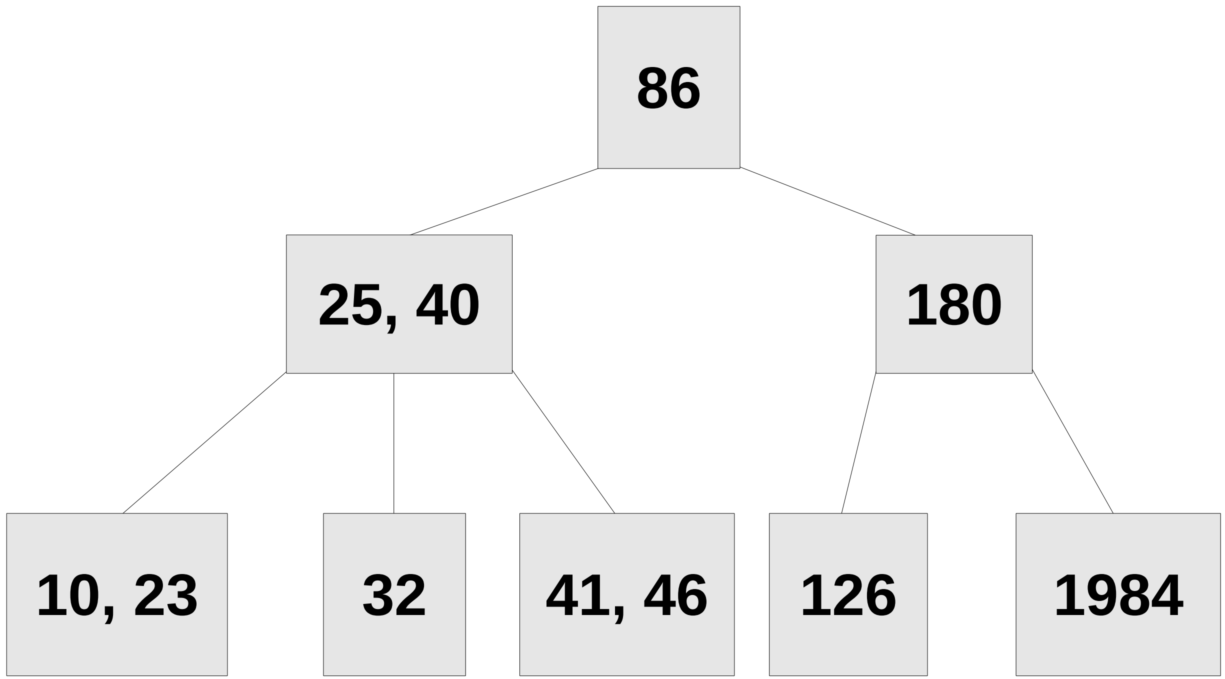
# 2-4 Trees

(Yet another balanced tree structure, but way more fun than AVL trees.)




Slides by **Sean Szumlanski**

for **COP 3503**, Computer Science II

*Fall 2021*

# 2-4 Tree Insertion

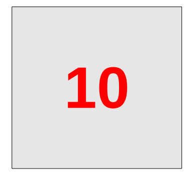
(Let's play the game!)

Insert 10 (into an initially empty tree):

# 2-4 Tree Insertion

(Let’s play the game!)

Insert 10 (into an initially empty tree):



# 2-4 Tree Insertion

(Let’s play the game!)

Insert 40:

**10**

# 2-4 Tree Insertion

(Let's play the game!)

Insert 40:

**10, 40**

# 2-4 Tree Insertion

(Let's play the game!)

Insert 180:

**10, 40**

# 2-4 Tree Insertion

(Let’s play the game!)

Insert 180:

| |
|---|
| **10, 40, 180** |

(Things are getting nice and cozy in there!)

# 2-4 Tree Insertion

(Let’s play the game!)

Insert 25:

| |
|---|
| **10, 40, 180** |

# 2-4 Tree Insertion

(Let's play the game!)

Insert 25:

**10, 25, 40, 180**

Oh no!

The node has too many elements. This is called an overflow.

It's time to...

SQUISH UP!

(This is also called "splitting," which is insufferably dull.)

Our contract this semester: Let's agree always to squish up the third element.

That's a bit arbitrary. We could also have chosen the second element. When coding these up, you just have to make a choice and stick with it.

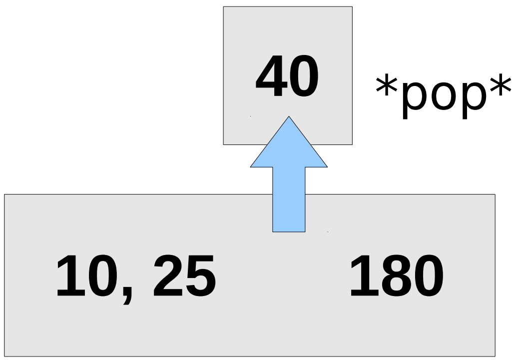
# 2-4 Tree Insertion

(Let’s play the game!)

**10, 25, 40, 180**

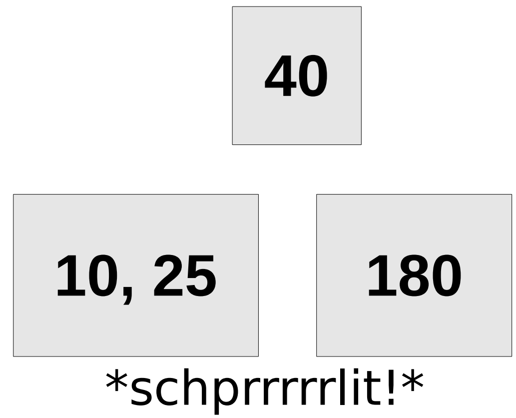
# 2-4 Tree Insertion

(Let’s play the game!)



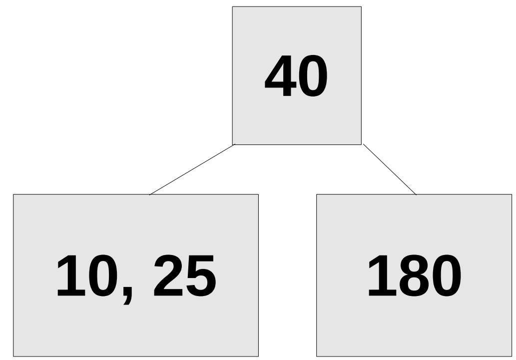
# 2-4 Tree Insertion

(Let’s play the game!)



# 2-4 Tree Insertion

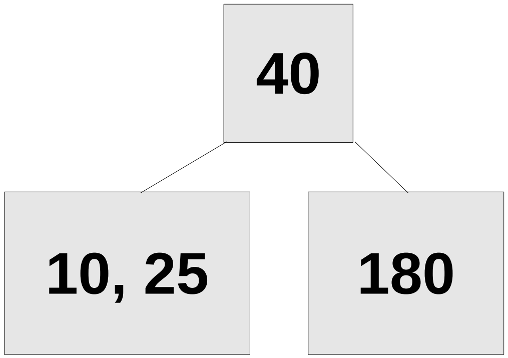
(Let’s play the game!)




*tada!*

# 2-4 Tree Insertion

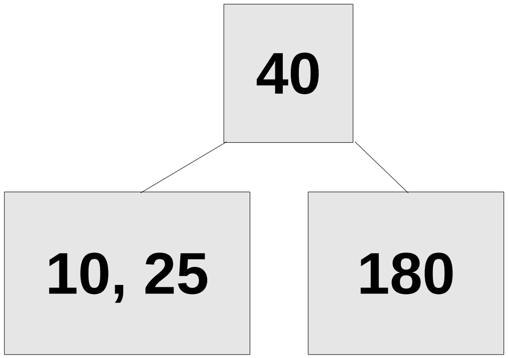
(Let’s play the game!)




Let’s continue inserting...

# 2-4 Tree Insertion

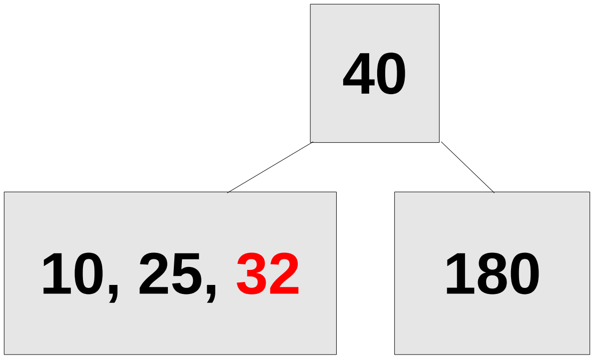
(Let’s play the game!)




Insert 32

# 2-4 Tree Insertion

(Let’s play the game!)

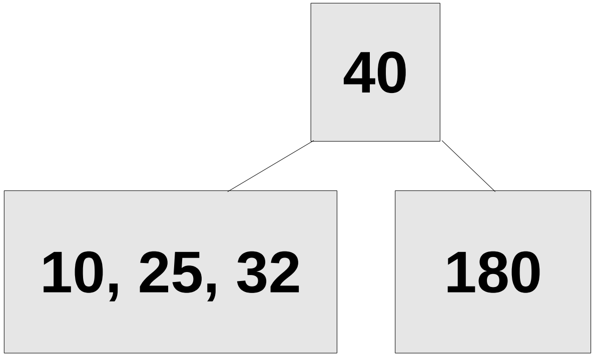



Insert 32

(Always insert at a leaf node.)

# 2-4 Tree Insertion

(Let’s play the game!)




Insert 1984

# 2-4 Tree Insertion

(Let’s play the game!)

**40**

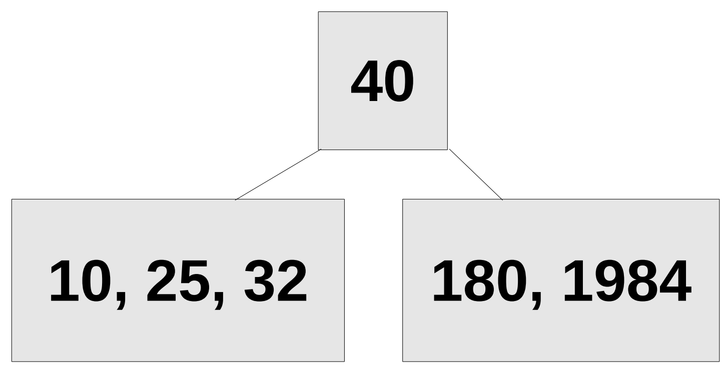
**10, 25, 32**

**180,** 


Insert 1984

# 2-4 Tree Insertion

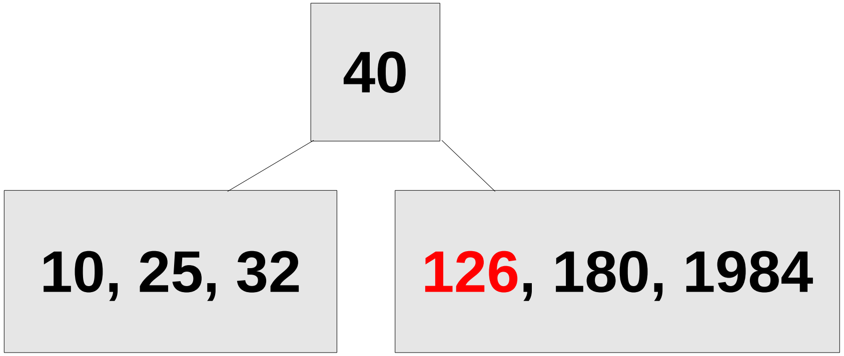
(Let’s play the game!)




