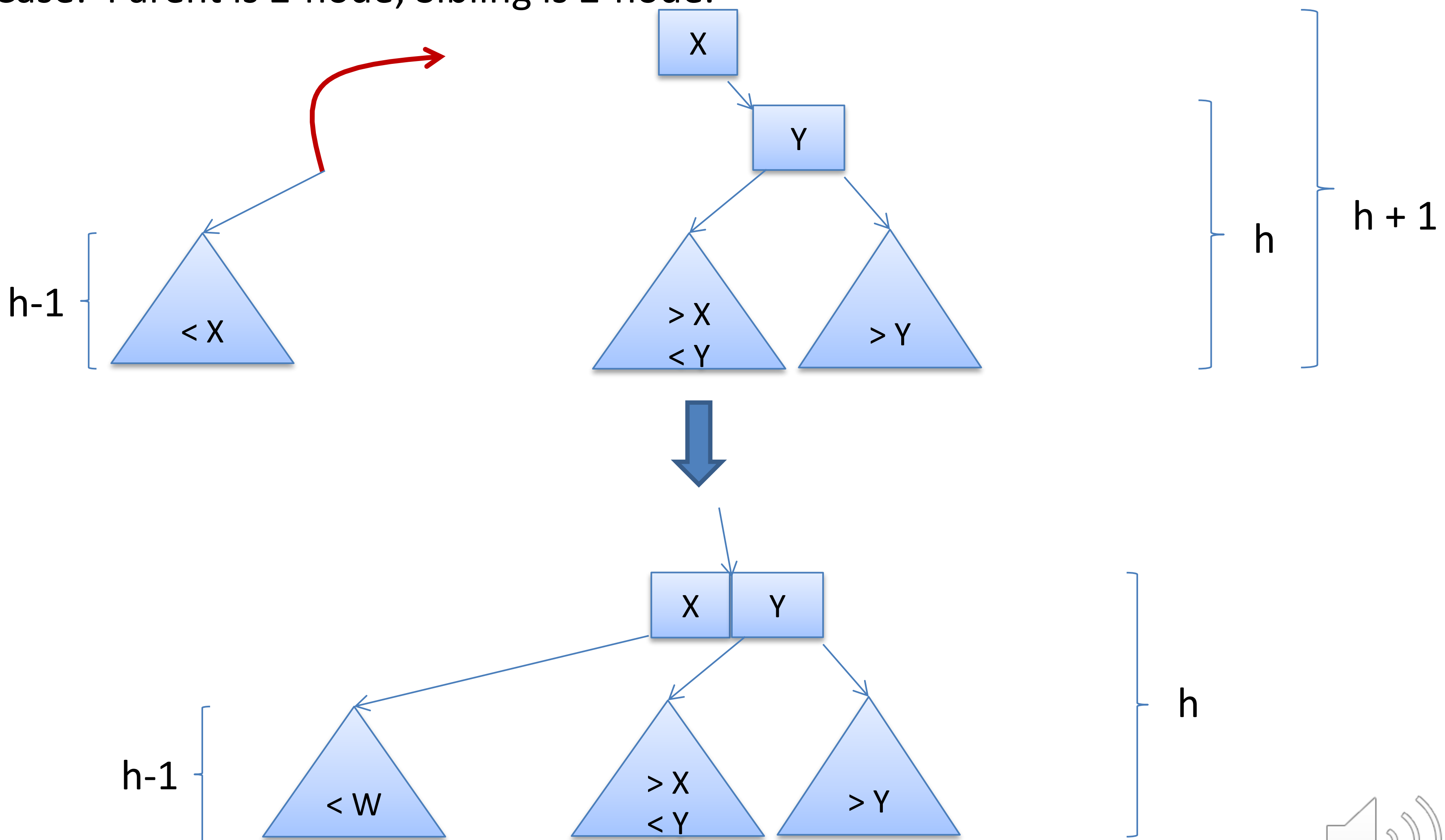
Case: Parent is 3-node; Sibling is 3-node.



