2-4 Trees

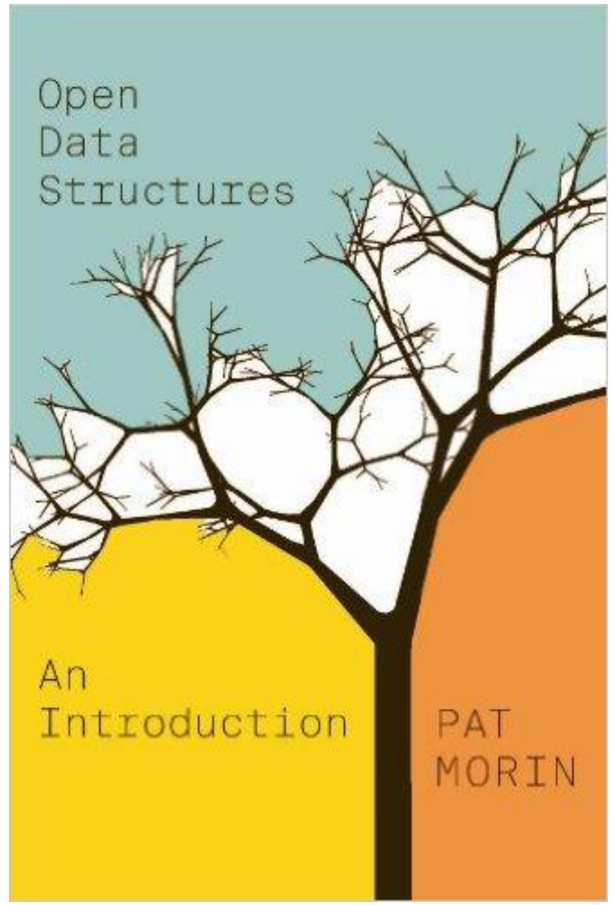
COMP 2402

Abstract Data Types & Algorithms

Readings

2-4 Trees

- Chapter 9.1



Multi-Way Search Trees

Recall a Binary Search Tree (**BST**)

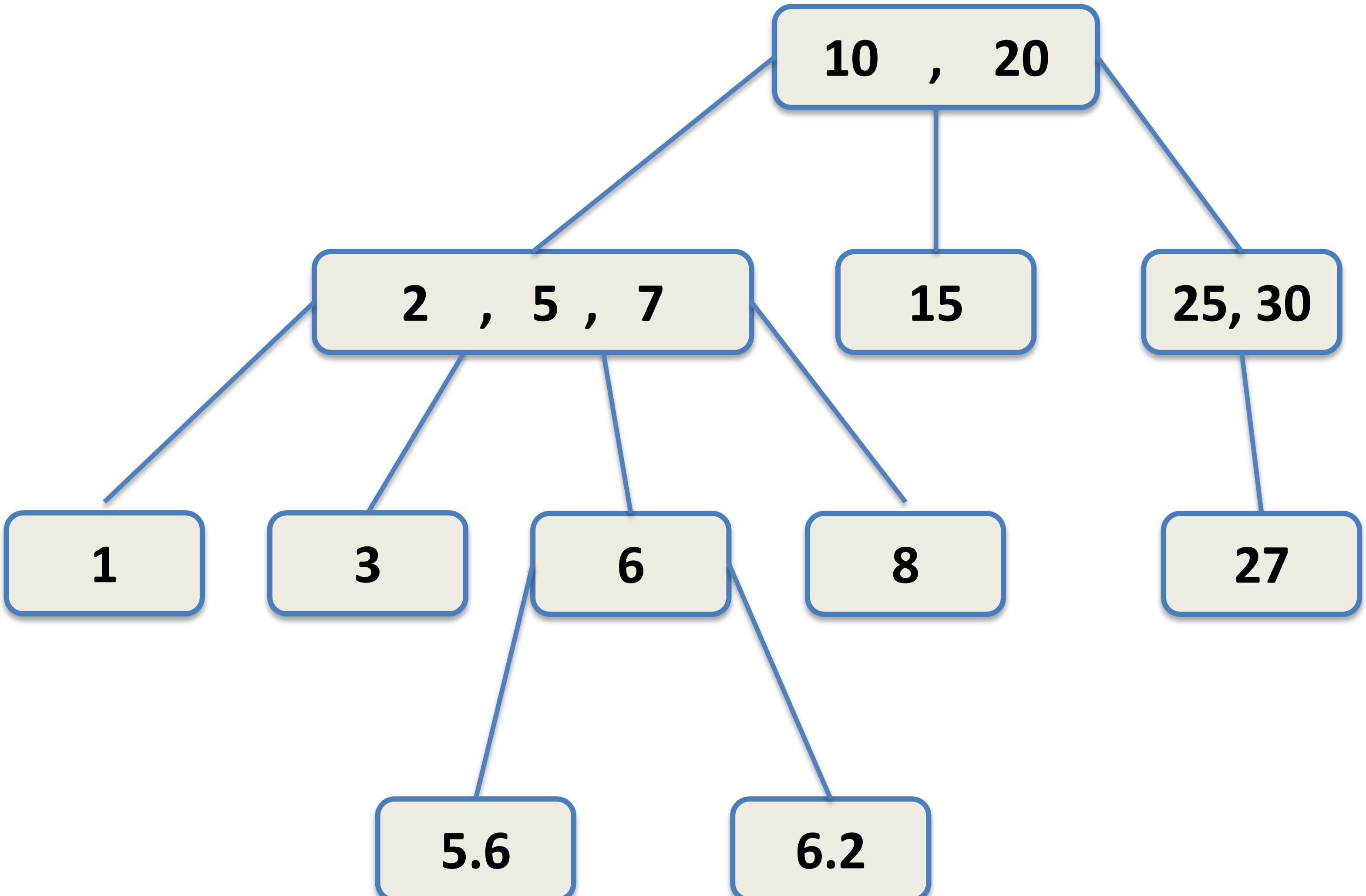
- Each node stores one value (or key)
- Each node has 0, 1 or 2 children
  - value of a node is greater than the value of every node in its left subtree
  - Value of a node is less than the value of every node in its right subtree

Multi-Way Search Trees

A multi-way search tree is a **generalization** of a binary search tree where nodes can hold more than one value (key) and have more than two children.

- If a node has d values $v_1, v_2, \ldots, v_d$ then it can have up to $d + 1$ children $c_1, c_2 \ldots., c_{d+1}$
- Let $v_0 = -\infty, v_{d+1} = \infty$.
  The value of all nodes in the subtree rooted at c_i must satisfy
  $$v_{i-1} < value(c_i) < v_i$$
- External nodes are just placeholders.

Multi-Way Search Trees



2-4 Trees

A **2-4 Tree** is a multi-way search tree that satisfies the following properties

Property 9.1 (**height**) All leaves have the same depth

Property 9.2 (**degree**) All internal nodes have 2,3 or 4 children.

2-4 Trees

Exercise: Prove that a 2-4 tree with n leaves has height at most $\log_2 n$.

Question: How do we maintain the two properties when adding/removing elements from the tree?

2-4 Trees

Adding a value (key) to a 2-4 tree.

Just as with a BST you search for the element you want to add and then add it at the bottom of the tree where it would have been if it was already there.

If the node has 1 or 2 values in it then this is easy.

If the node already has 3 values in it this is an **overflow**.

2-4 Trees

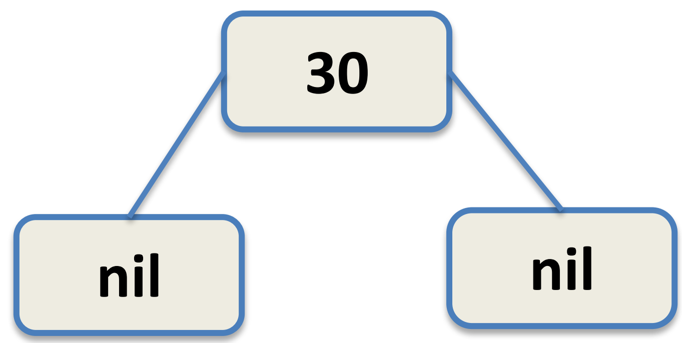
If an **overflow** occurs when trying to add a value to a node u then we **split** the node into two nodes u_1, u_2 such that

- u_1 contains the 2 smallest values (of the 4)
- u_2 contains the largest values (of the 4)
- The 3rd value is pushed up the parent node

Addition in 2-4 Trees

start from empty tree

add(30)

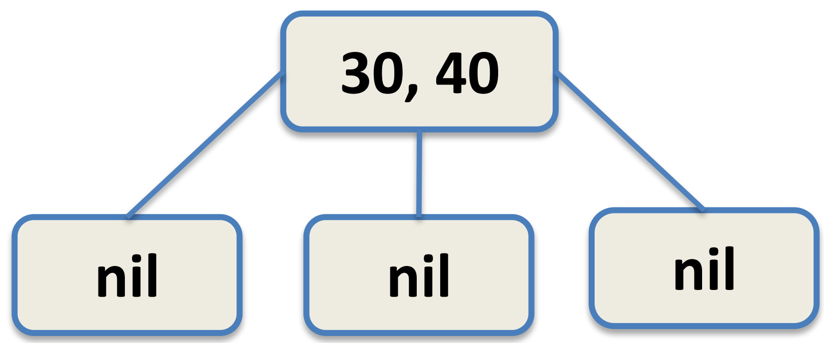


Addition in 2-4 Trees

start from empty tree

add(30)

add(40)



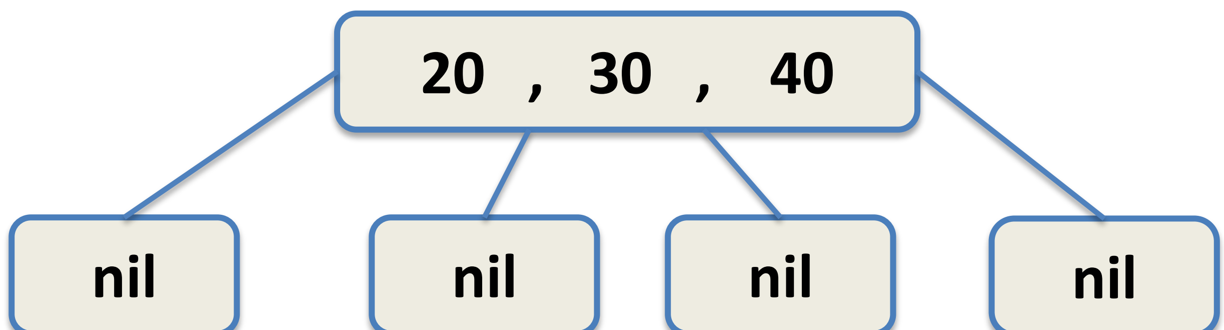
Addition in 2-4 Trees

start from empty tree

add(30)

add(40)

add(20)



Addition in 2-4 Trees

start from empty tree

add(30)

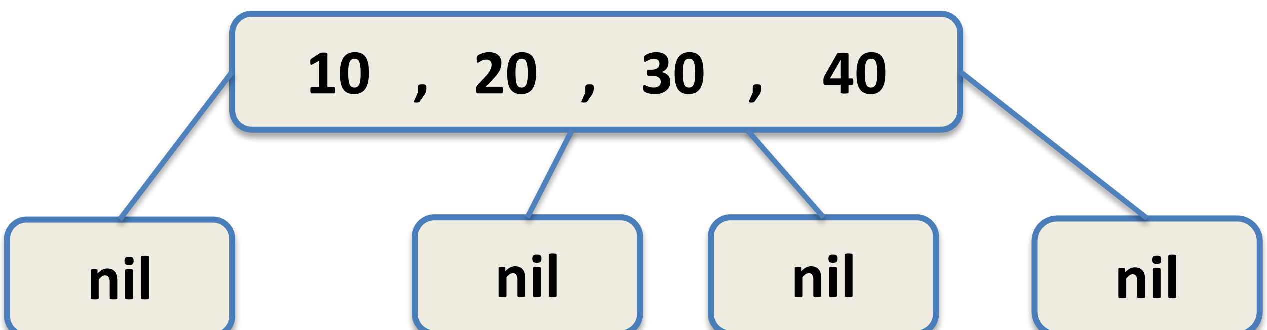
add(40)

add(20)

add(10)

overflow!

we cannot have 4 items in a node



Addition in 2-4 Trees

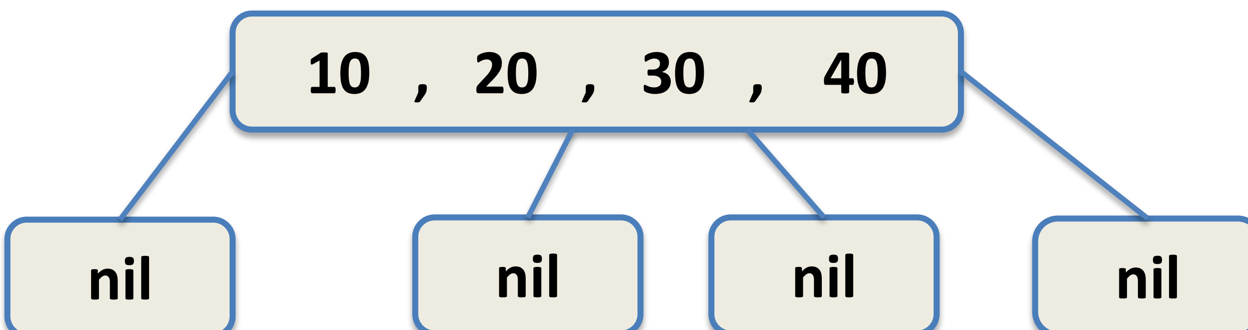
start from empty tree

Note that a Node in the tree CANNOT have 4 values. This illustration is just for a visualization purposes.

ow!

have 4

a nice



Addition in 2-4 Trees

start from empty tree

add(30)