Insert 126

# 2-4 Tree Insertion

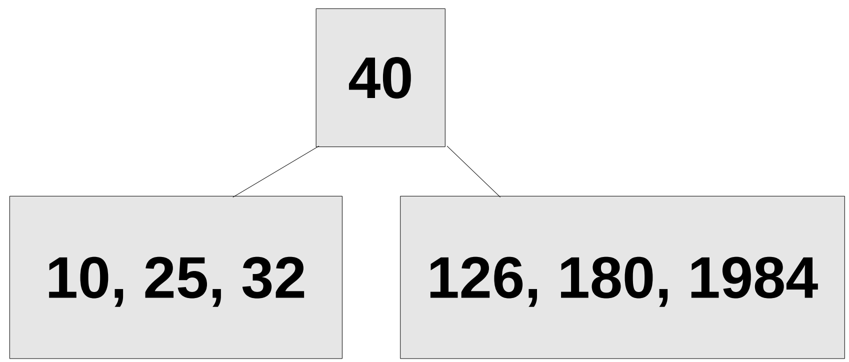
(Let’s play the game!)




Insert 126

# 2-4 Tree Insertion

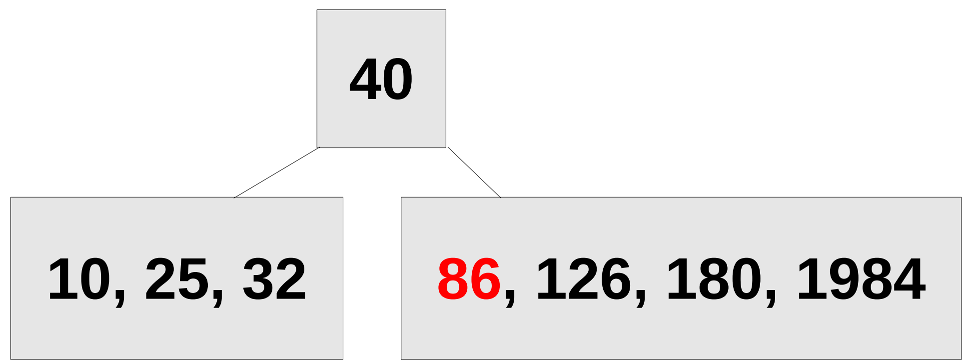
(Let’s play the game!)




Insert 86

# 2-4 Tree Insertion

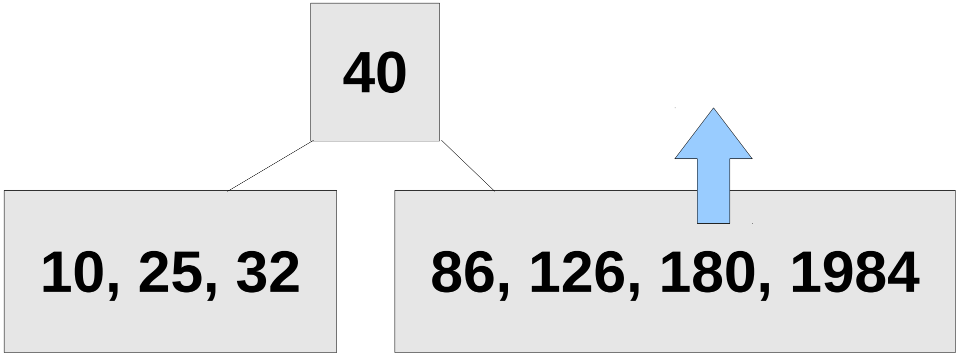
(Let’s play the game!)




Insert 86

# 2-4 Tree Insertion

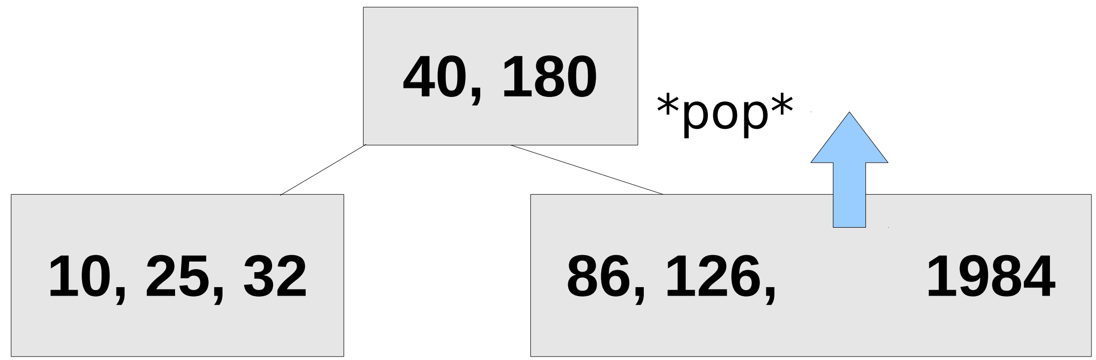
(Let’s play the game!)




Squish up! (Which element?)

# 2-4 Tree Insertion

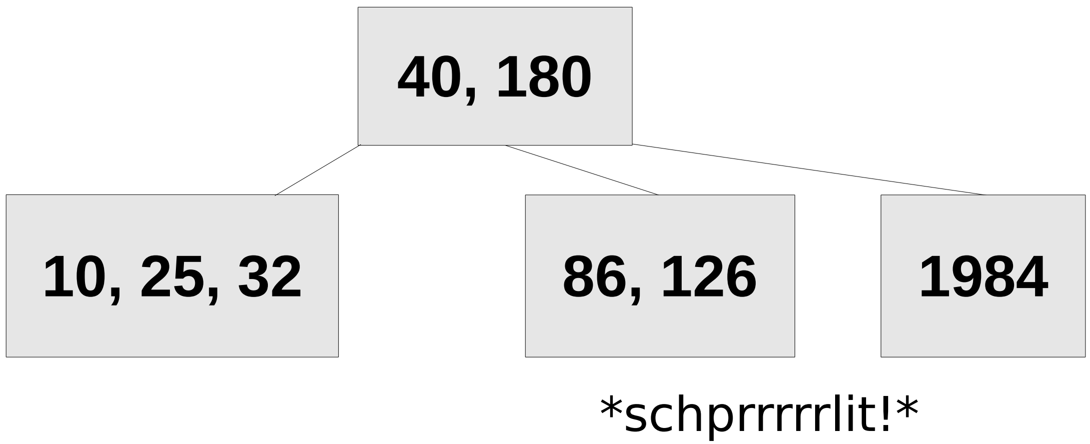
(Let's play the game!)




Squish up! (Which element?)

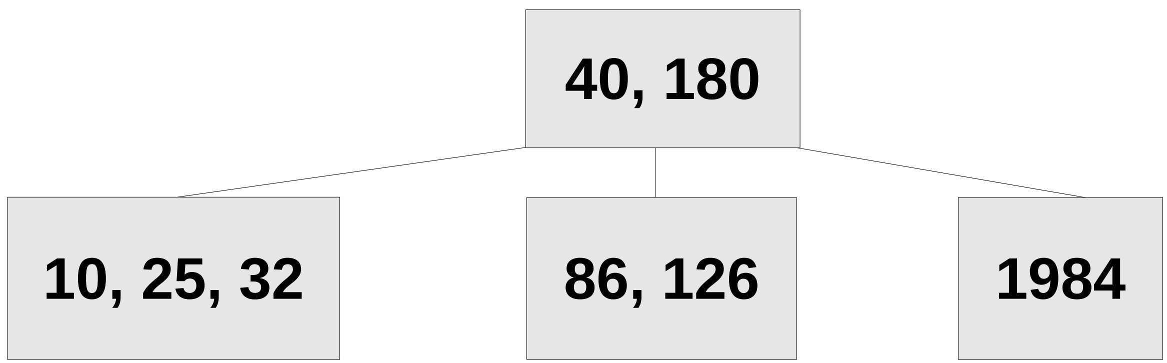
# 2-4 Tree Insertion

(Let's play the game!)



# 2-4 Tree Insertion

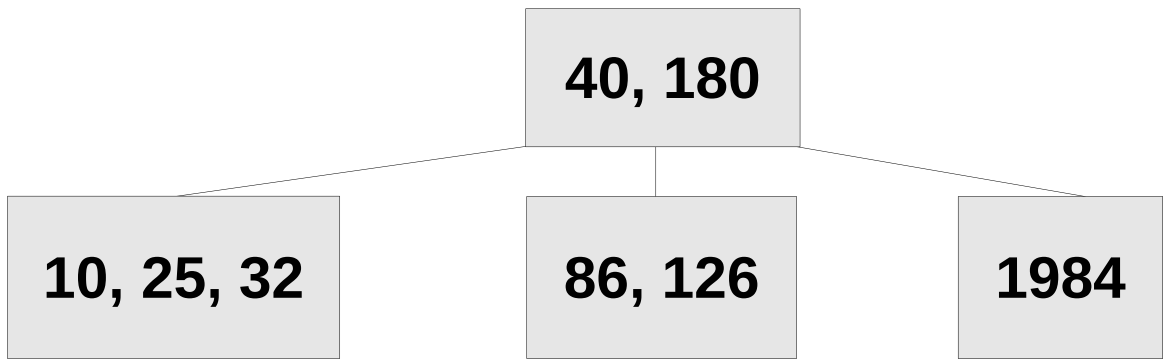
(Let’s play the game!)