How to Delete

More generally, when returning with a tree of decreased height.

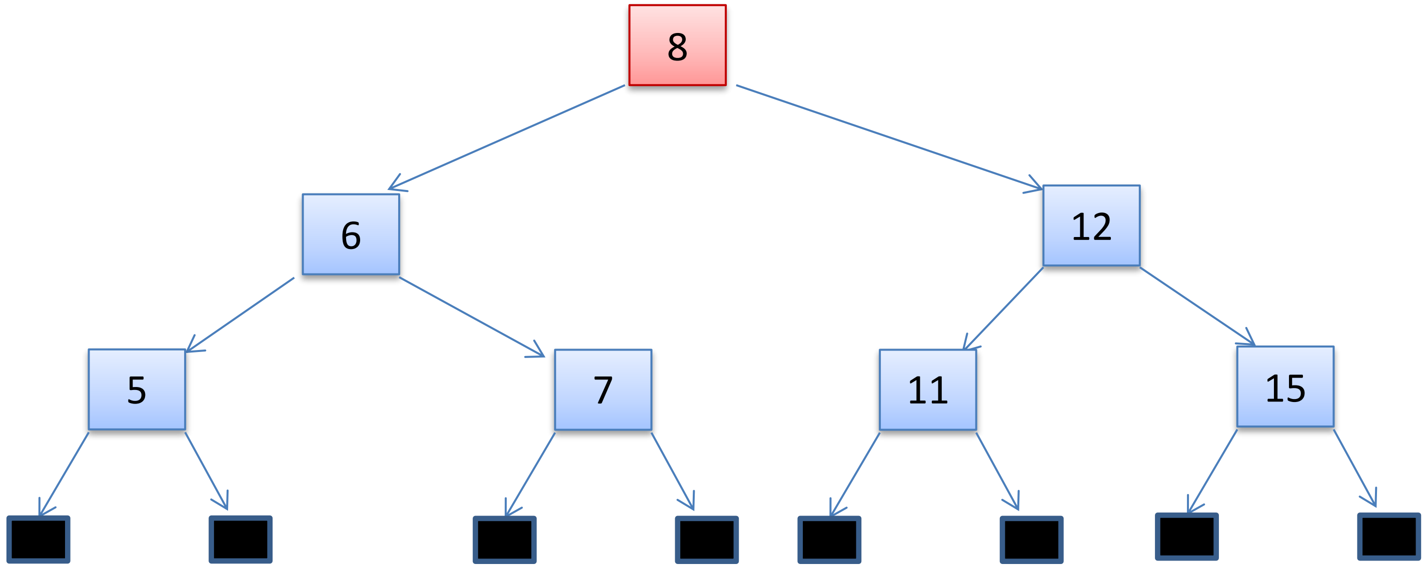
Case: Parent is 2-node; Sibling is 2-node.