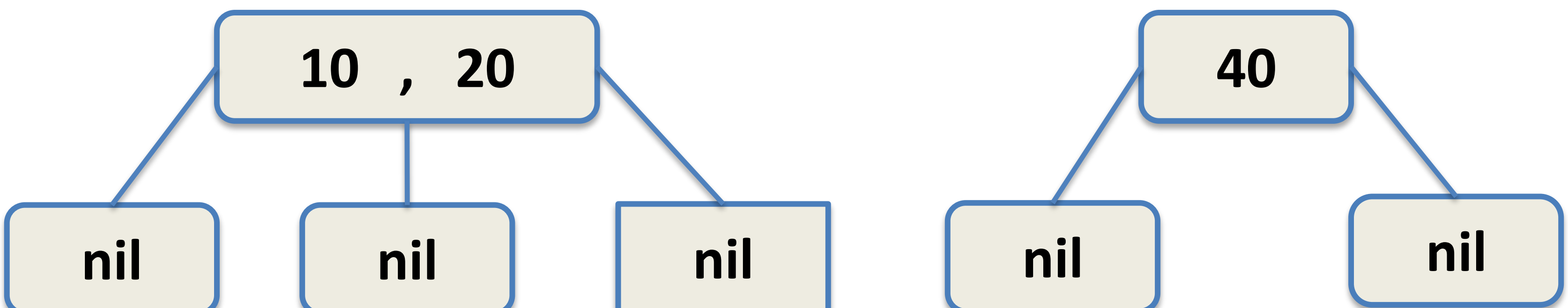
add(40)

add(20)

add(10)

split the node into
2 nodes

2 smallest (left)
largest (right)



Addition in 2-4 Trees

start from empty tree

add(30)

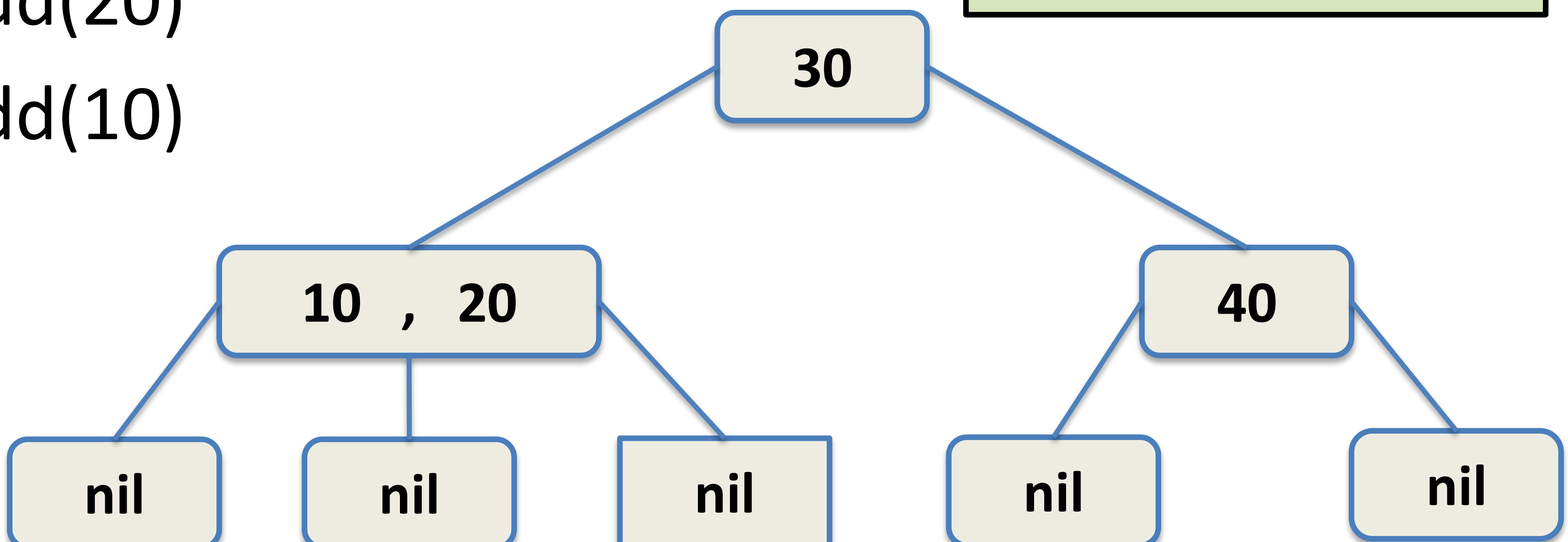
add(40)

add(20)

add(10)

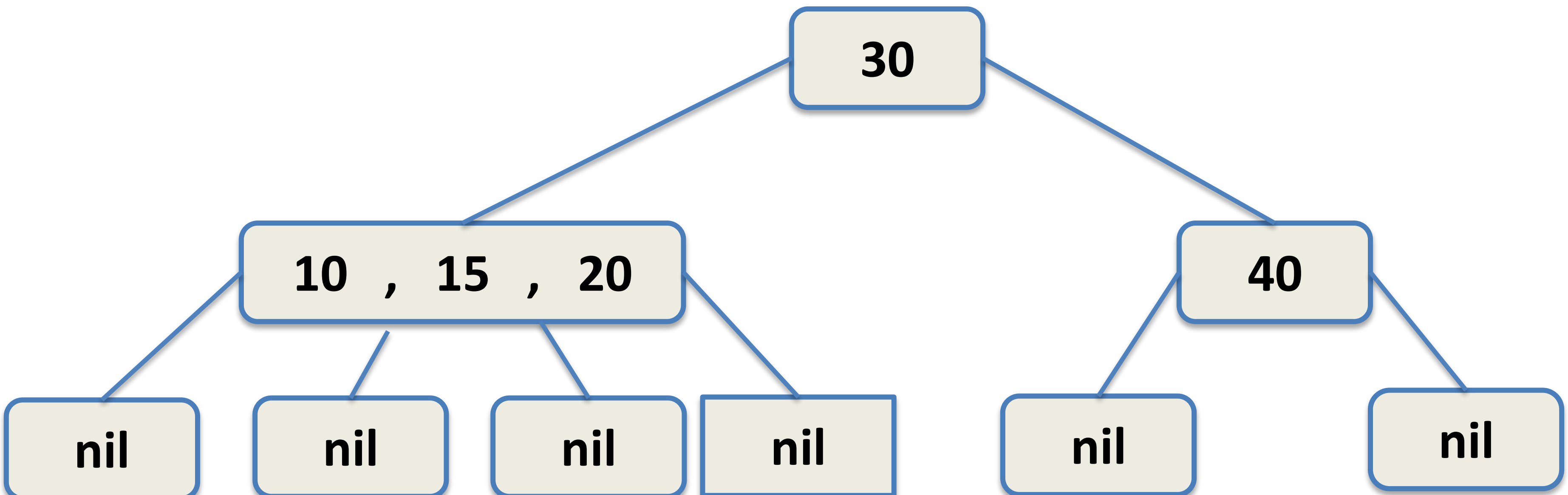
push the 3rd item up to parent node

(creating new root) in this case)



Addition in 2-4 Trees

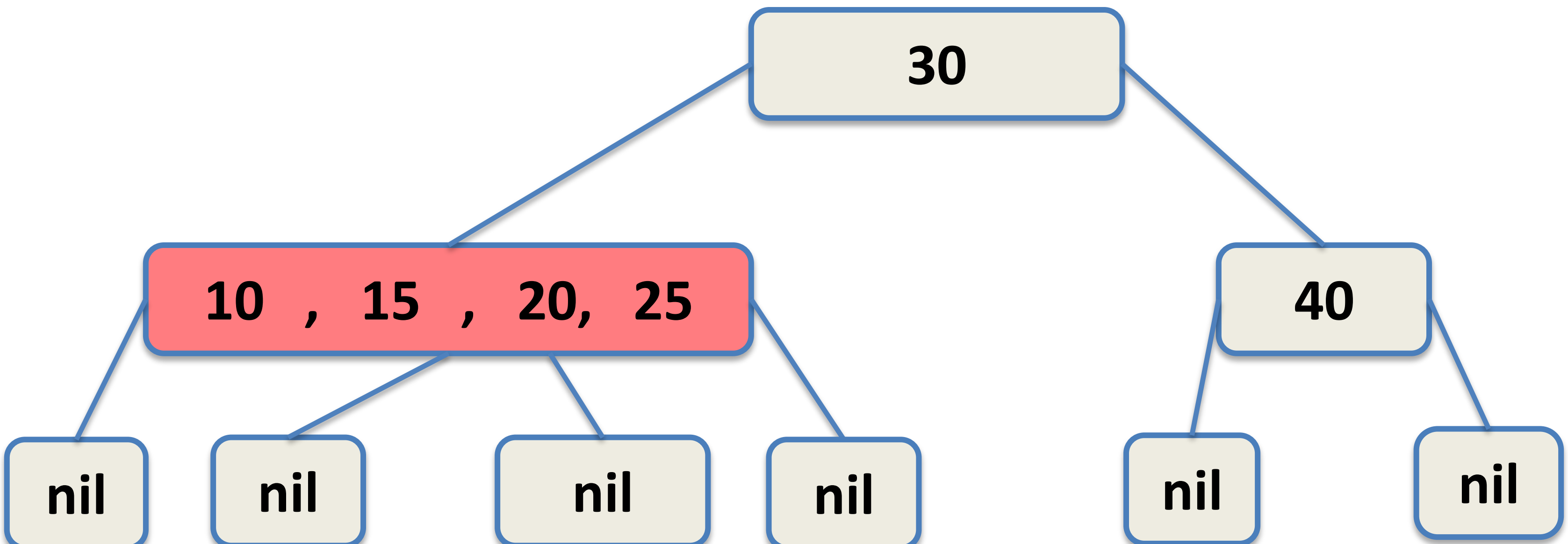
add(15)



Addition in 2-4 Trees

add(15)

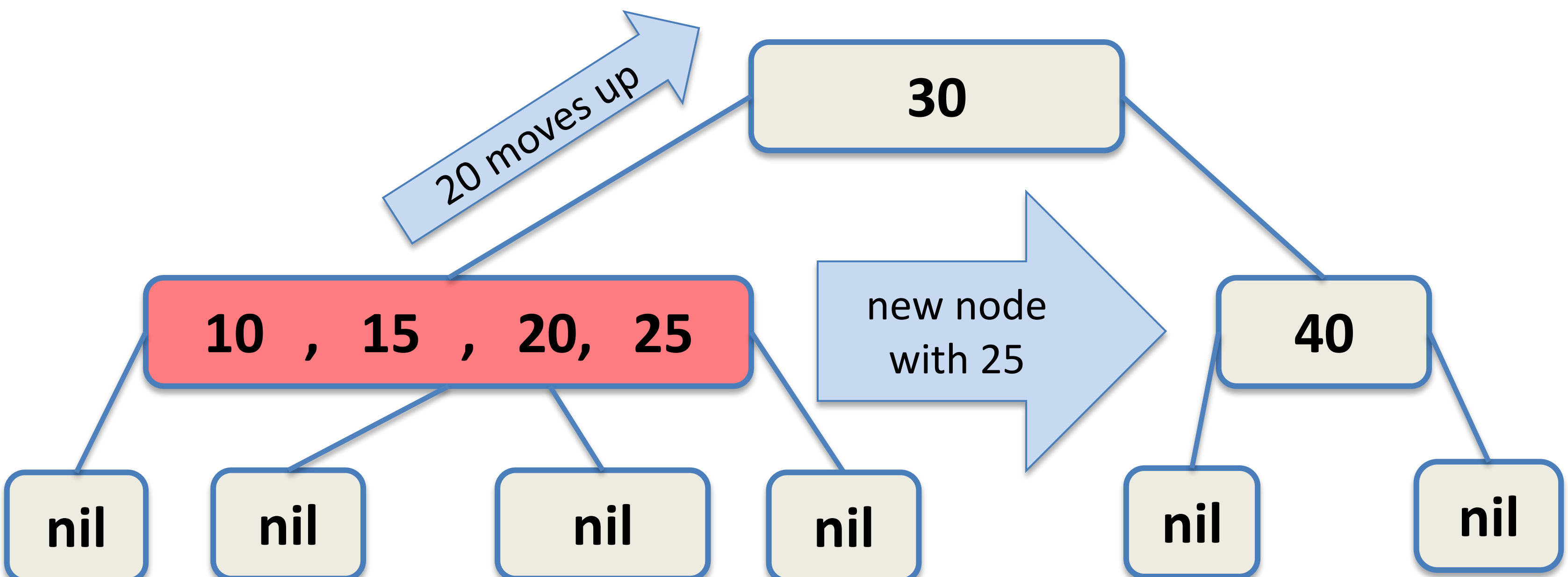
add(25) [[overflow!]]



Addition in 2-4 Trees

add(15)

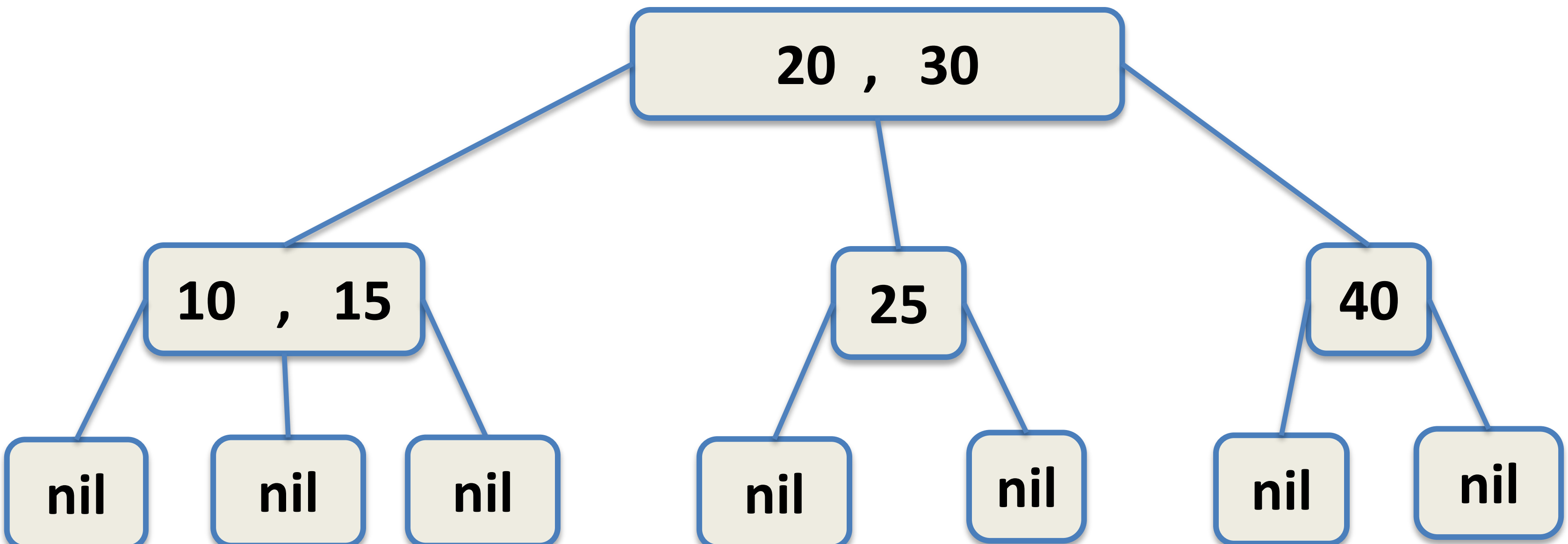
add(25) [[overflow!]