*tada!*

# 2-4 Tree Insertion

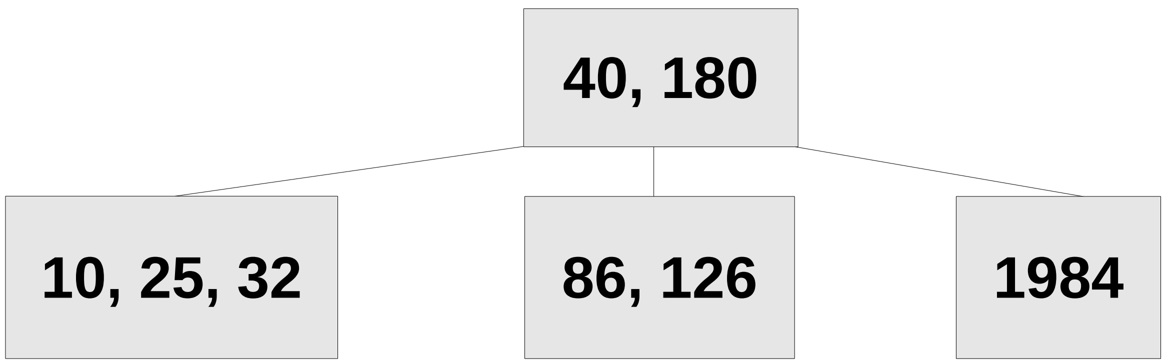
(Let’s play the game!)




Let’s continue inserting...

# 2-4 Tree Insertion

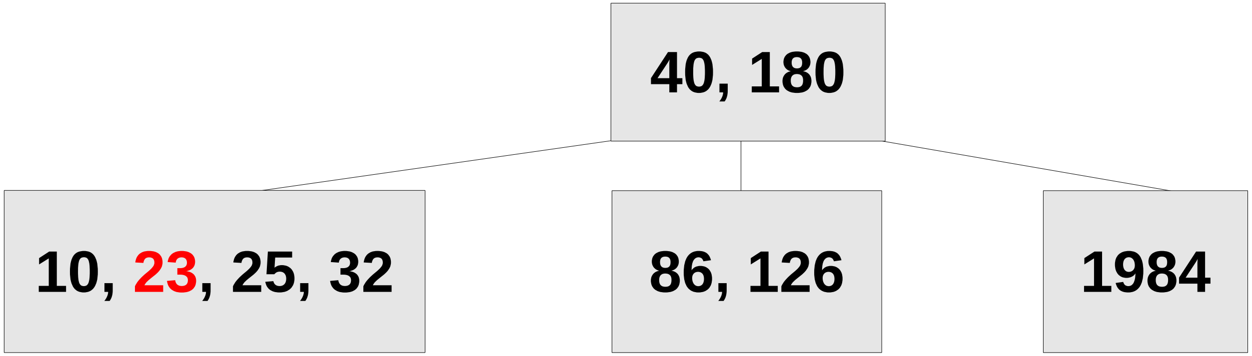
(Let’s play the game!)




Insert 23

# 2-4 Tree Insertion

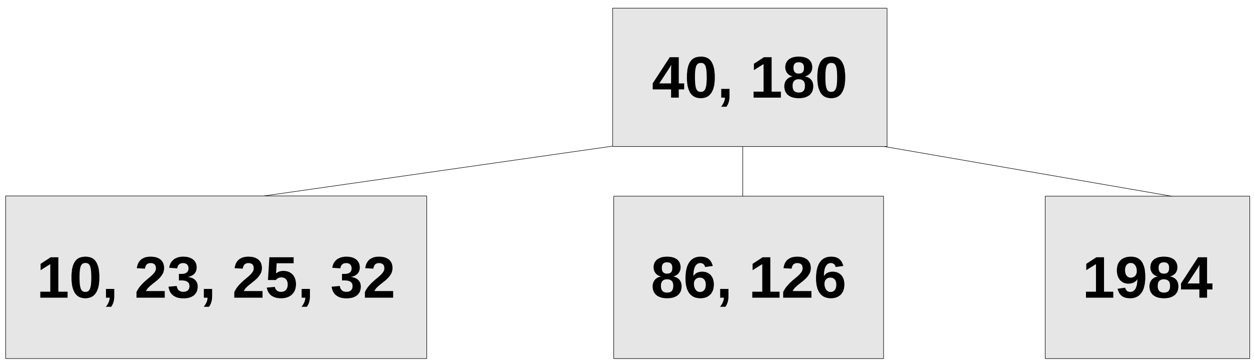
(Let’s play the game!)




Insert 23

# 2-4 Tree Insertion

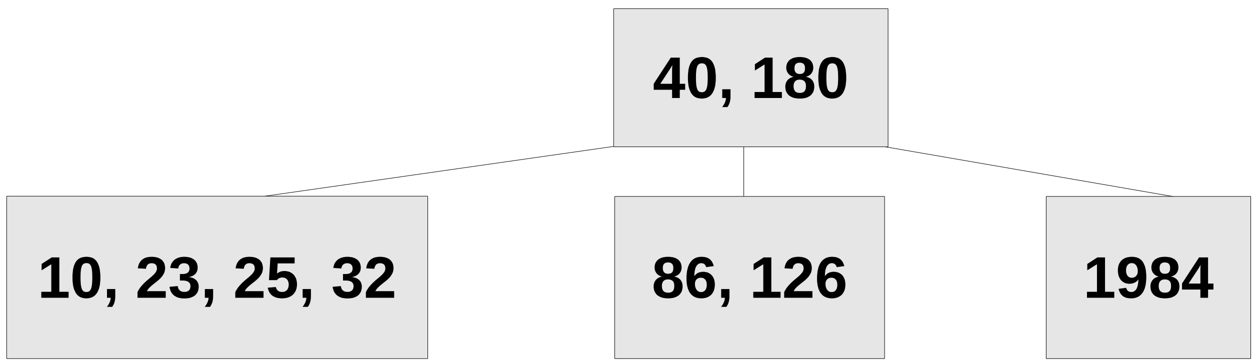
(Let’s play the game!)




25 gets squished up.

# 2-4 Tree Insertion

(Let’s play the game!)

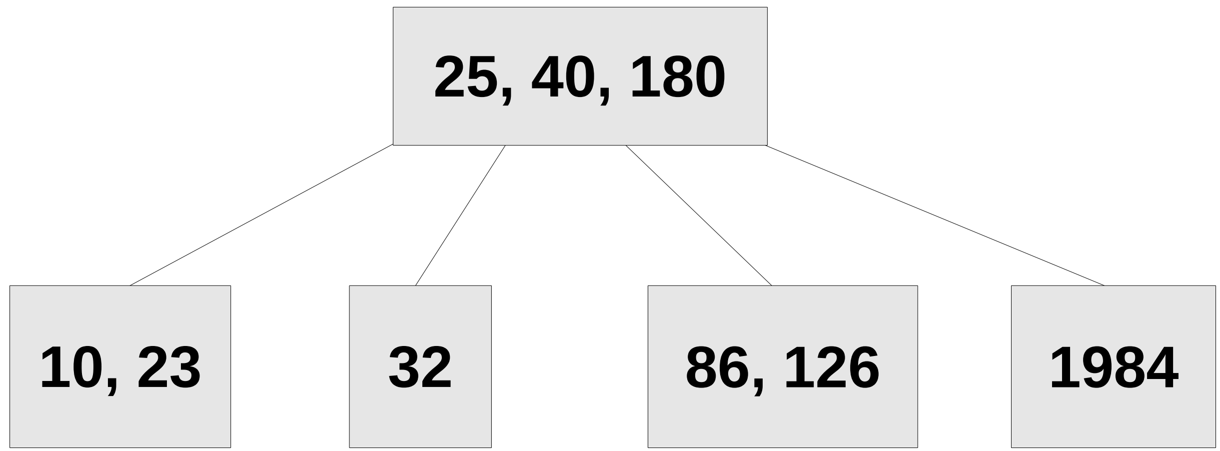



Let’s skip the theatrics for the sake of time.

Do you see how the structure will change?

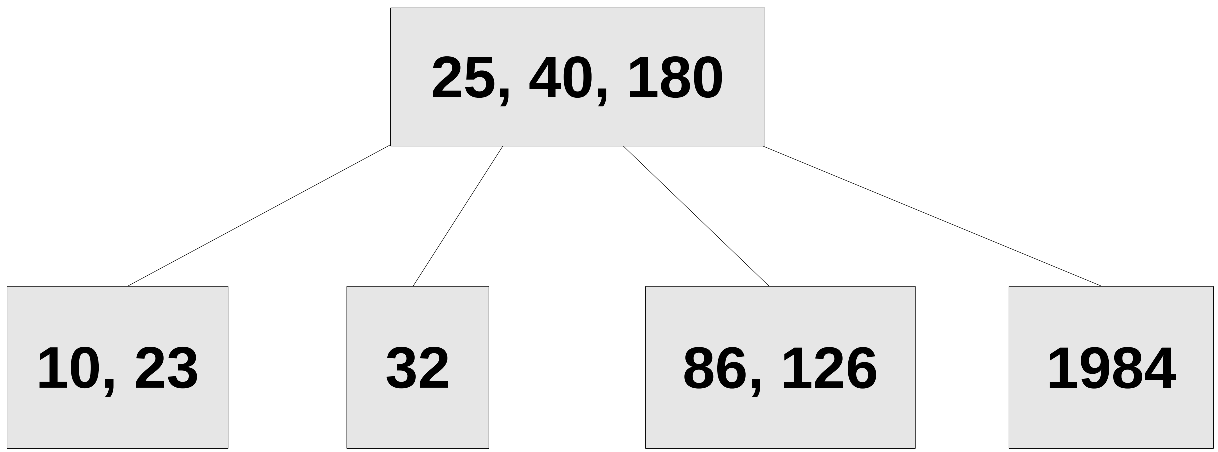
# 2-4 Tree Insertion

(Let’s play the game!)



# 2-4 Tree Insertion

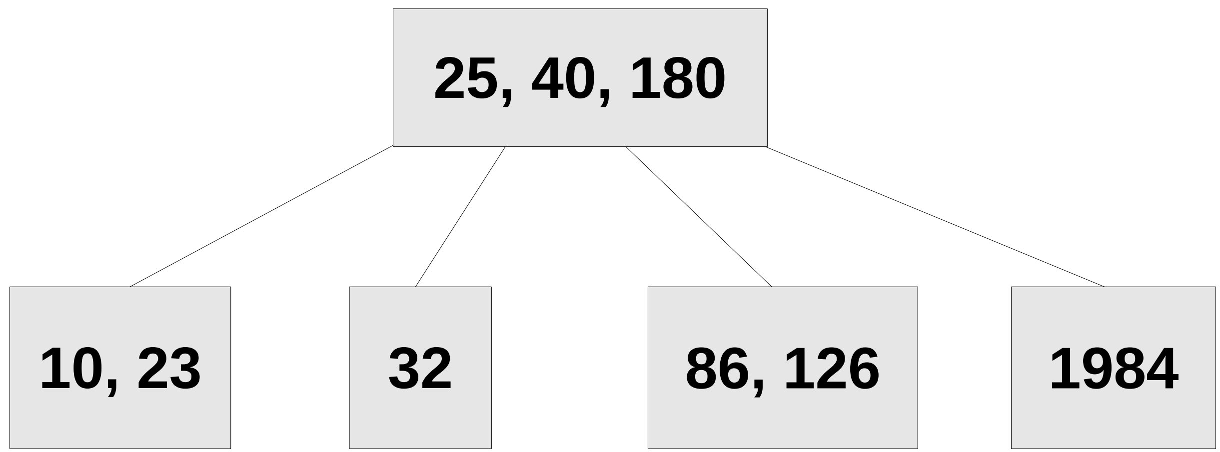
(Let’s play the game!)