height of new tree is one less that original

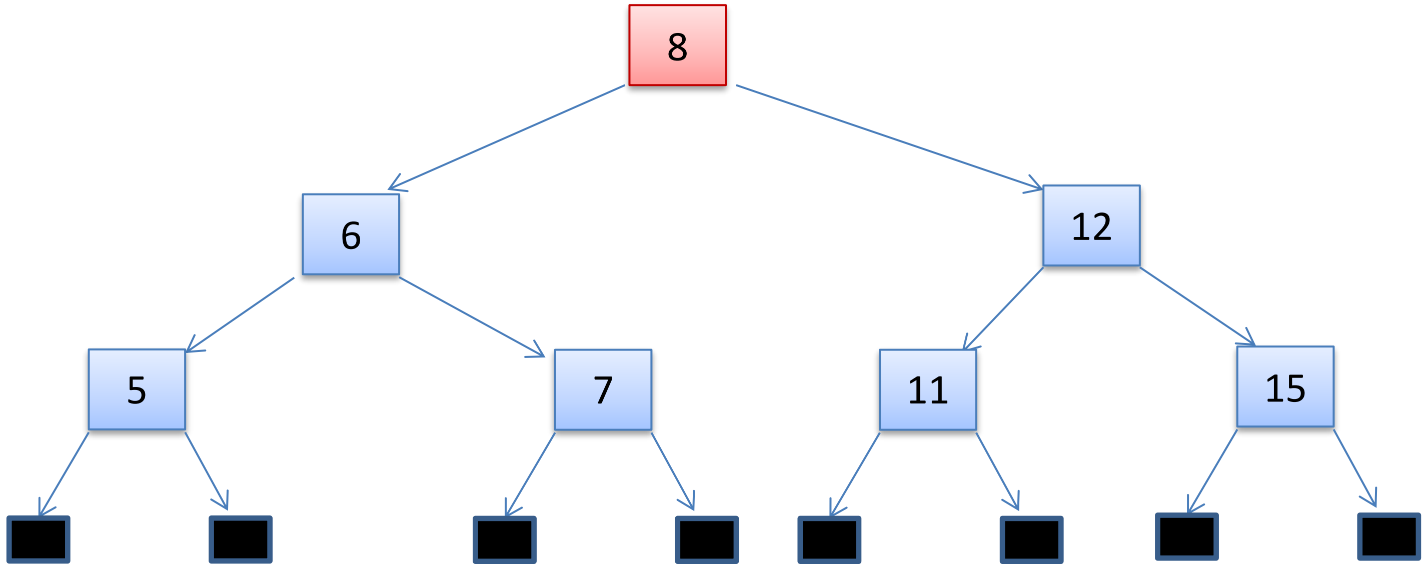
How to Delete Non-terminal Nodes?

Delete 8 in:



How to Delete Non-terminal Nodes?

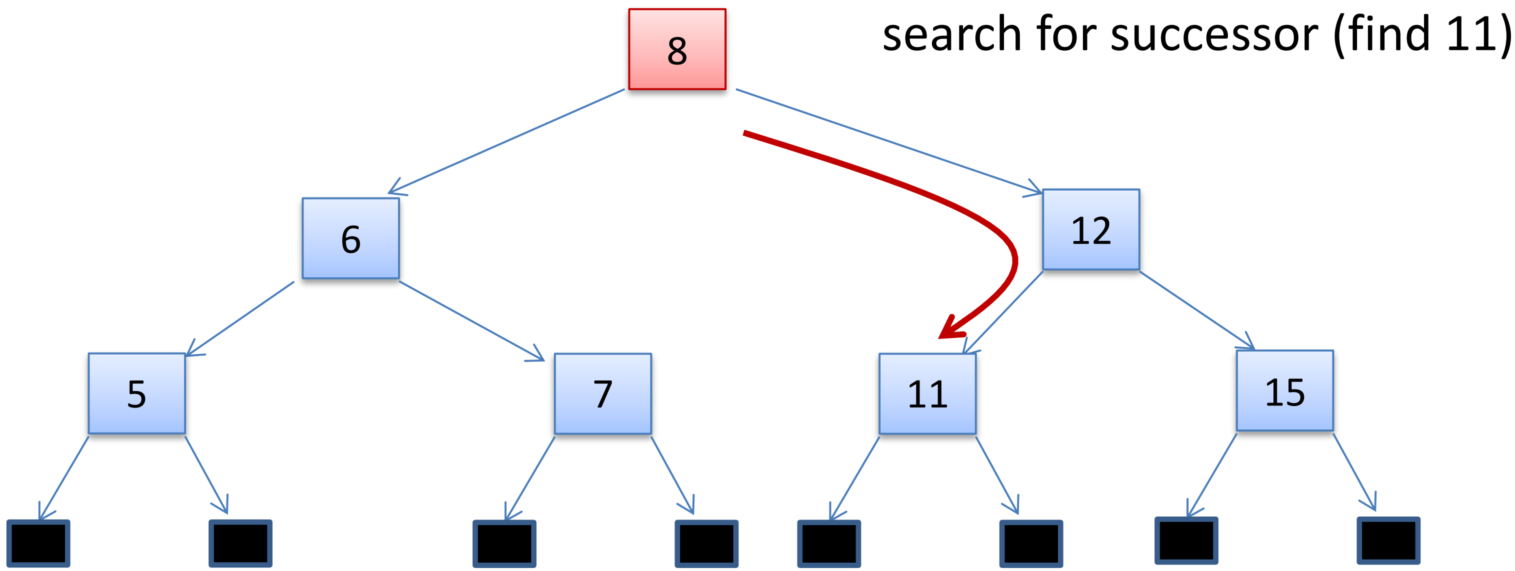
Delete 8 in:



1. Find the node's immediate successor S, which will be in a terminal node.
2. Replace current node's value with S
3. Continue the algorithm, deleting the occurrence of S in the subtree

How to Delete Non-terminal Nodes?

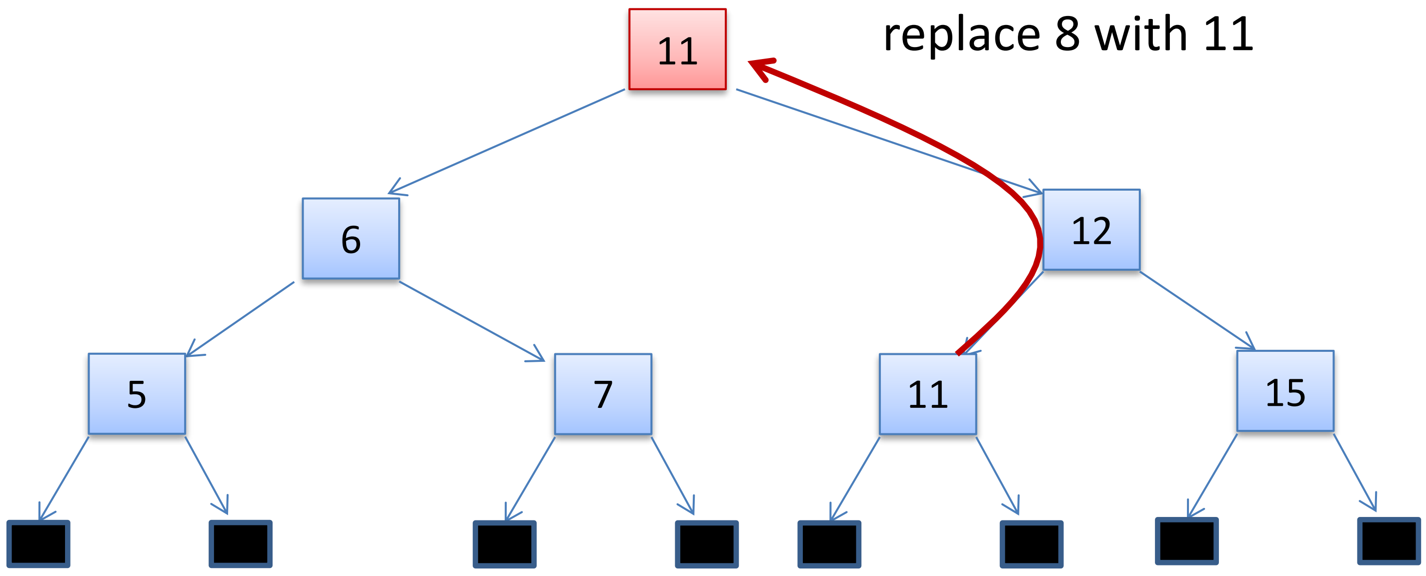
Delete 8 in:



1. Find the node's immediate successor S, which will be in a terminal node.
2. Replace current node's value with S
3. Continue the algorithm, deleting the occurrence of S in the subtree

How to Delete Non-terminal Nodes?