Addition in 2-4 Trees

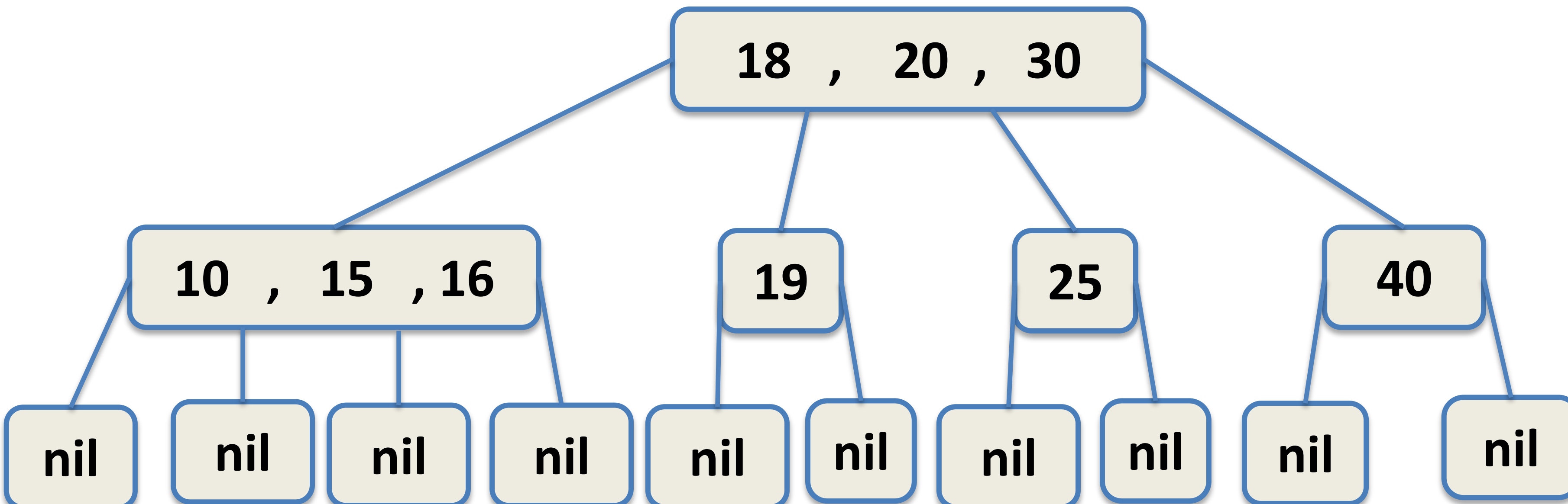
add(15)

add(25)



Addition in 2-4 Trees

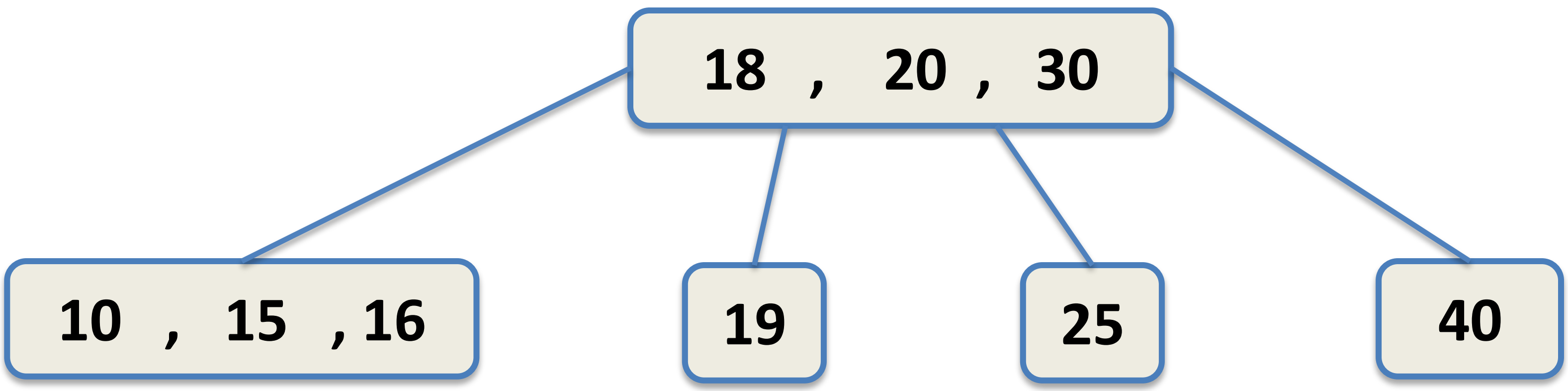
keep adding...



Addition in 2-4 Trees

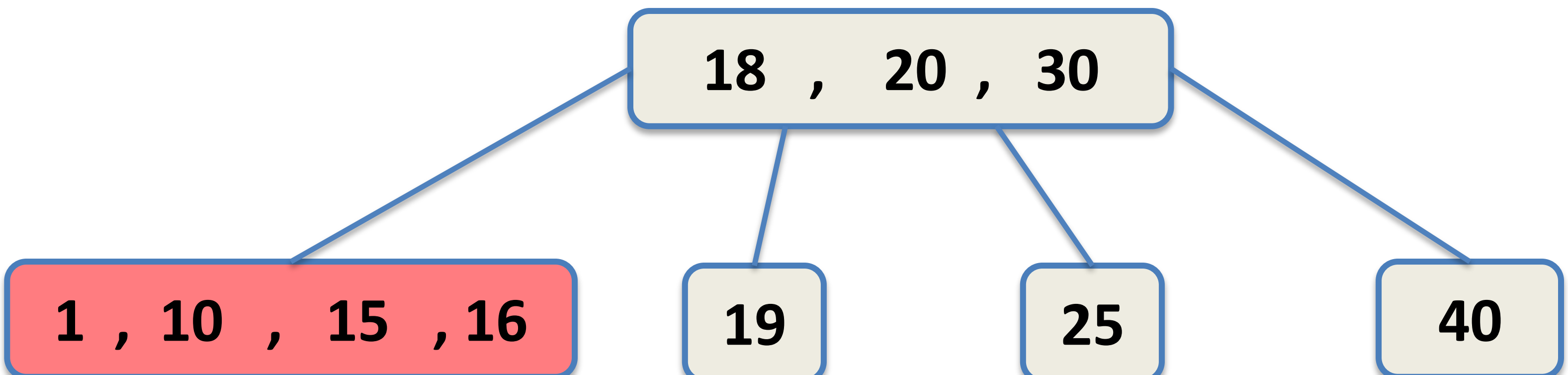
keep adding…

all external nodes are NIL so let's just not draw them and we'll know they are there…