Just two more insertions to make
this thing bust...

# 2-4 Tree Insertion

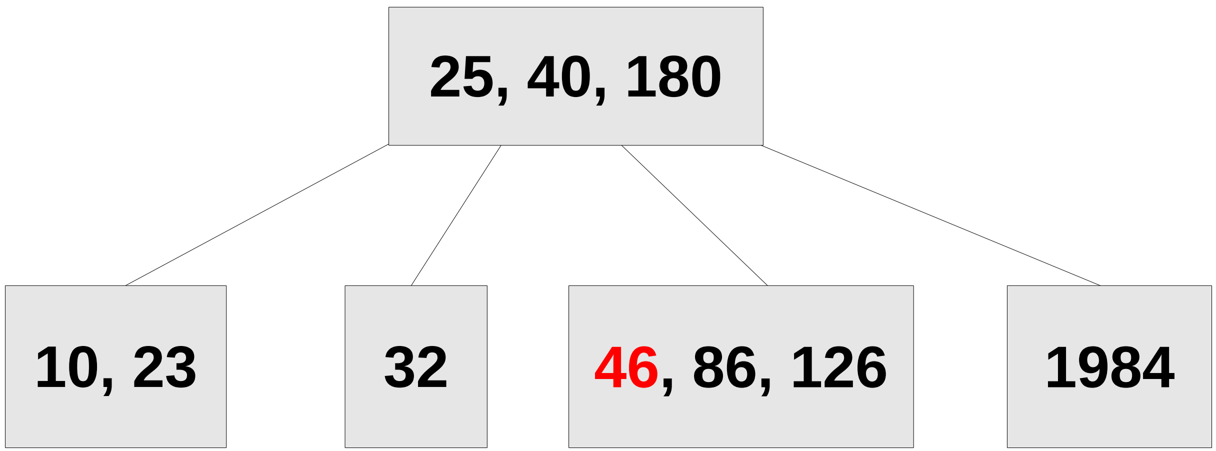
(Let’s play the game!)




Insert 46

# 2-4 Tree Insertion

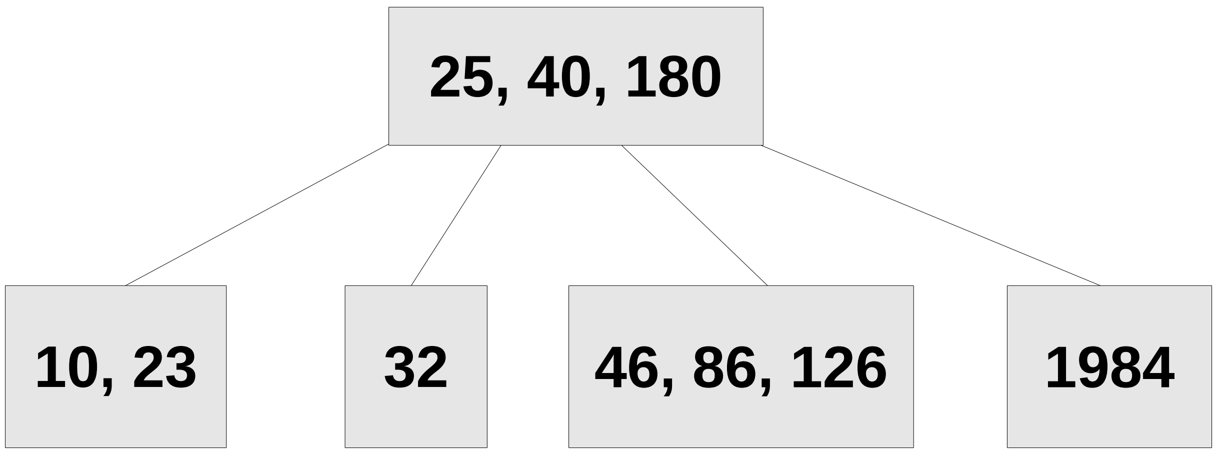
(Let’s play the game!)




Insert 46

# 2-4 Tree Insertion

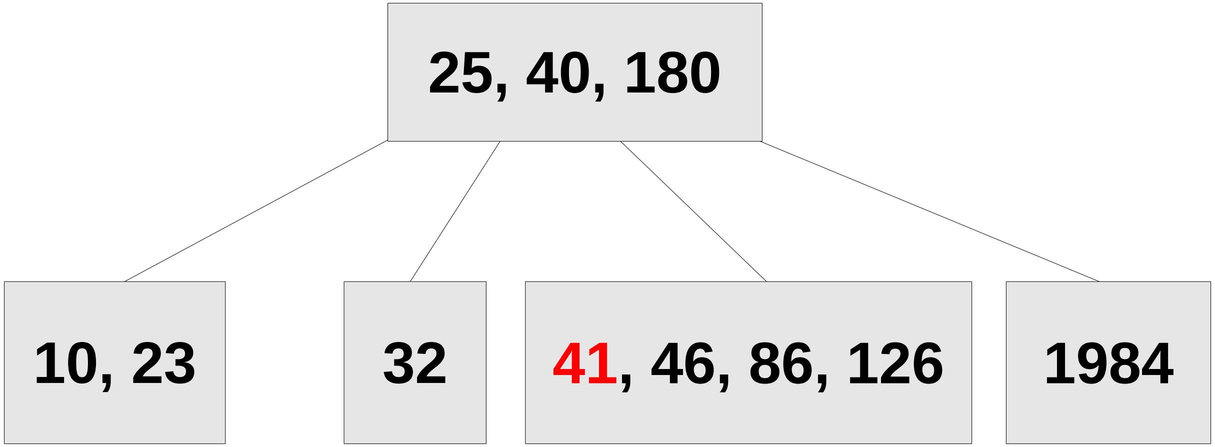
(Let’s play the game!)




Insert 41

# 2-4 Tree Insertion

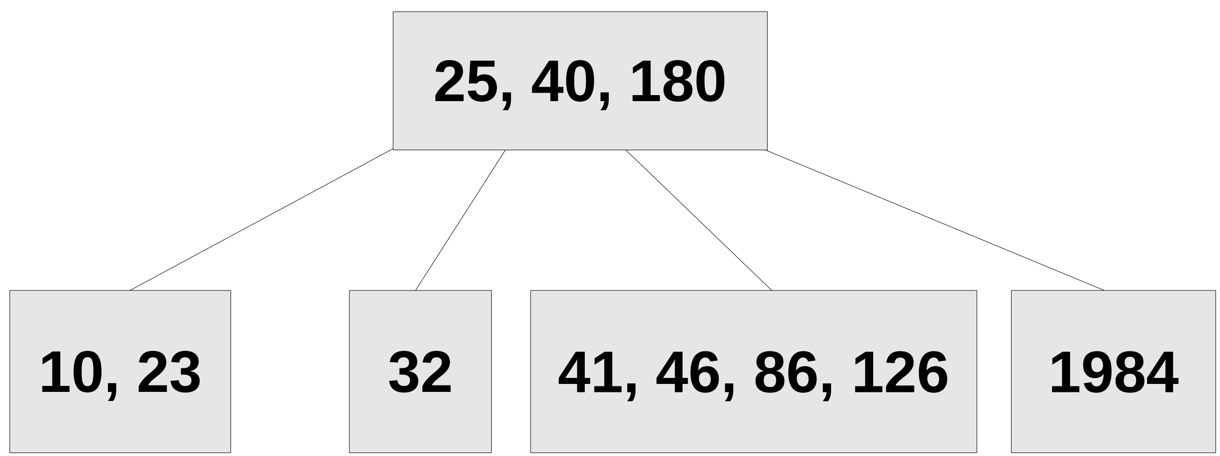
(Let’s play the game!)




Insert 41

# 2-4 Tree Insertion

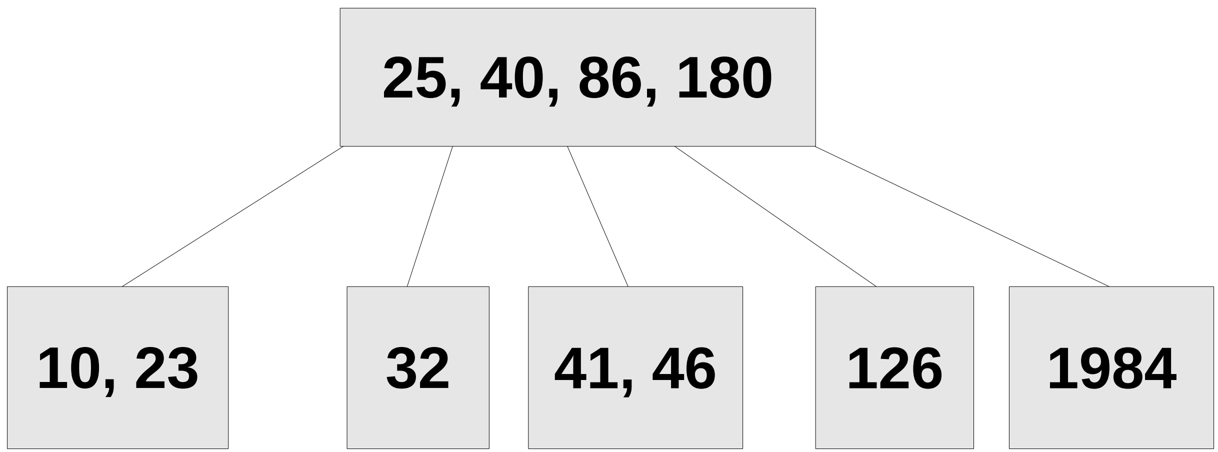
(Let’s play the game!)




86 gets squished up

# 2-4 Tree Insertion

(Let's play the game!)