Delete 8 in:

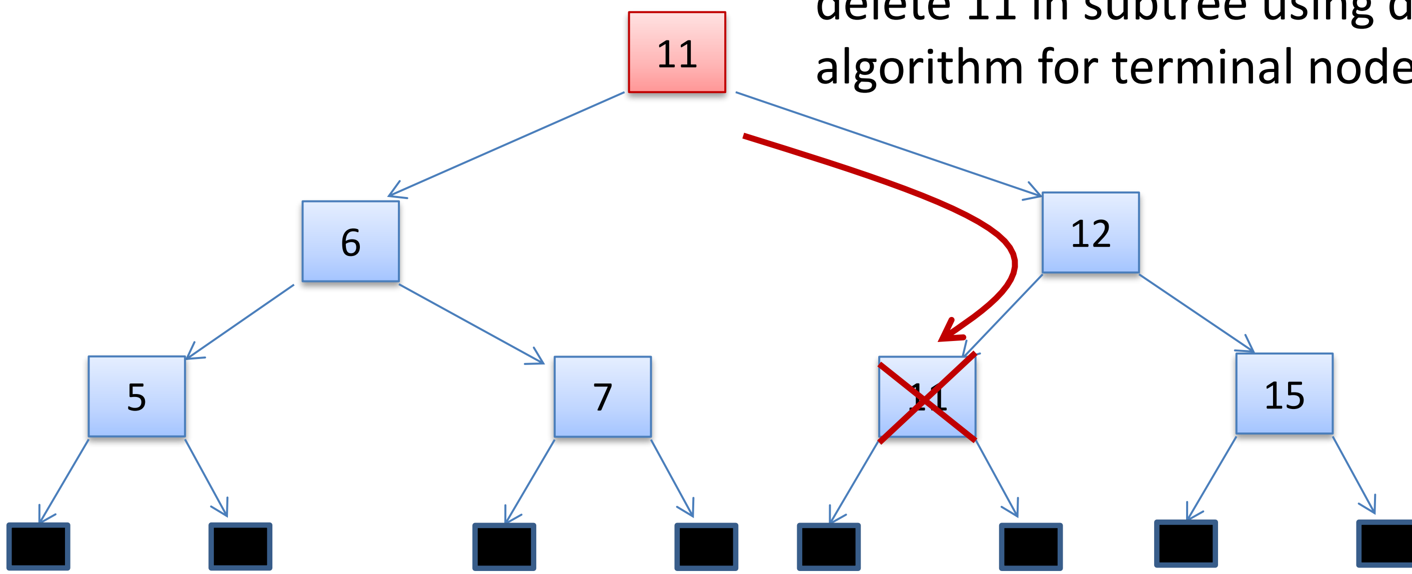


1. Find the node's immediate successor S, which will be in a terminal node.
2. Replace current node's value with S
3. Continue the algorithm, deleting the occurrence of S in the subtree

How to Delete Non-terminal Nodes?

Delete 8 in:

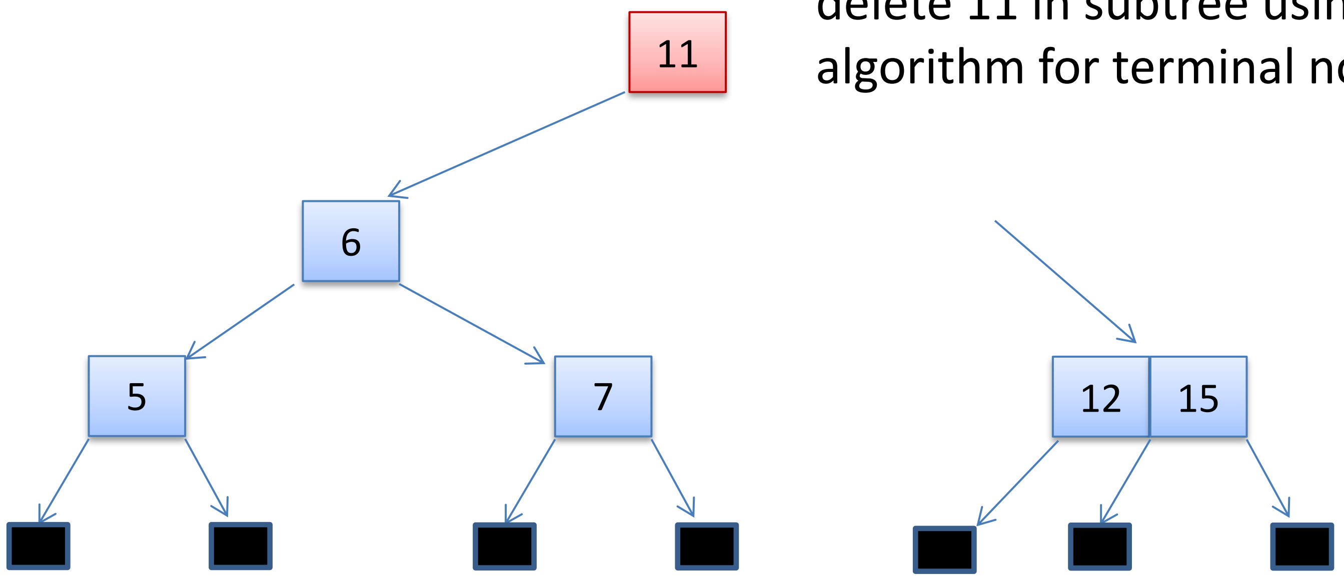
delete 11 in subtree using deletion algorithm for terminal nodes



1. Find the node's immediate successor S, which will be in a terminal node.
2. Replace current node's value with S
3. Continue the algorithm, deleting the occurrence of S in the subtree

How to Delete Non-terminal Nodes?

Delete 8 in:



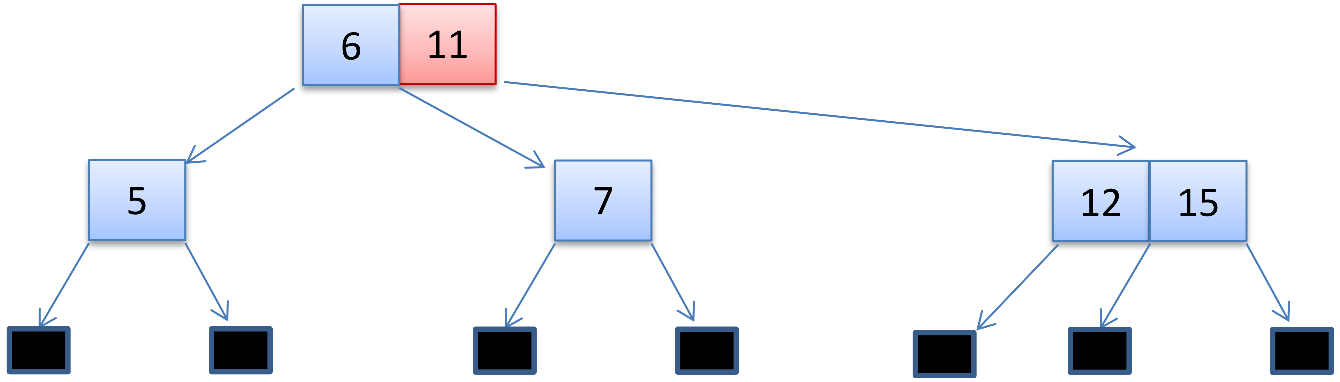
delete 11 in subtree using deletion algorithm for terminal nodes

1. Find the node's immediate successor S, which will be in a terminal node.
2. Replace current node's value with S
3. Continue the algorithm, deleting the occurrence of S in the subtree

How to Delete Non-terminal Nodes?