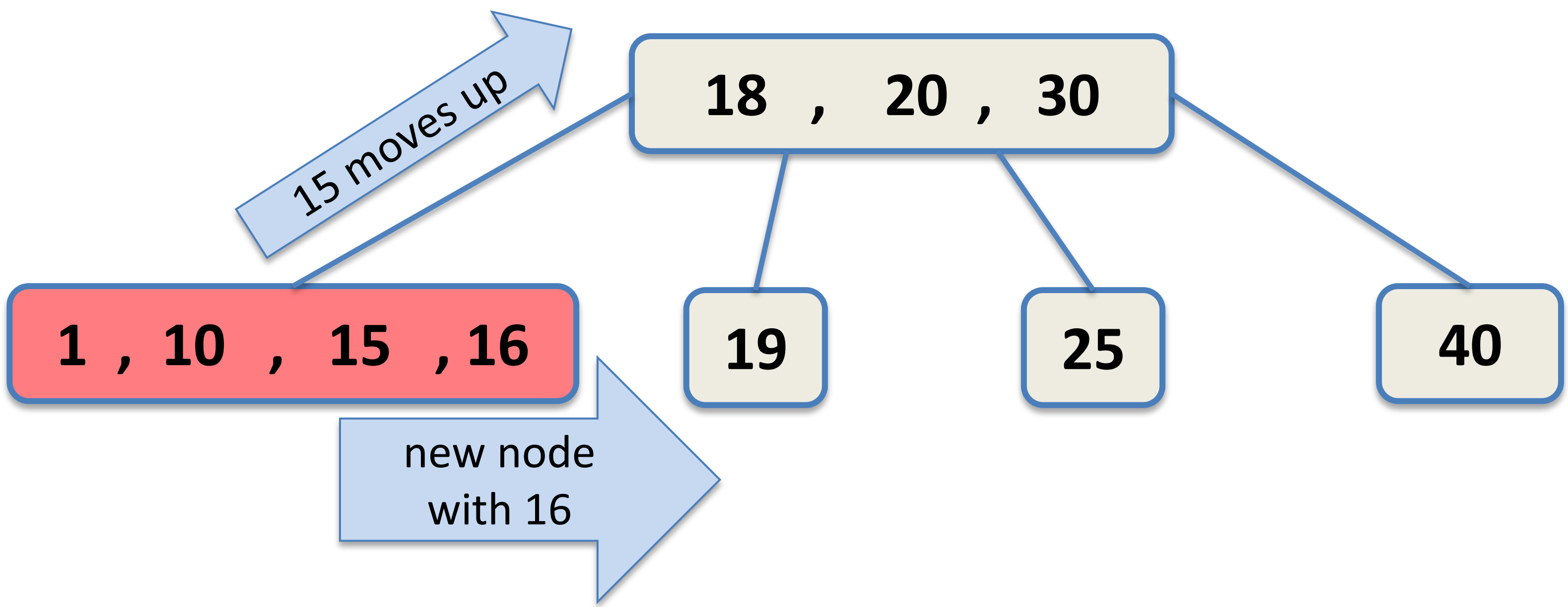
Addition in 2-4 Trees

add(1) <<overflow>>



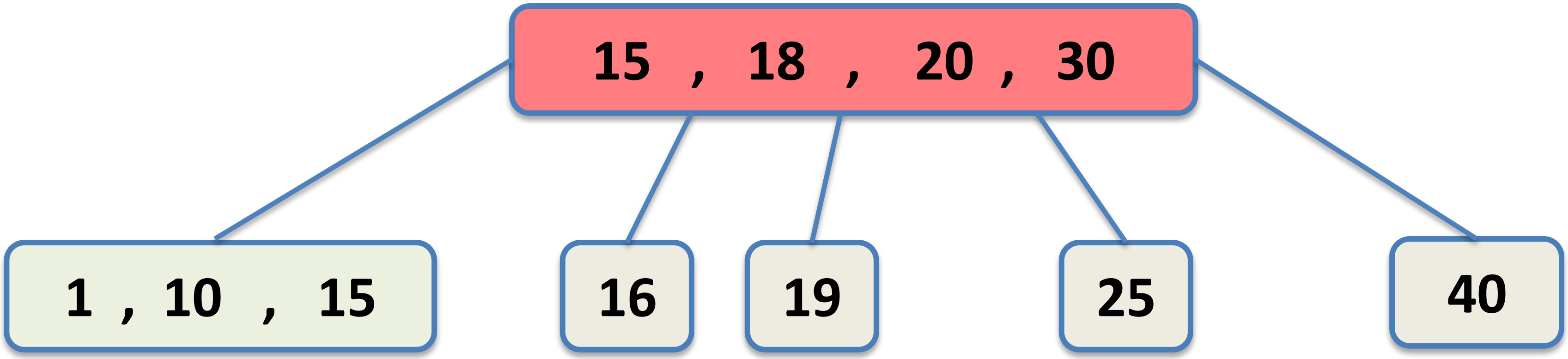
Addition in 2-4 Trees

add(1) <<overflow>>



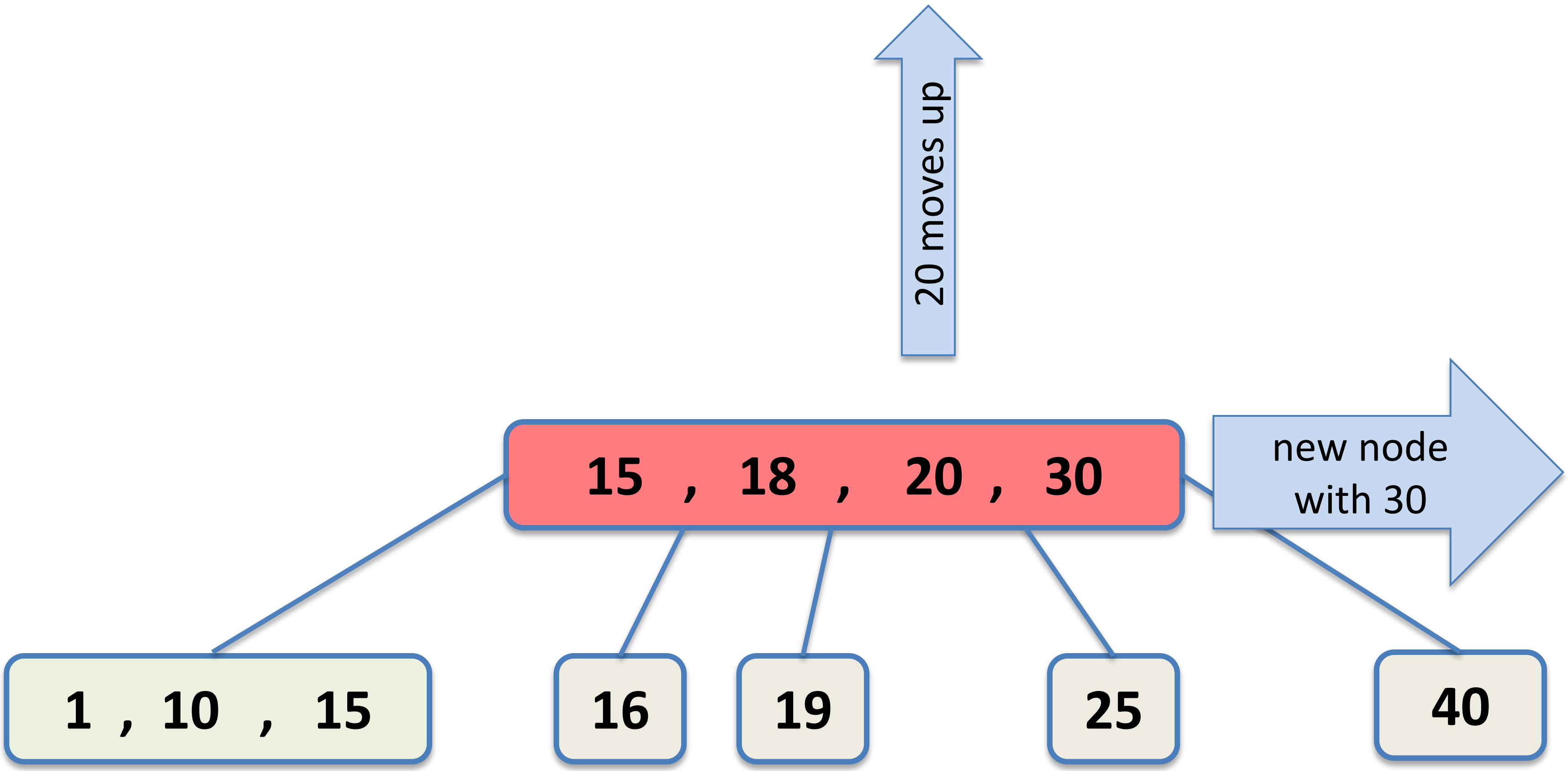
Addition in 2-4 Trees

add(1) <<overflow>>



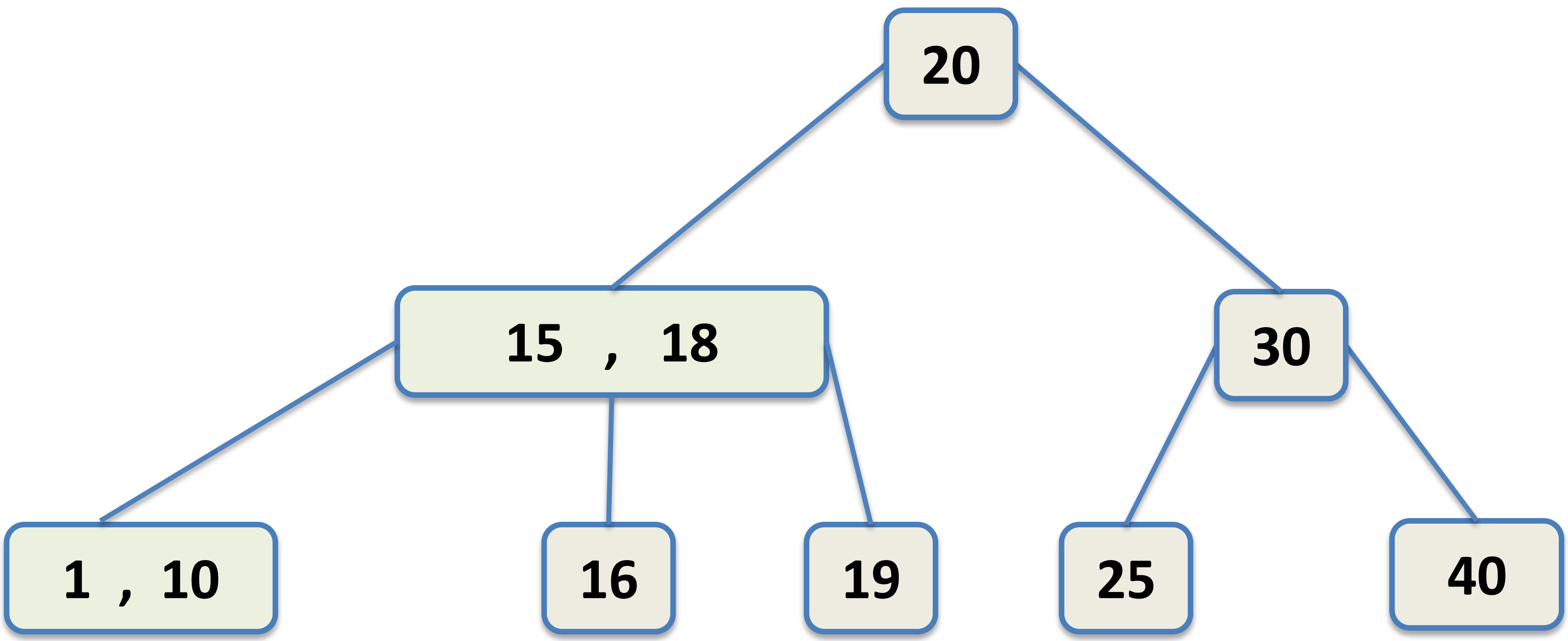
Addition in 2-4 Trees

add(1) <<overflow>>



Addition in 2-4 Trees

add(1) <<overflow>>



2-4 Trees

An **overflow** may cascade all the way up to the root node in the tree.

What is the cost of adding to a 2-4 tree?

2-4 Trees

Removing a value (key) to a 2-4 tree.

Just as with a BST you find the element to remove. If it is not at the bottom of the tree then you swap it with its in-order **predecessor**. (That is the biggest element that is smaller than it.)

You then remove the node from the bottom of the tree. An **underflow** occurs if that node would be empty after removing the key.

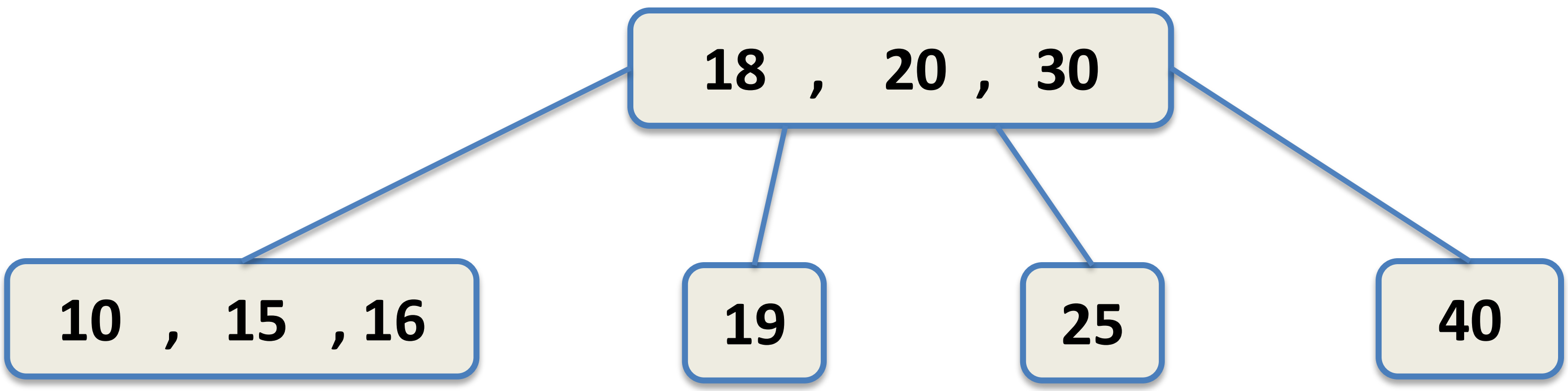
2-4 Trees

An **underflow** occurs when trying to remove a value from a node with only one key.

- If a sibling node has **enough** values we perform a **transfer**. We take a key from the parent and replace it with a key from the sibling node
- If sibling doesn’t have enough values we perform a **fuse**. Merge (empty) node with sibling and take a key from the parent.

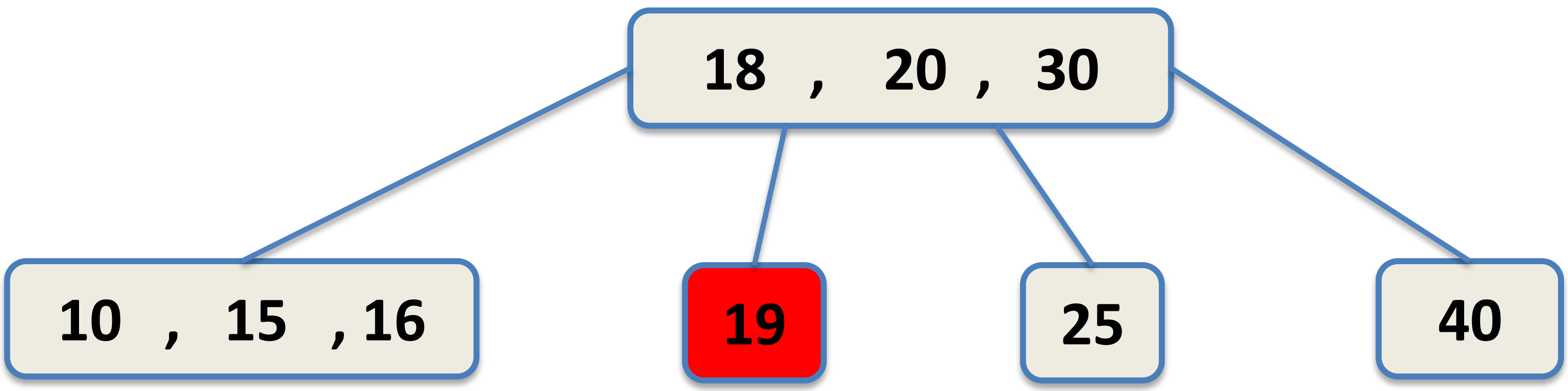
Removal in 2-4 Trees

remove(19)



Removal in 2-4 Trees

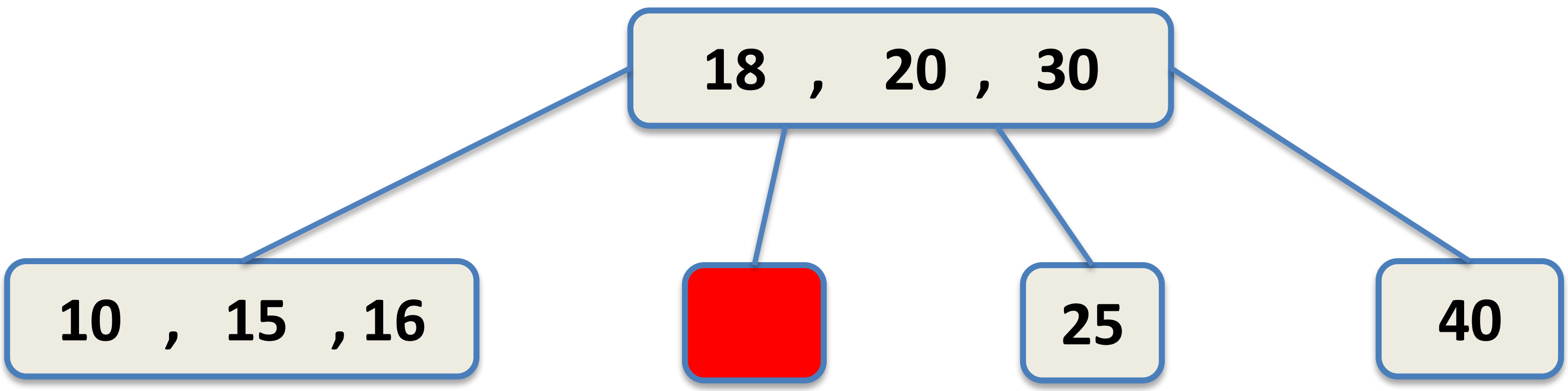
remove(19)



Removal in 2-4 Trees

remove(19)

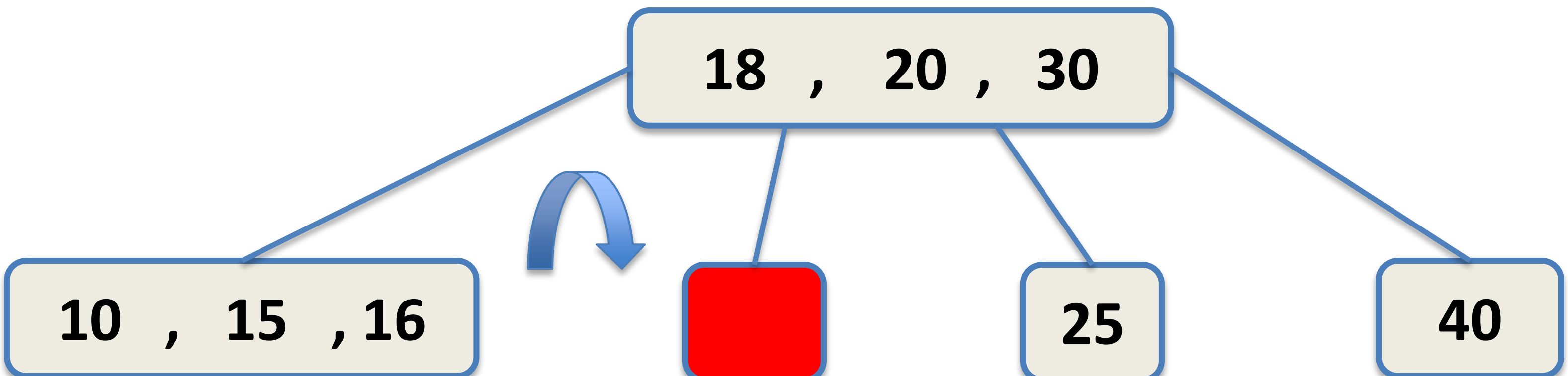
we have an **underflow** since the node is now empty