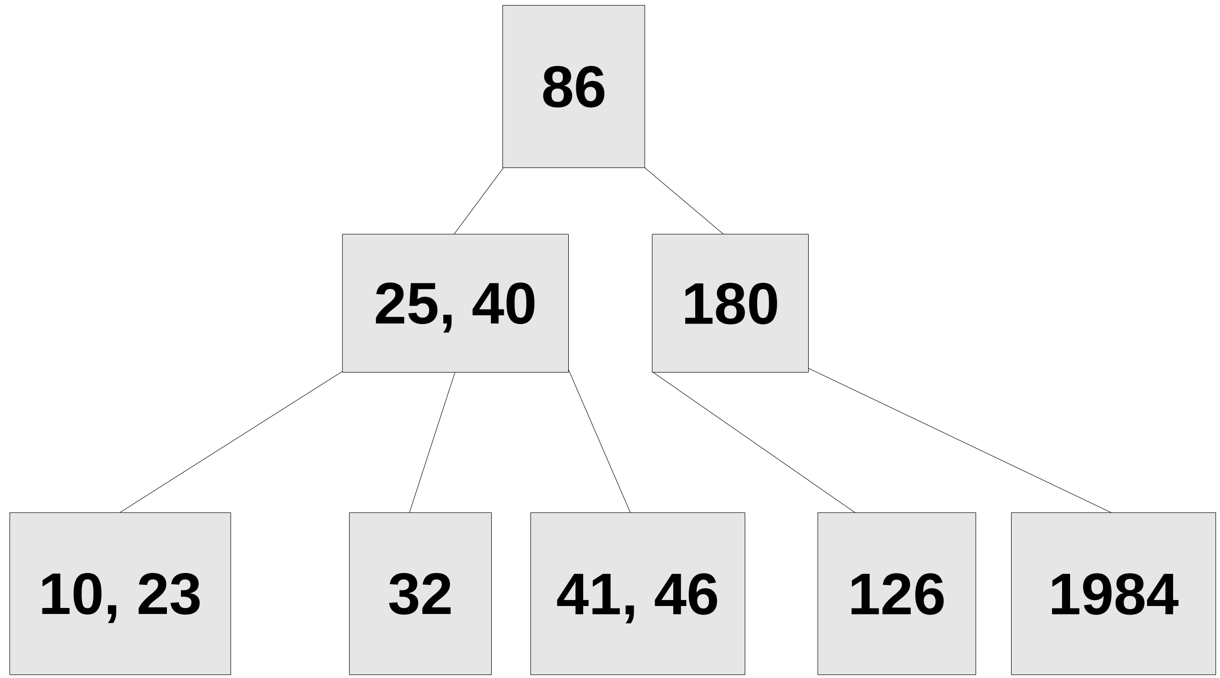



But now the root node has an overflow!

86 gets squished up again.

# 2-4 Tree Insertion

(Let’s play the game!)

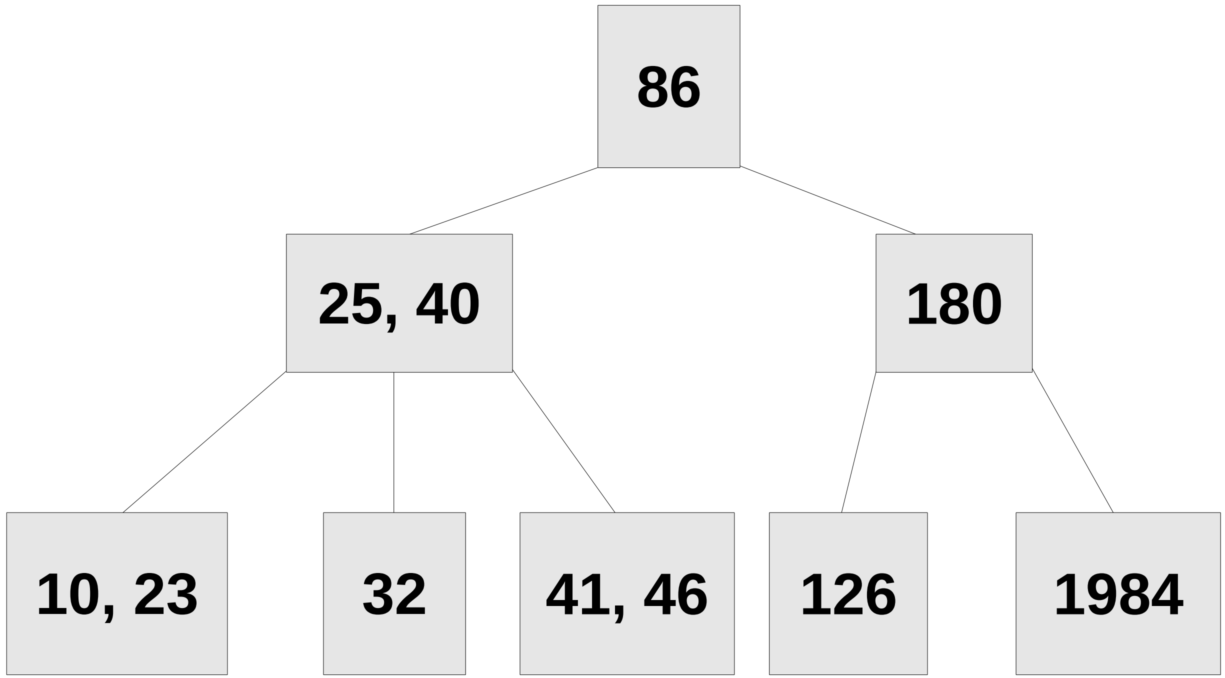



But now the root node has an overflow!

86 gets squished up again.

# 2-4 Tree Insertion

(Let's play the game!)



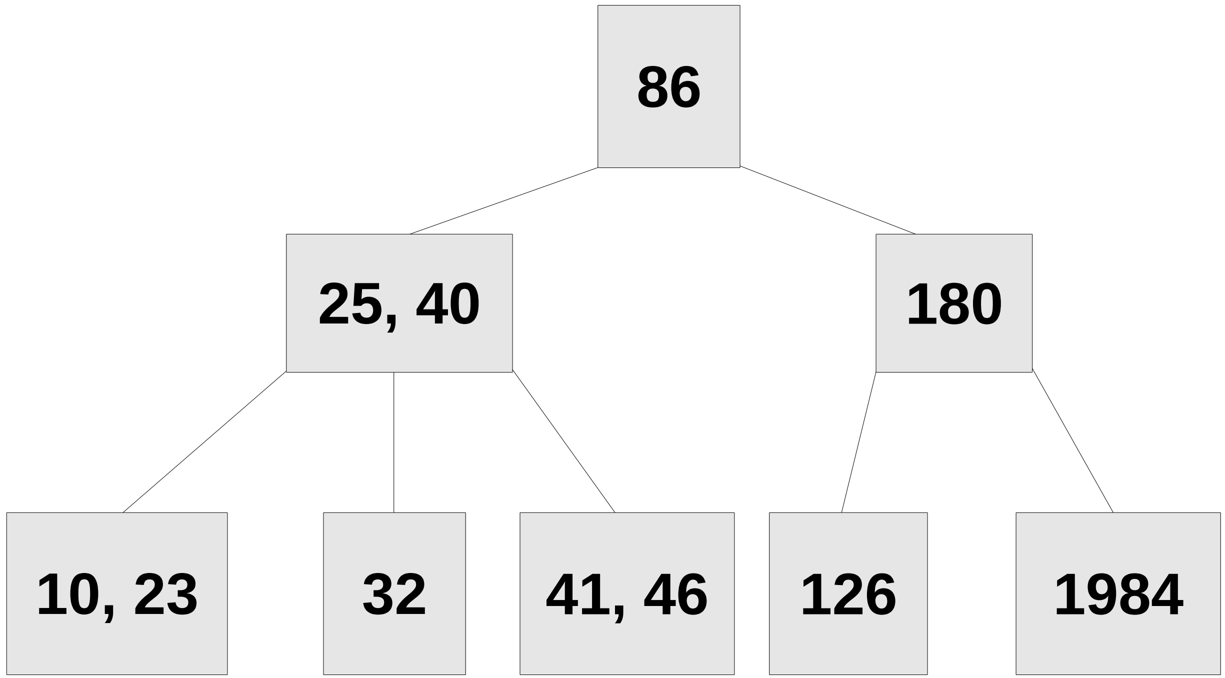

But now the root node has an overflow!

86 gets squished up again.

Let’s take a look at deletion now.

# 2-4 Tree Deletion

(this is going to get crazy)



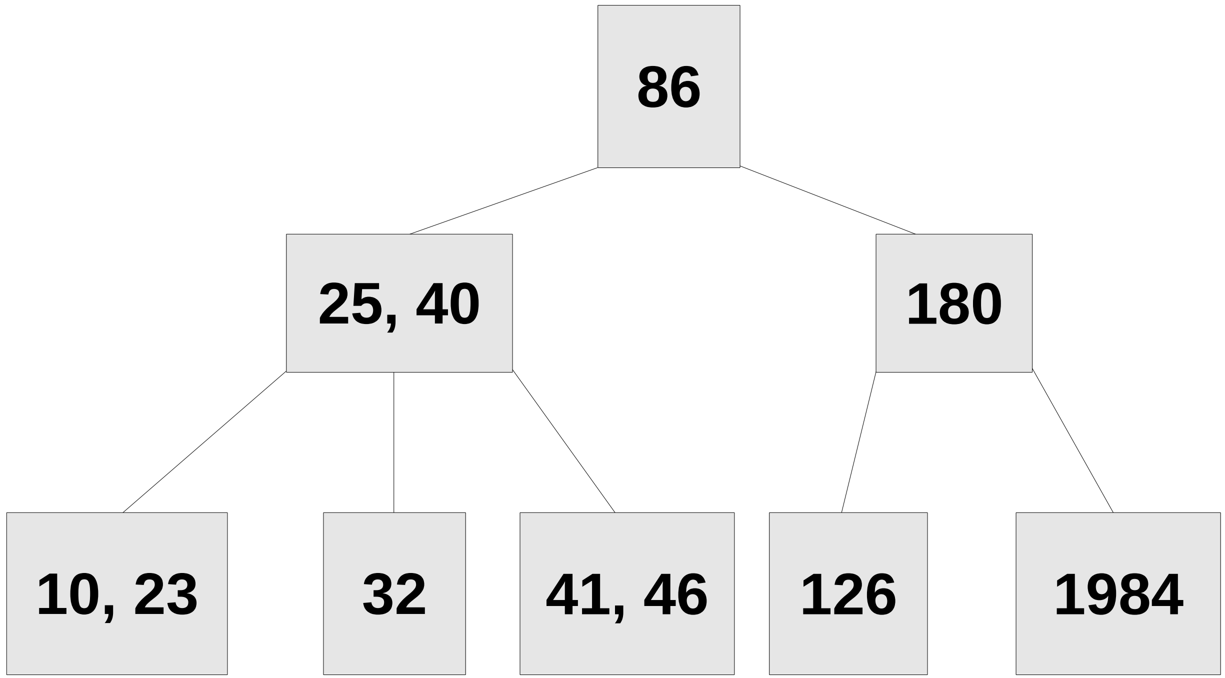

If we delete a non-leaf node, we do it BST style:

replace with largest value in left subtree
*--or--*
replace with smallest value in right subtree

But that means we're always bring up a value from a leaf node!