Delete 8 in:

delete 11 in subtree using deletion algorithm for terminal nodes



1. Find the node's immediate successor S, which will be in a terminal node.
2. Replace current node's value with S
3. Continue the algorithm, deleting the occurrence of S in the subtree

OCAML IMPLEMENTATION

OCaml 2-3 Trees

```
type pair = key * value

type dict =
  | Leaf
  | Two of dict * pair * dict
  | Three of dict * pair * dict * pair * dict
```

Valid 2-3 trees must be:

- in order
- balanced (equal height subtrees)

You will write an invariant function to check that the trees produced by your functions are valid 2-3 trees.

This is going to help you debug your routines a lot.

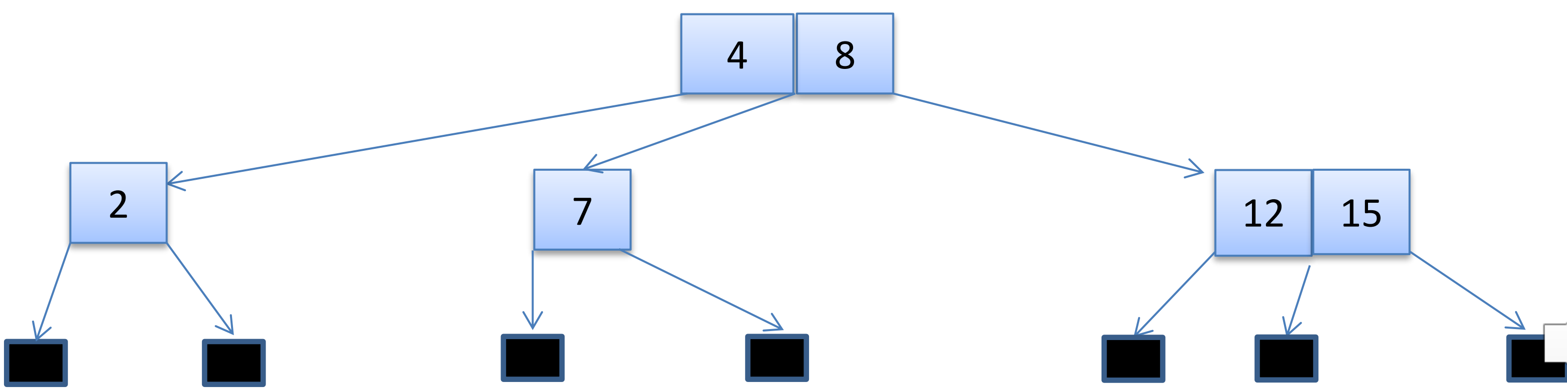
The OCaml Insert Function

insert_to_tree : dict -> key -> value -> bool * dict

Key Property:

If d is a valid 2-3 tree and insert_to_tree d k v = (grow, d') then

- d' is a valid 2-3 tree
- d' contains all of the elements of d as well as (k,v)
- if grow then height(d') = height(d) + 1,
- else height(d') = height(d)