Removal in 2-4 Trees

remove(19)

we have an **underflow** since the node is now empty

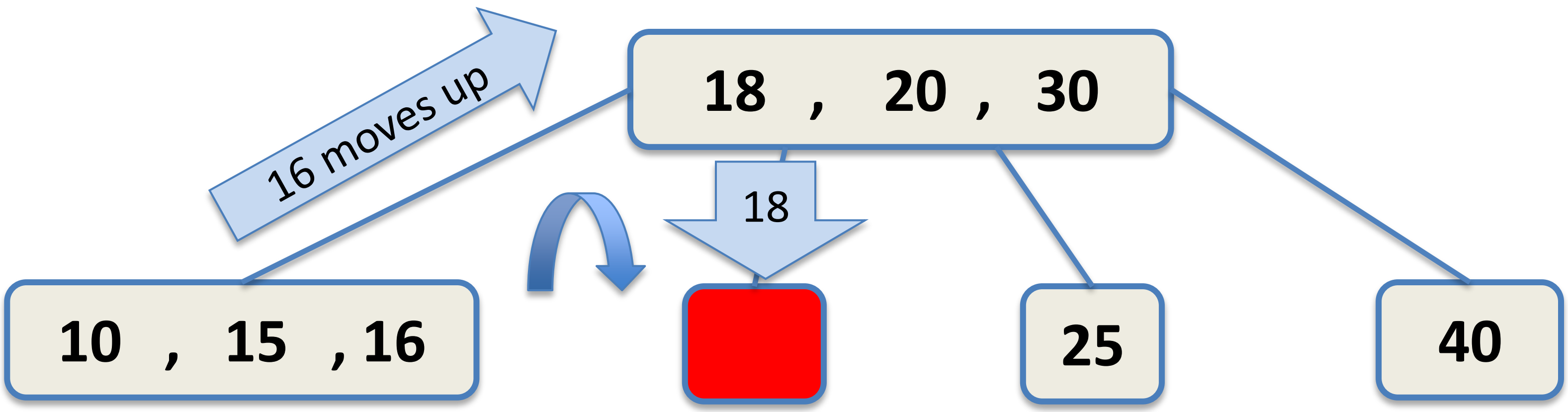


left sibling has enough values (more than 1) so we can **transfer**

Removal in 2-4 Trees

remove(19)

we have an **underflow** since the node is now empty

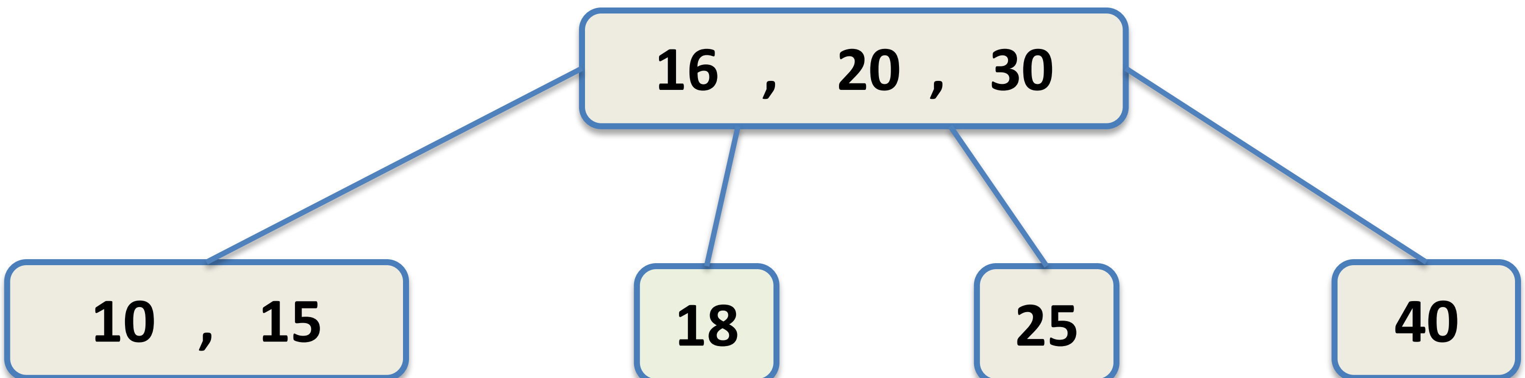


left sibling has enough values (more than 1) so we can **transfer**

Removal in 2-4 Trees

remove(19)

we have an **underflow** since the node is now empty



left sibling has enough values (more than 1) so we can **transfer**

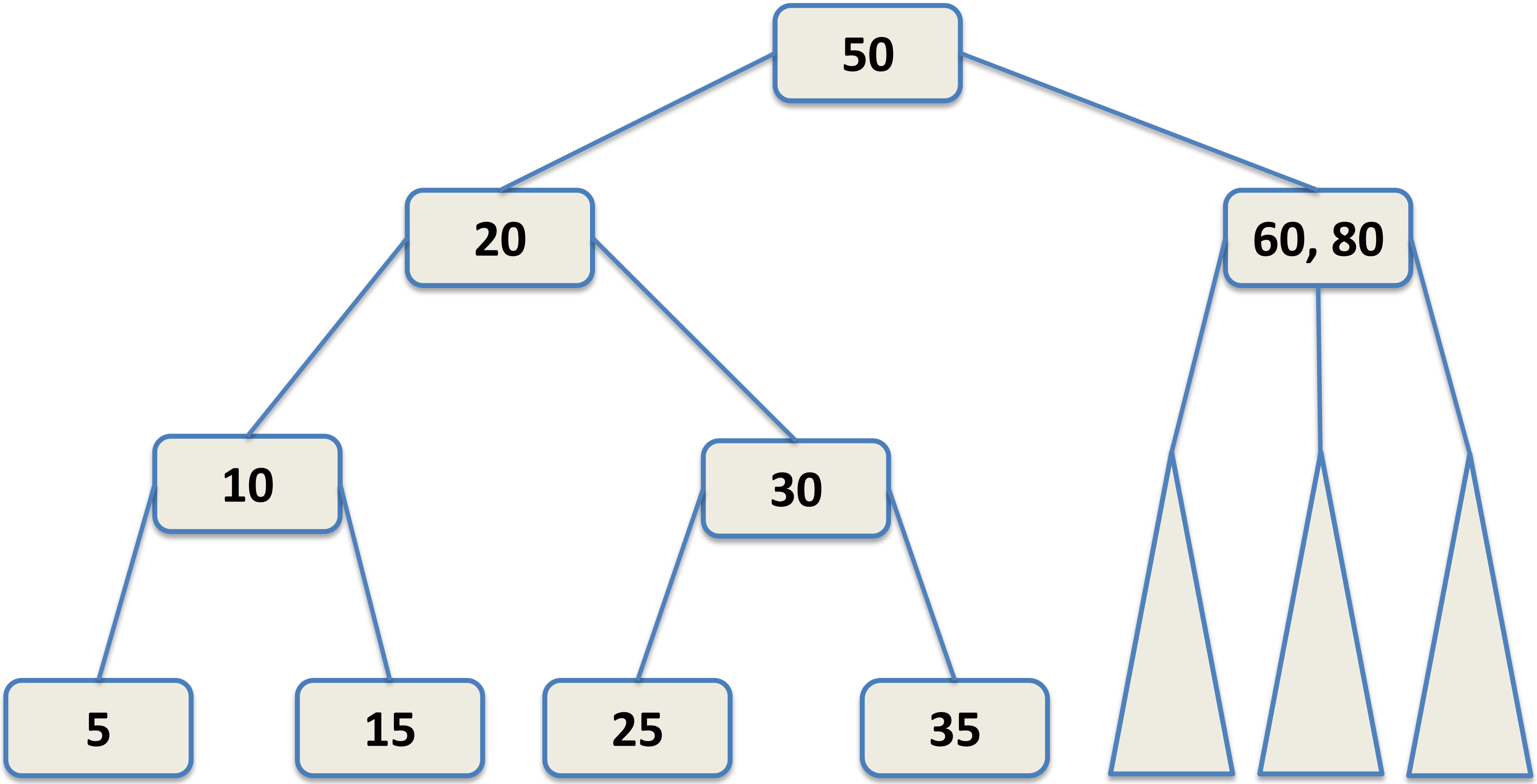
2-4 Trees

Note than an **underflow** may cascade up the tree just like an **overflow** did for additions.

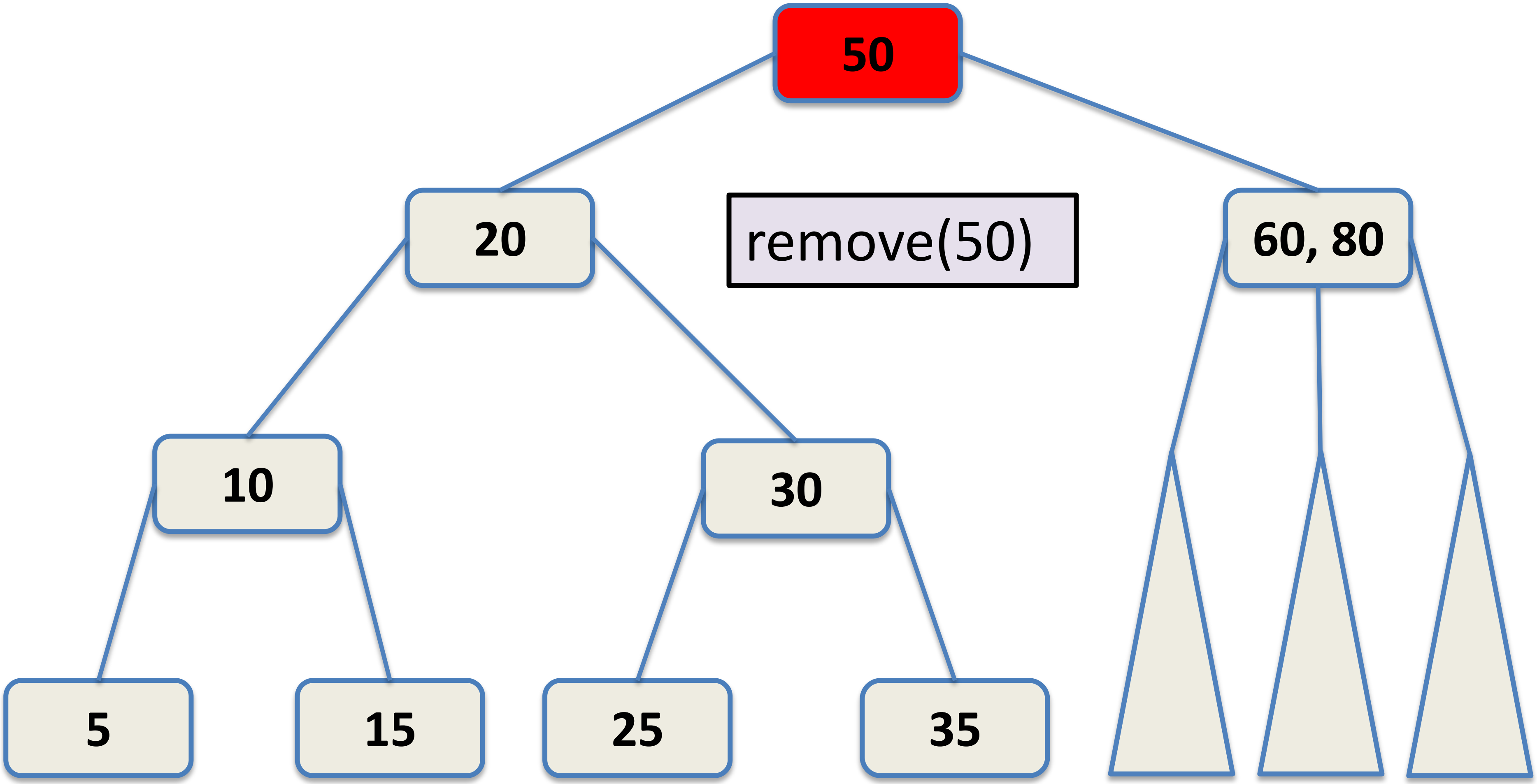
When doing a **transfer**,

- Always transfer from the **immediate** sibling that has **more** values
- If there is a tie, always transfer from the **left** sibling.