# 2-4 Tree Deletion

(this is going to get crazy)

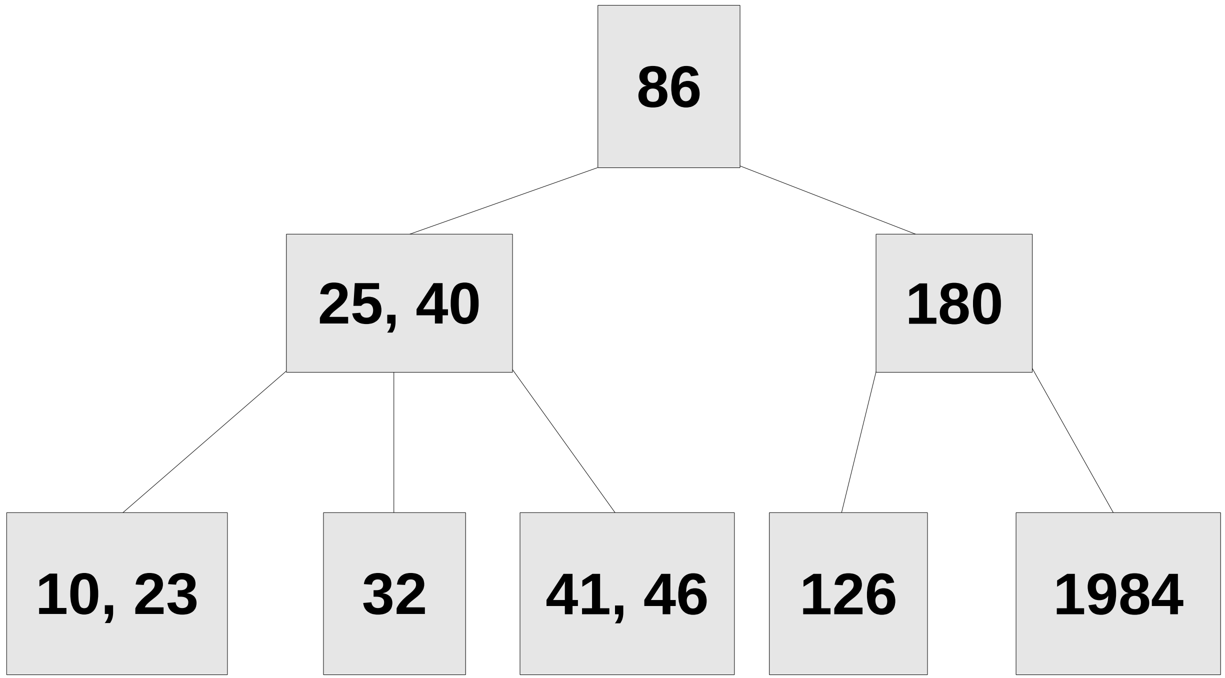



So, leaf node deletions are the only interesting deletions.

Let’s take a look...

# 2-4 Tree Deletion

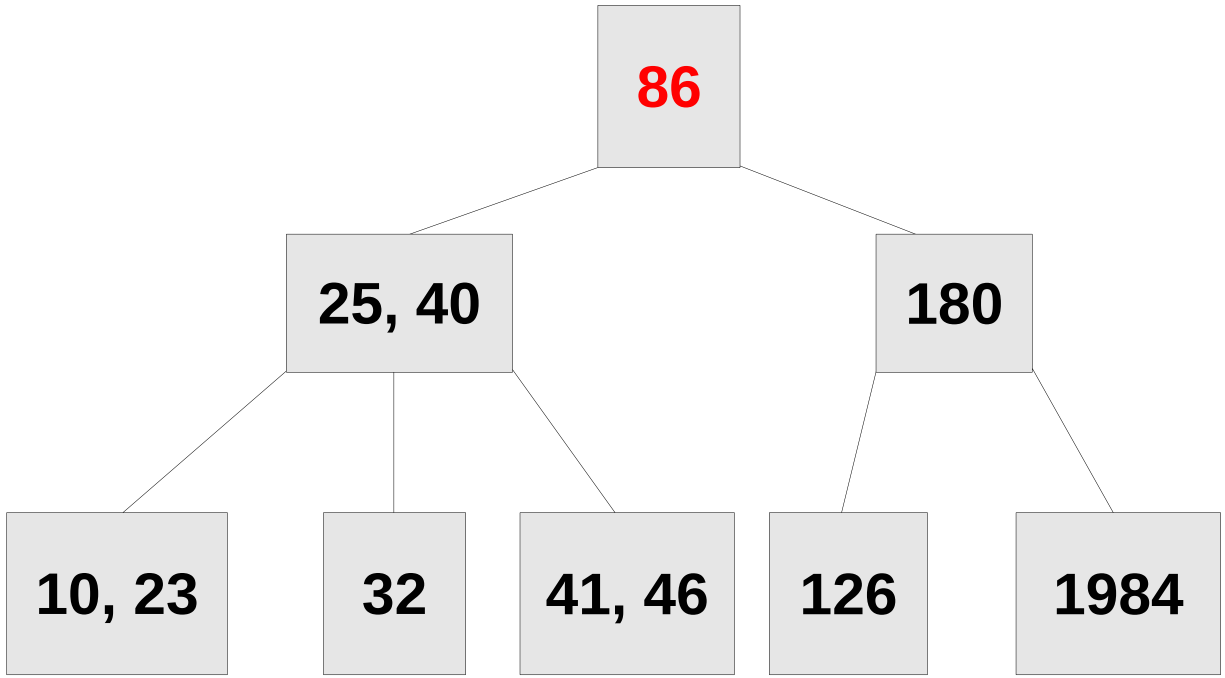
(this is going to get crazy)




Delete 86

# 2-4 Tree Deletion

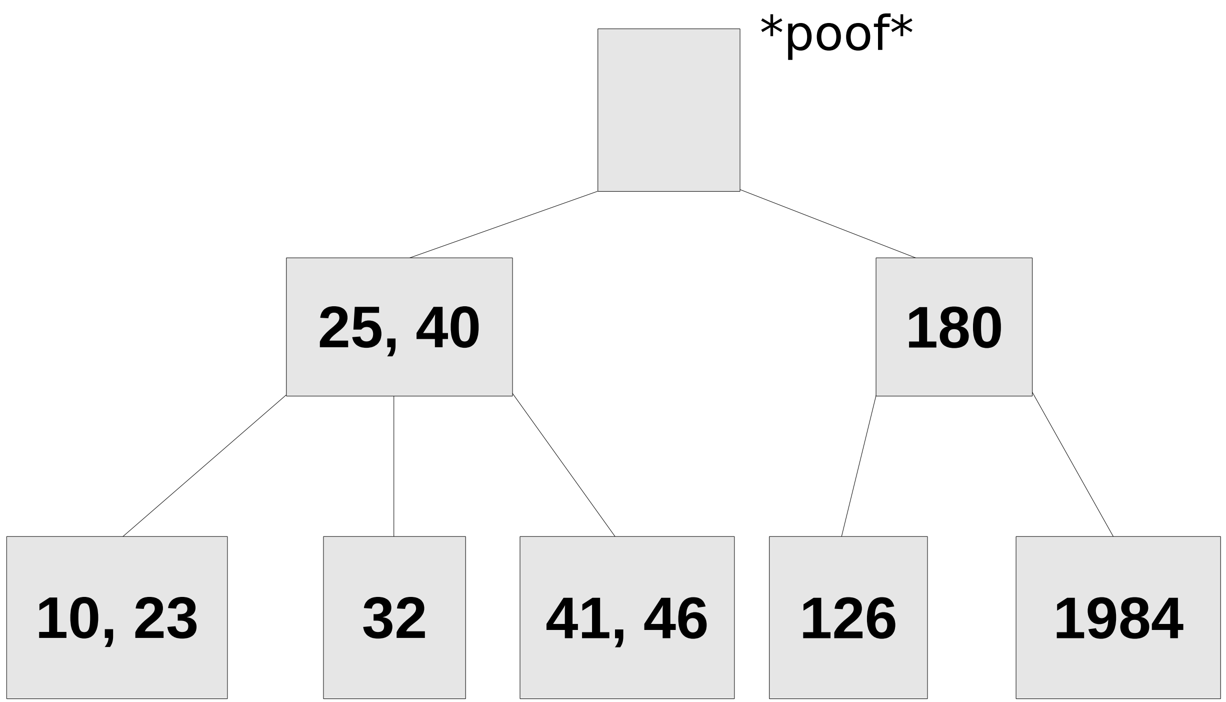
(this is going to get crazy)




Delete 86

# 2-4 Tree Deletion

(this is going to get crazy)

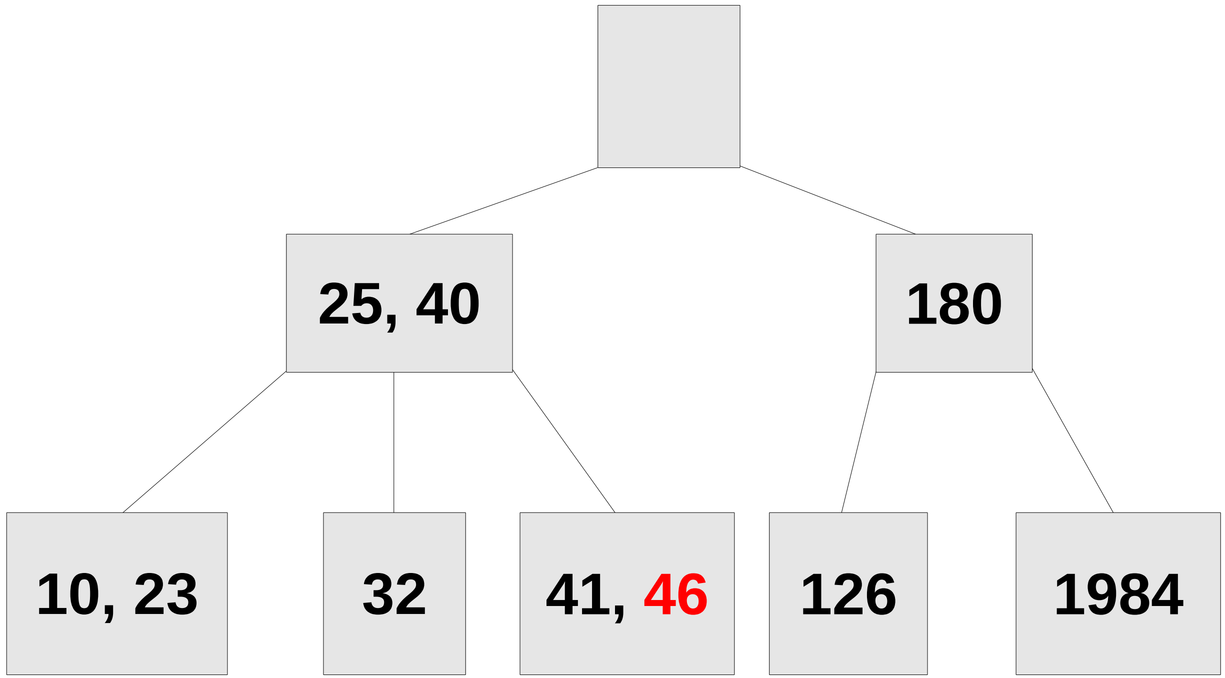



Delete 86

Which element moves up?