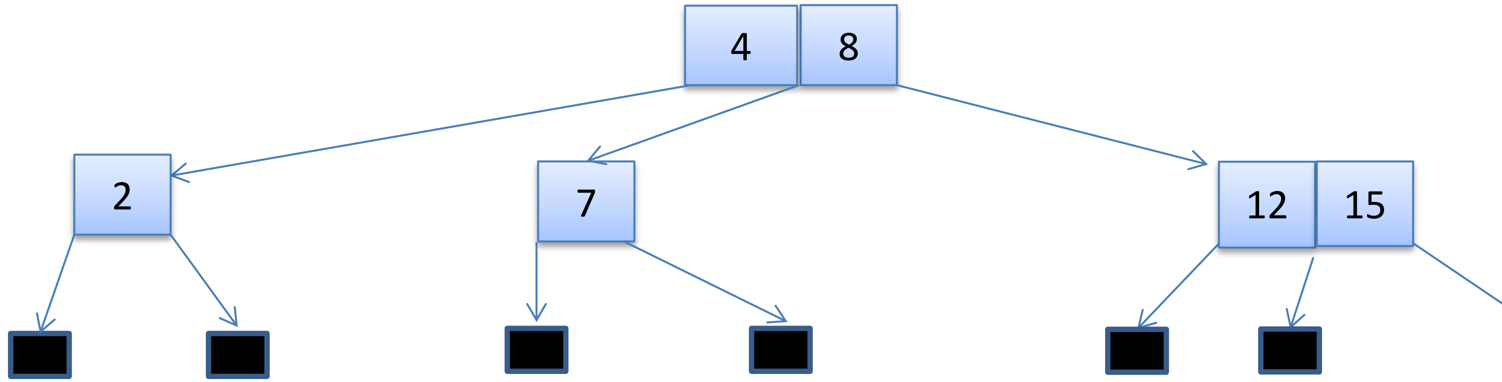
The OCaml Remove Function

remove_from_tree : dict -> key -> bool * dict

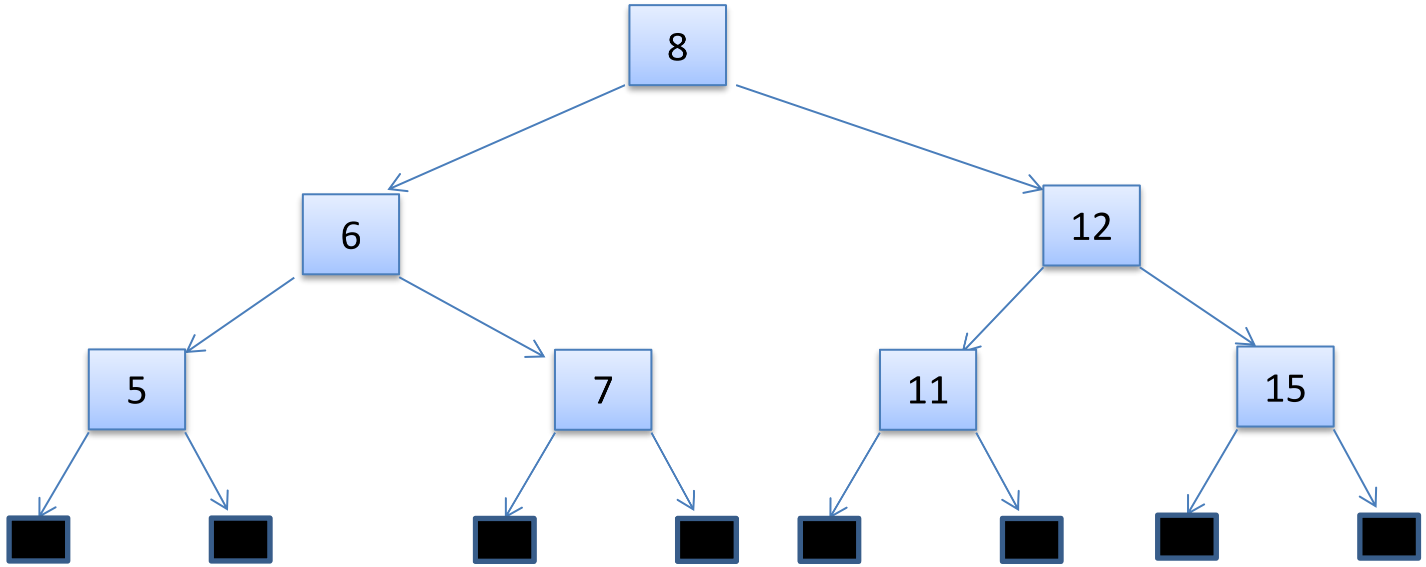
Key Property:

If d is a valid 2-3 tree and remove_from_tree d k = (shrink, d') then

- d' is a valid 2-3 tree
- d' contains all of the elements of d except the one for k
- if shrink then height(d') = height(d) - 1,
- else height(d') = height(d)



A Possible Implementation Strategy



1. Implement the 2-3 invariant to help you debug
2. Implement insert
3. Implement remove for terminal nodes
4. Implement remove for internal nodes