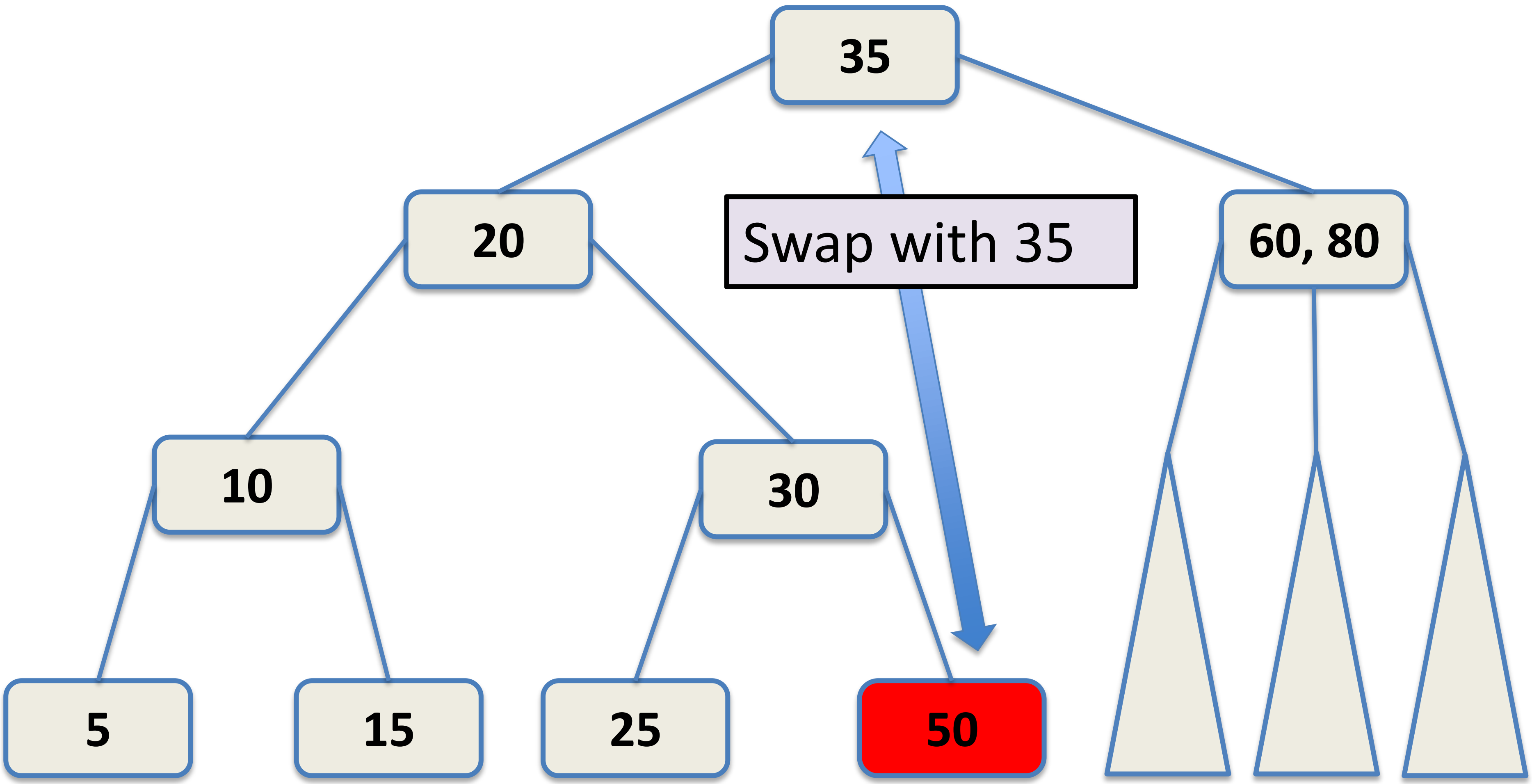
Removal in 2-4 Trees



Removal in 2-4 Trees

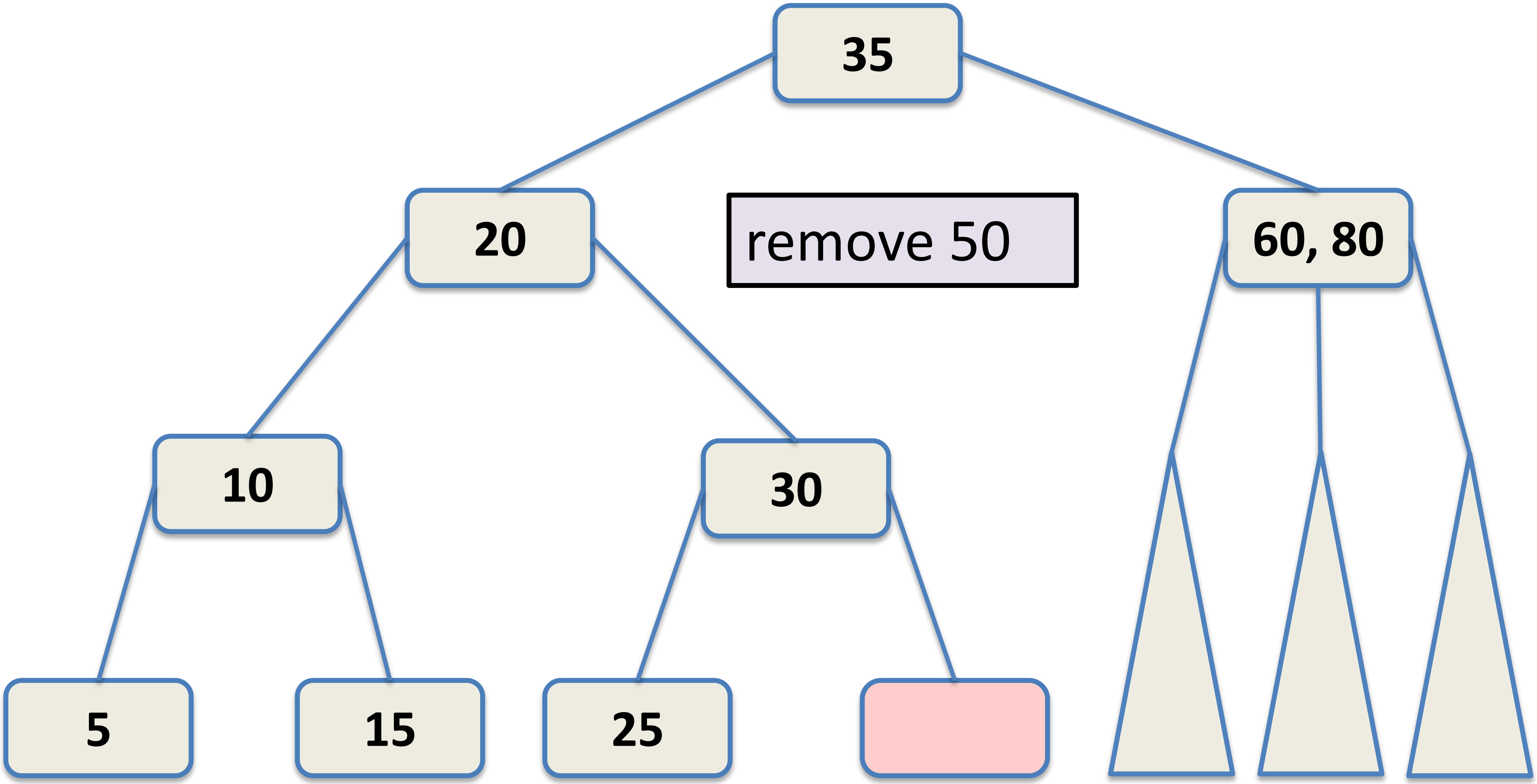


Removal in 2-4 Trees



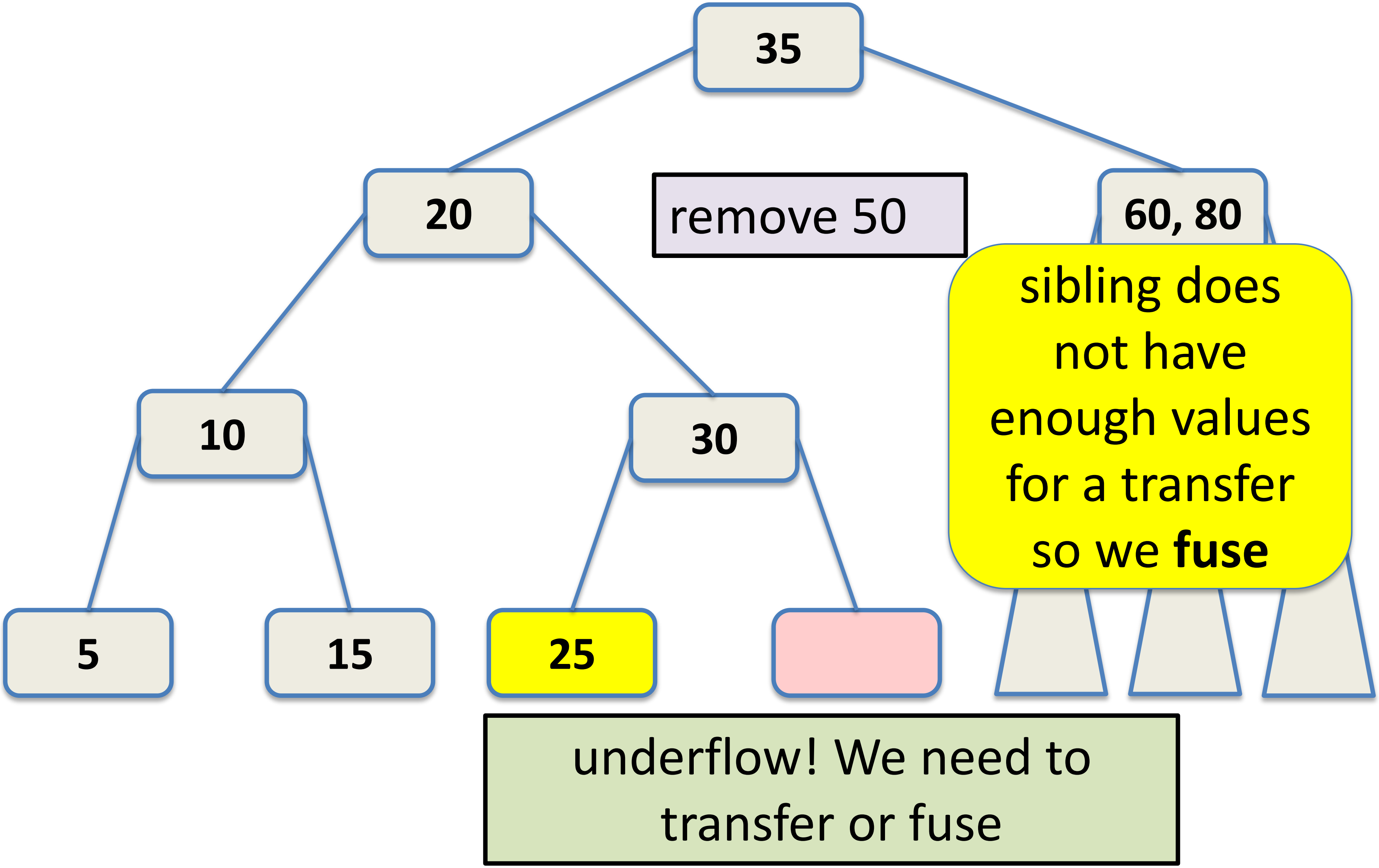
swap with **predecessor**

Removal in 2-4 Trees

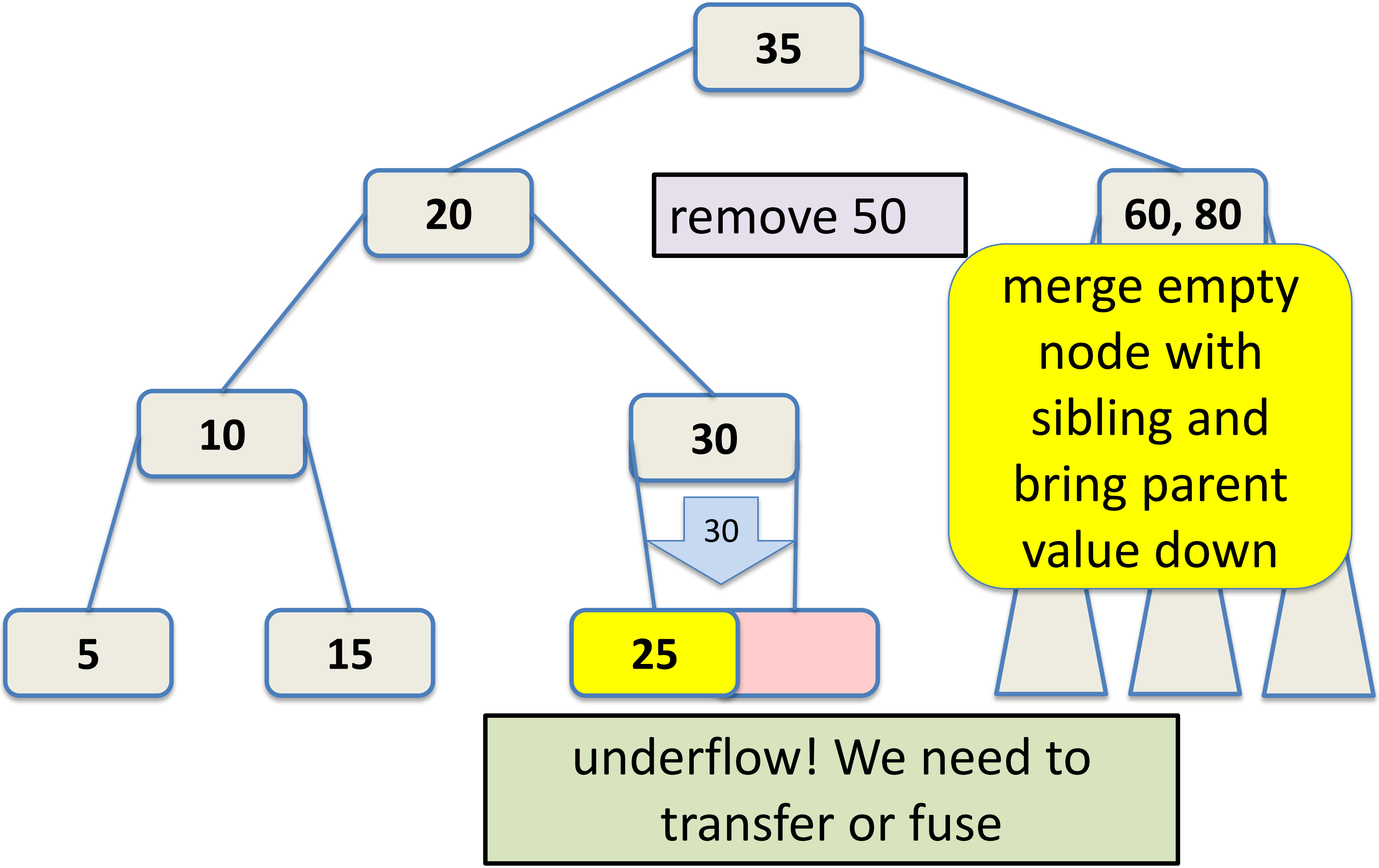


underflow! We need to transfer or fuse

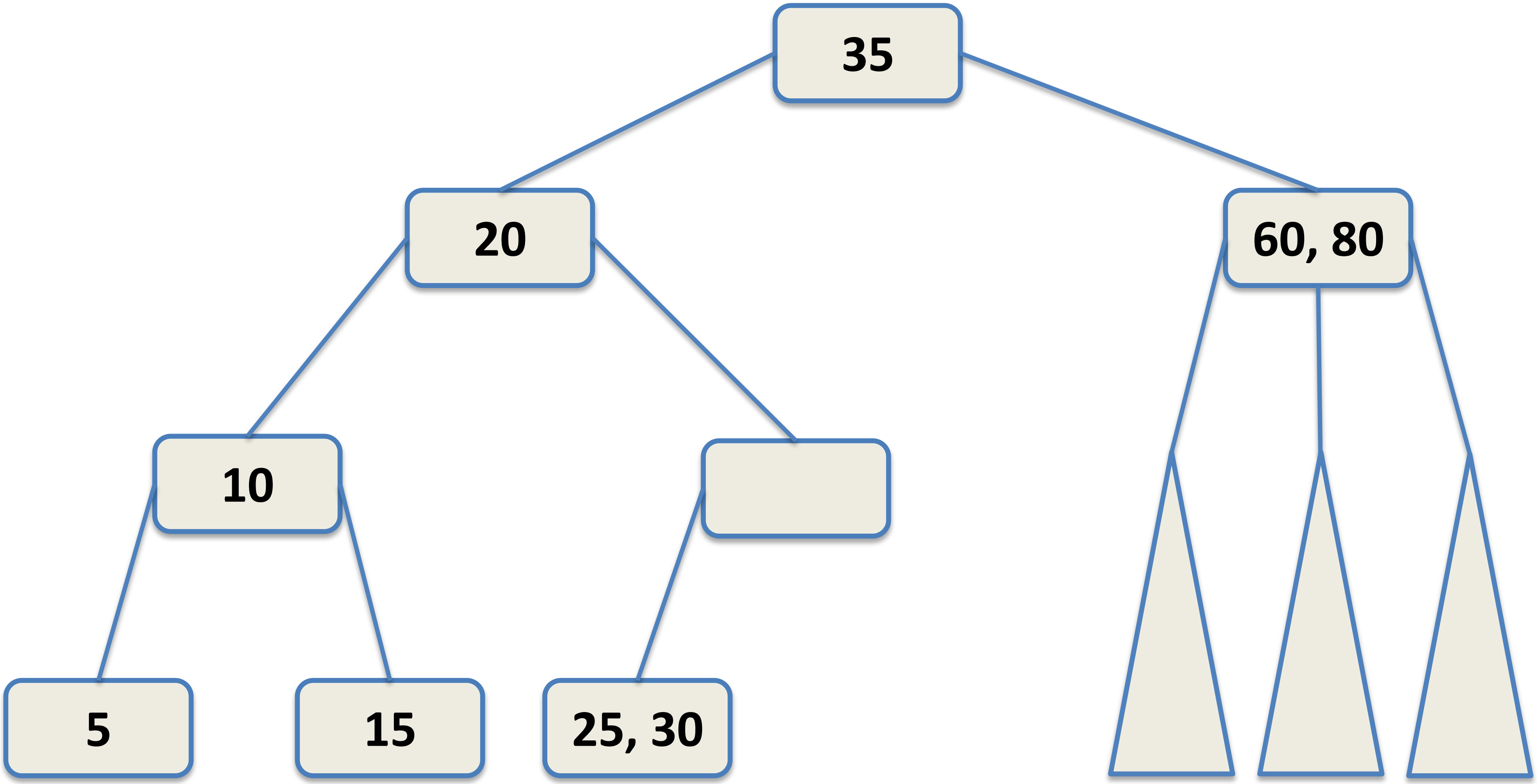
Removal in 2-4 Trees