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 86

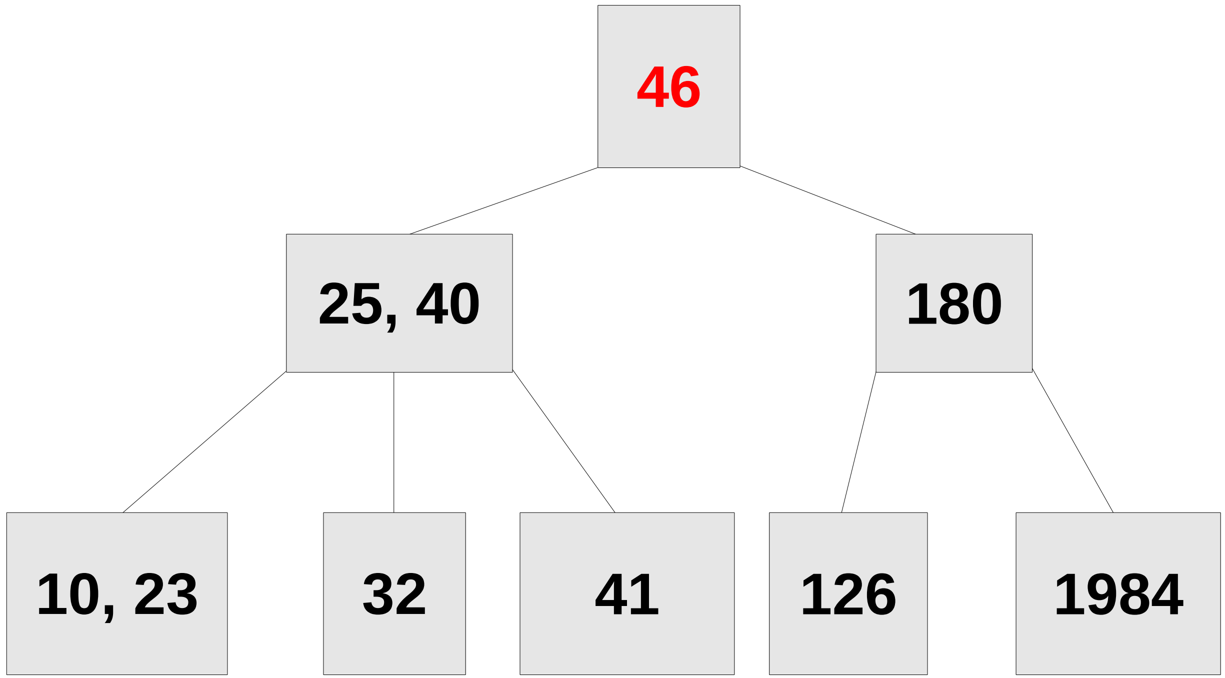
Which element moves up?

If we grab an element from a leaf node with multiple elements, we’re okay:

No change in leaf node heights. No children to worry about.

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 86

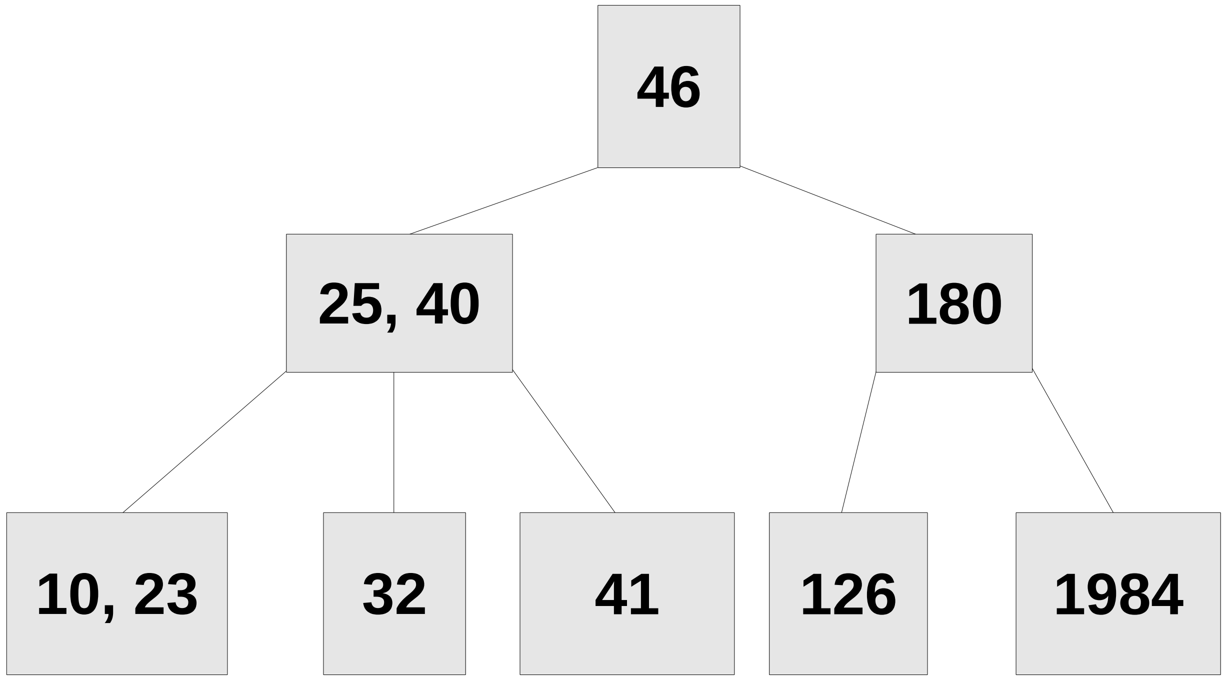
Which element moves up?

If we grab an element from a leaf node with multiple elements, we're okay:

No change in leaf node heights. No children to worry about.

# 2-4 Tree Deletion

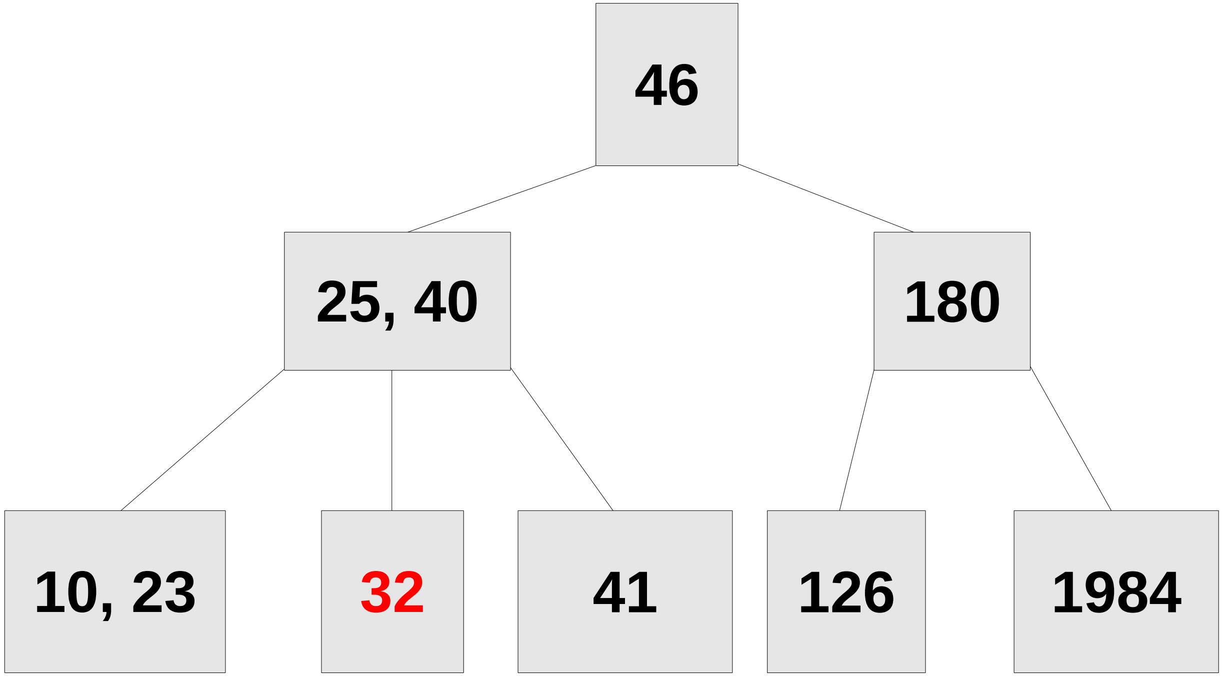
(this is going to get crazy)