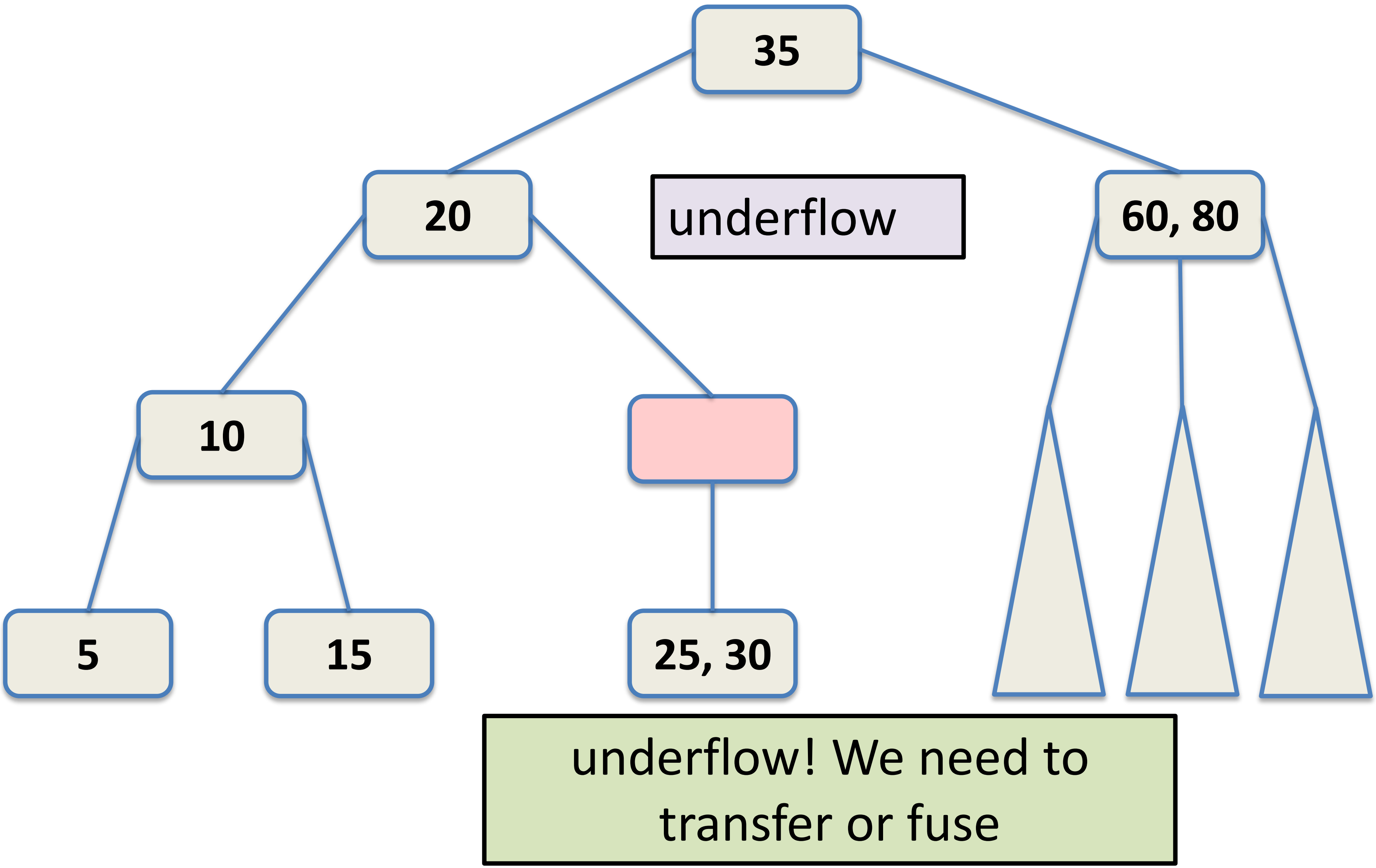
Removal in 2-4 Trees



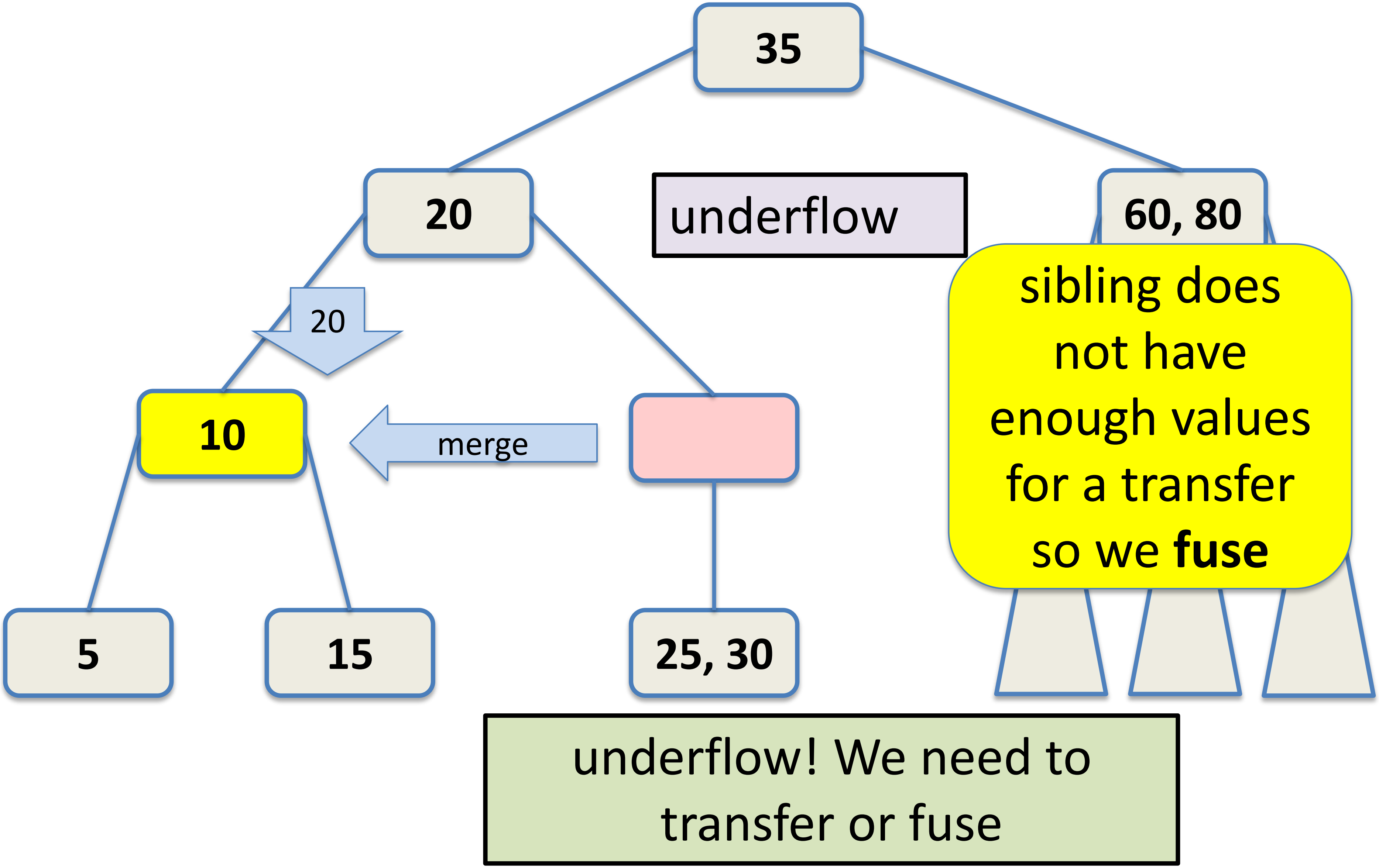
Removal in 2-4 Trees



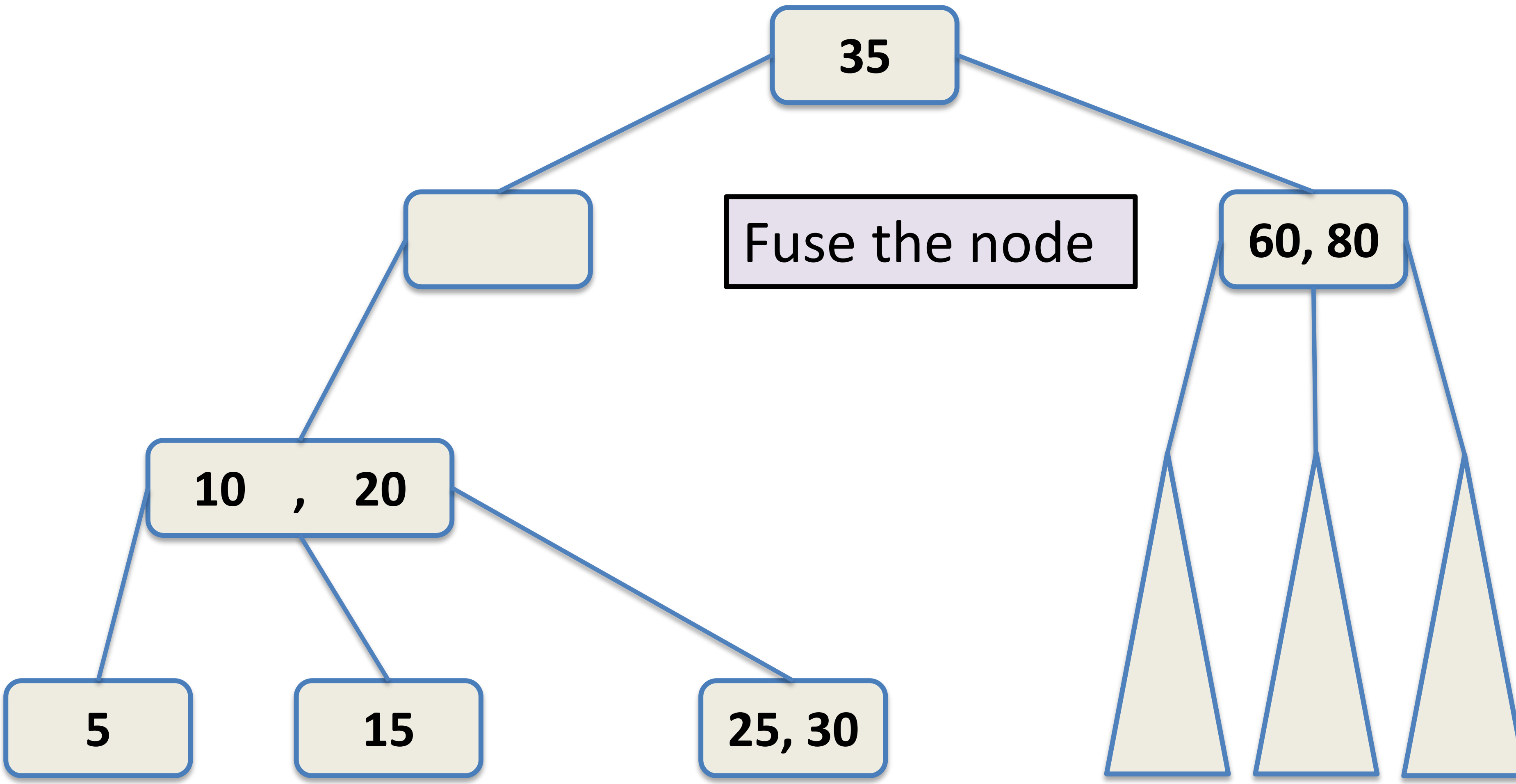
Removal in 2-4 Trees



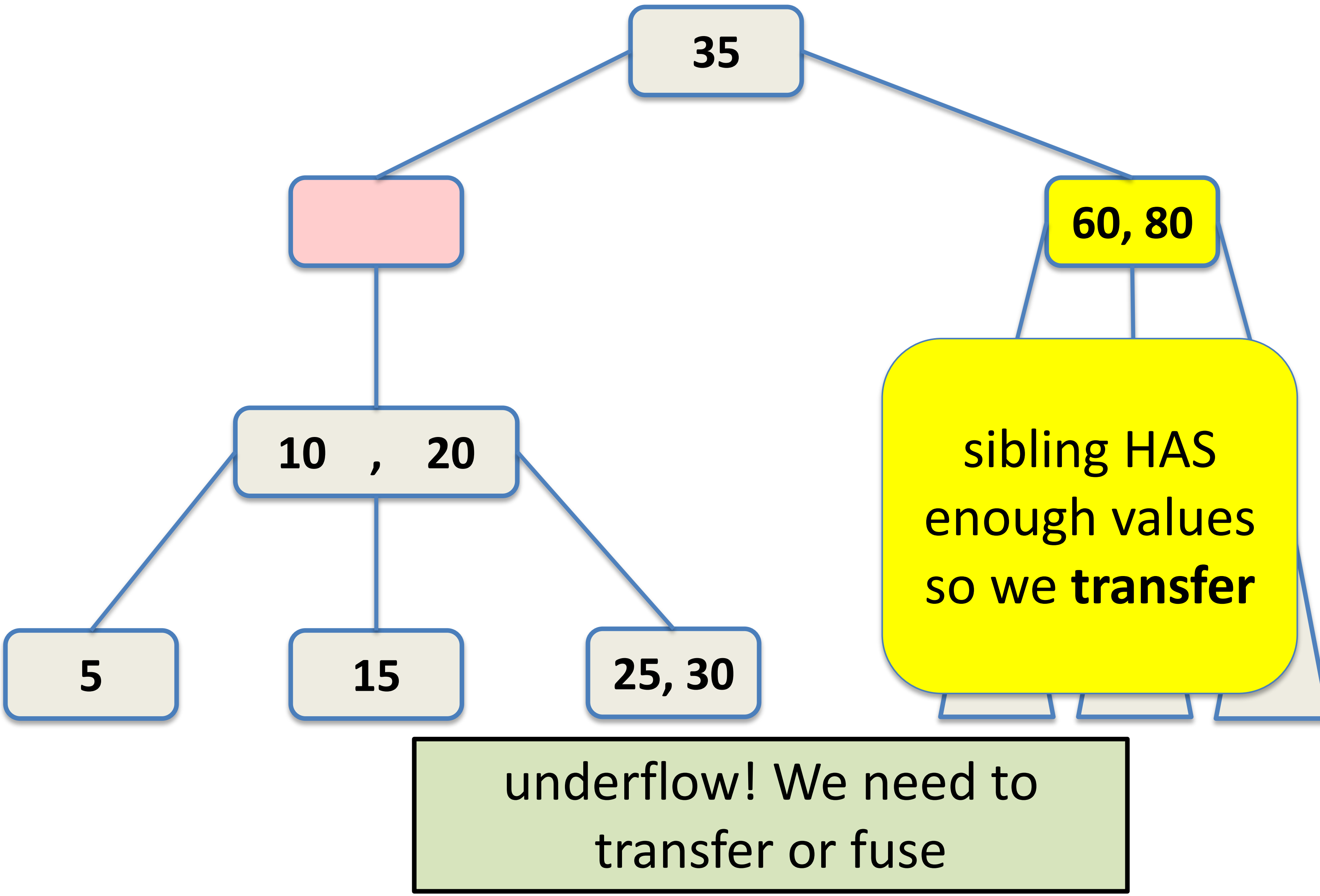
Removal in 2-4 Trees



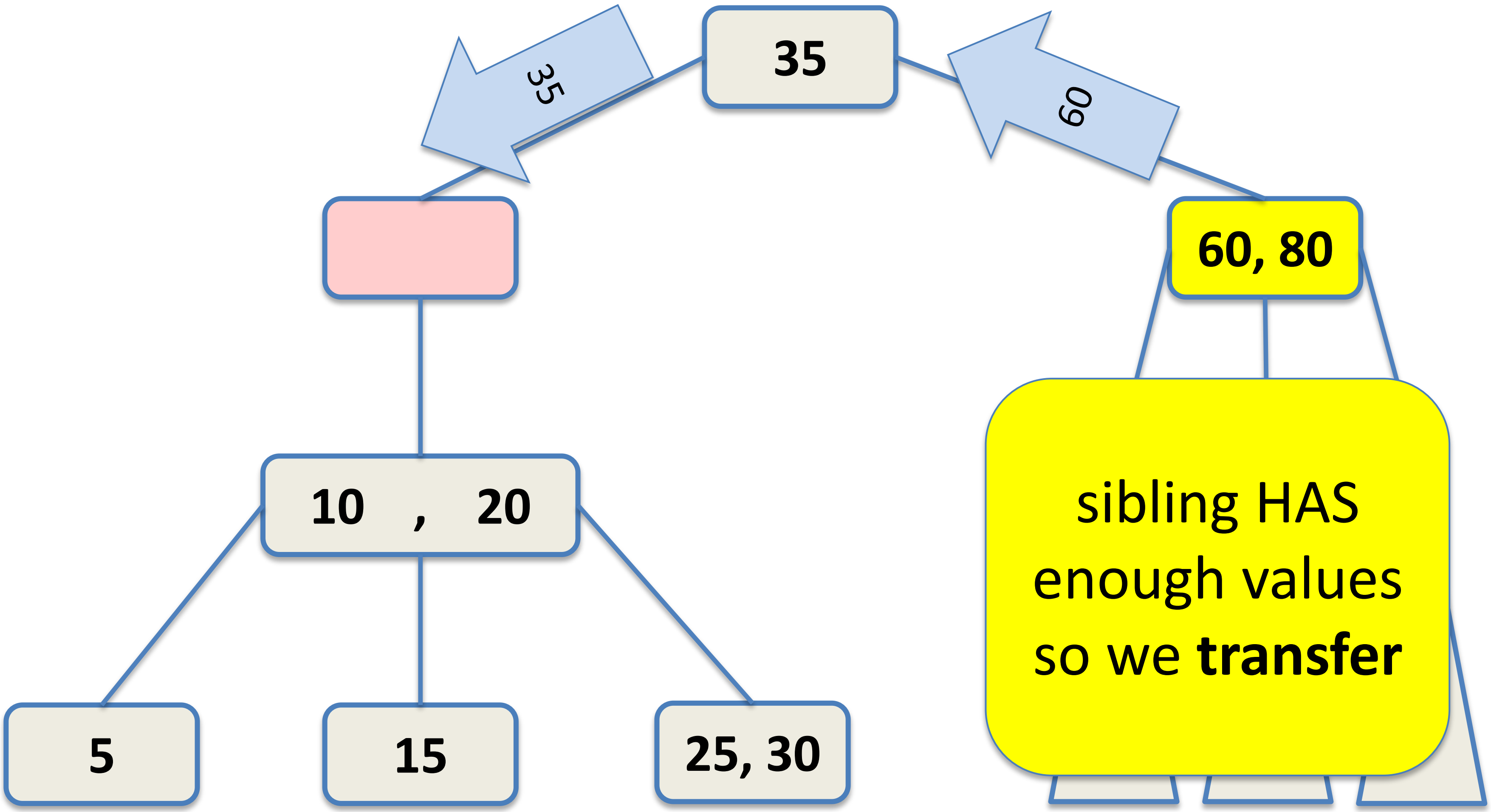
Removal in 2-4 Trees



Removal in 2-4 Trees

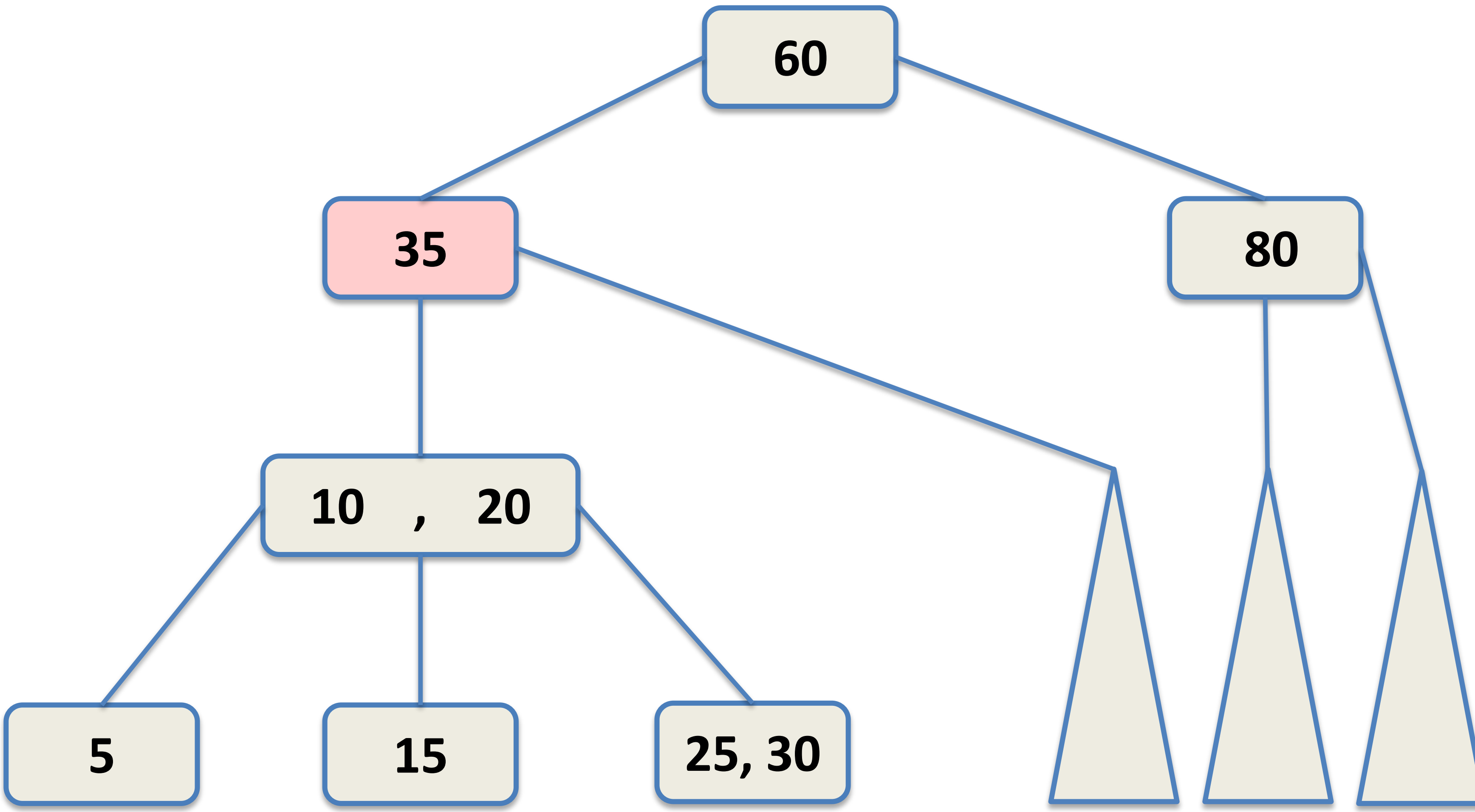


Removal in 2-4 Trees

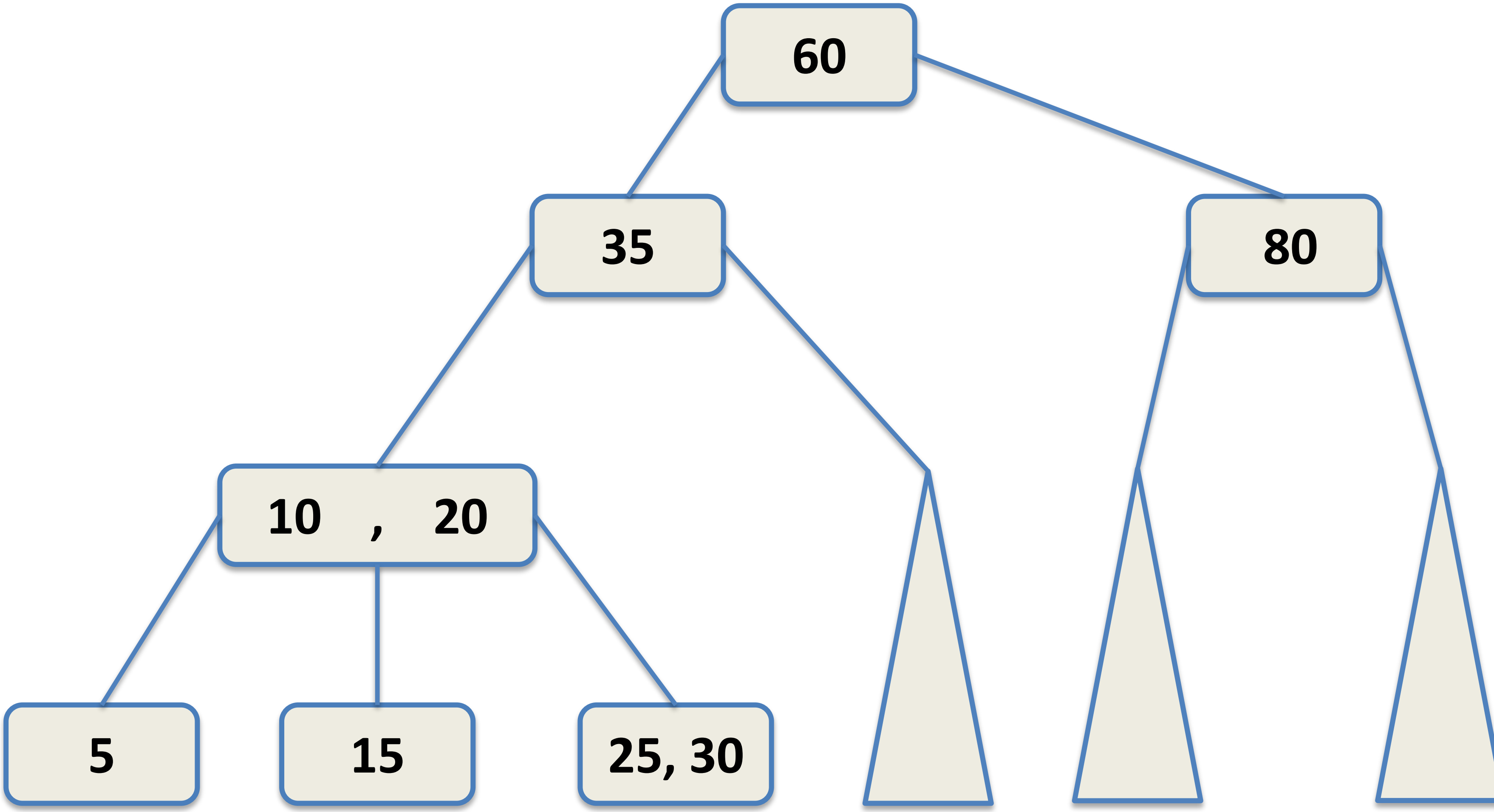


underflow! We need to transfer or fuse

Removal in 2-4 Trees



Removal in 2-4 Trees



2-4 Trees

Lemma A 2-4 Tree implements the SSet interface and supports the operations `add(x)`, `remove(x)` and `find(x)` in $O(\log n)$ time per operation

Removal in 2-4 Trees

remove(40)

there is a sibling that has enough values for a transfer, but not an immediate sibling.