Delete 32

# 2-4 Tree Deletion

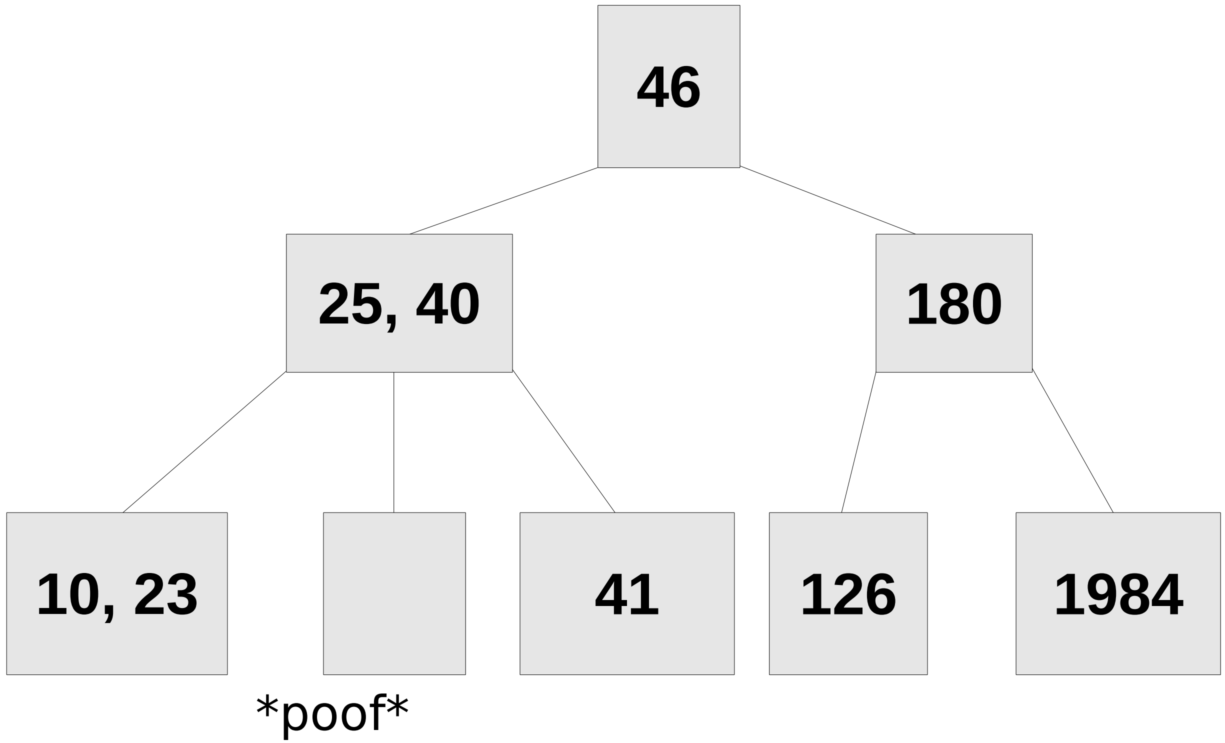
(this is going to get crazy)




Delete 32

# 2-4 Tree Deletion

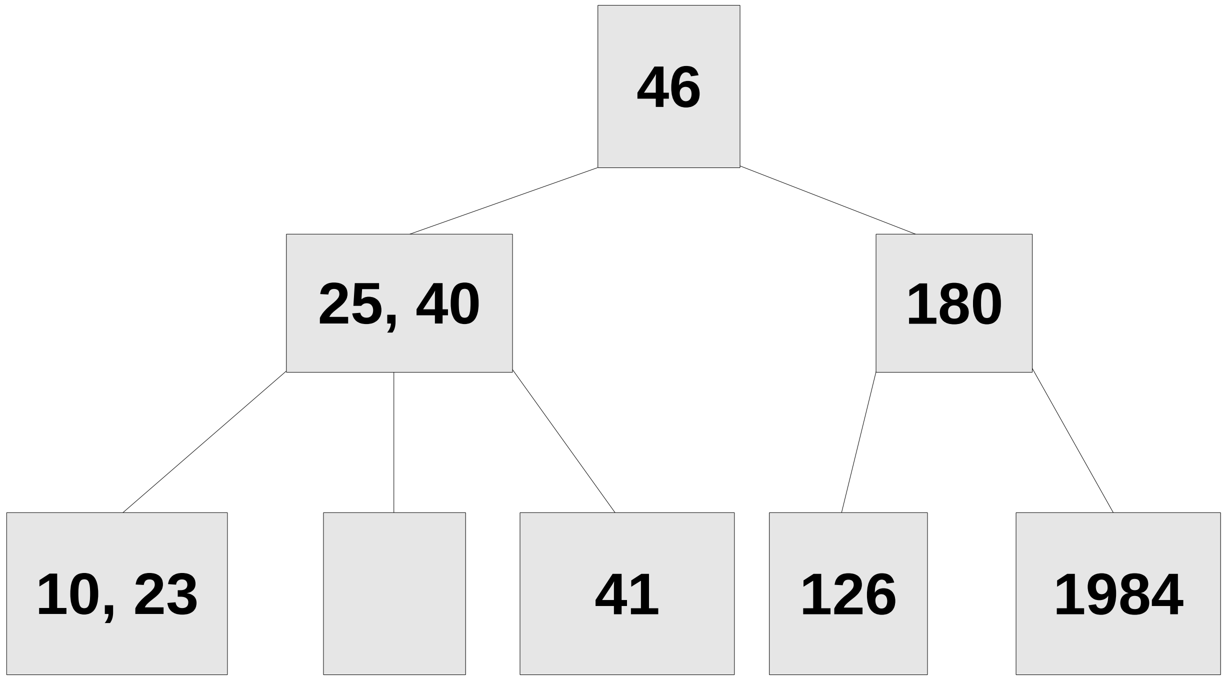
(this is going to get crazy)




Delete 32

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 32

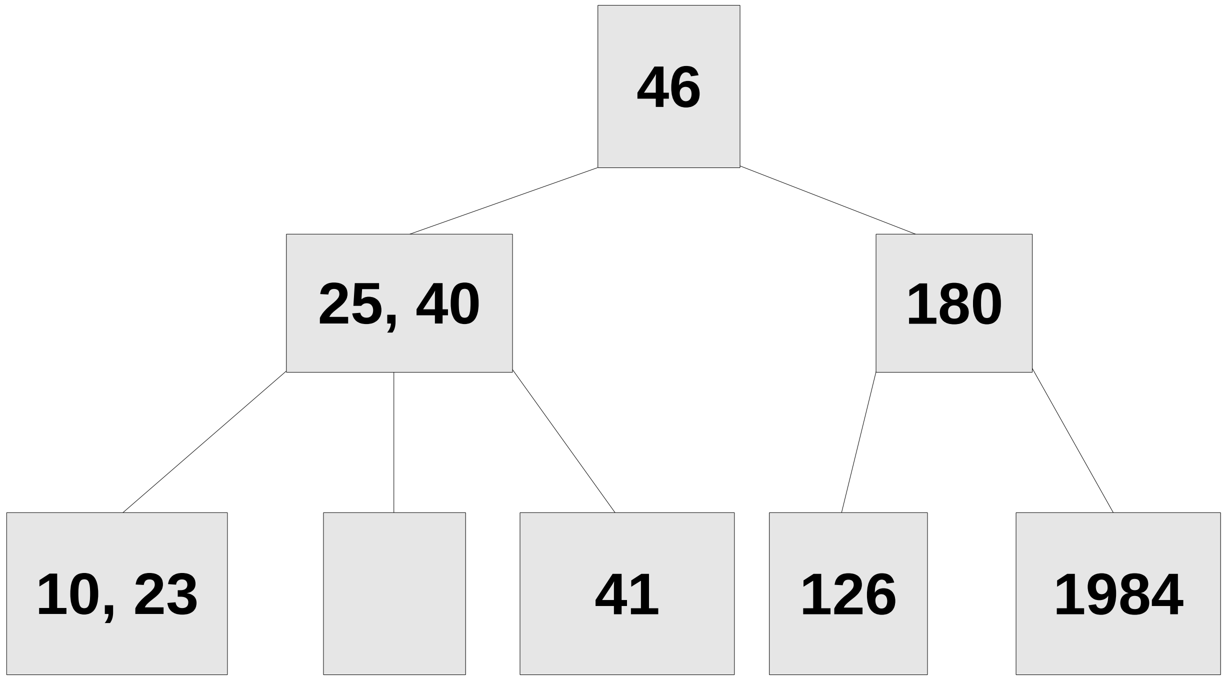
We can’t eliminate that node, though! It won’t be a valid 2-4 tree if we do.

Instead, we go begging for help from our siblings.

First look left, then look right. Our left sibling has some elements to spare.

# 2-4 Tree Deletion

(this is going to get crazy)



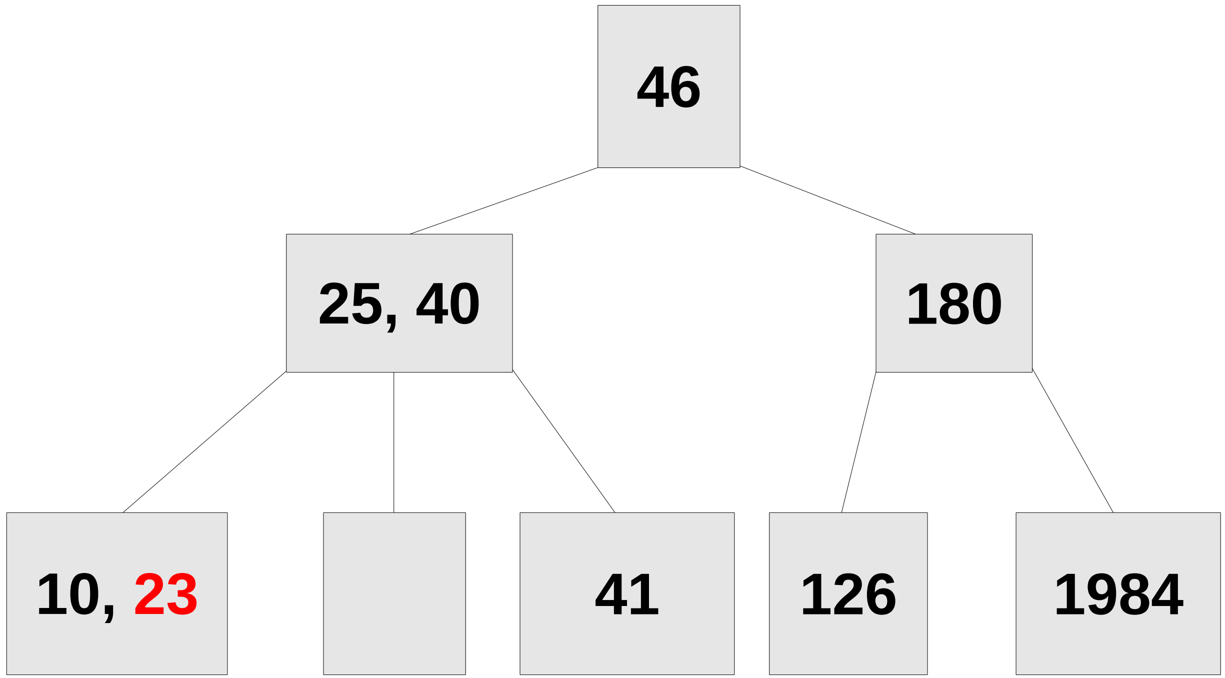

Delete 32

We can't just move 23 over. That would violate our ordering property.

Instead, 23 moves up and knocks its parent down.

# 2-4 Tree Deletion

(this is going to get crazy)



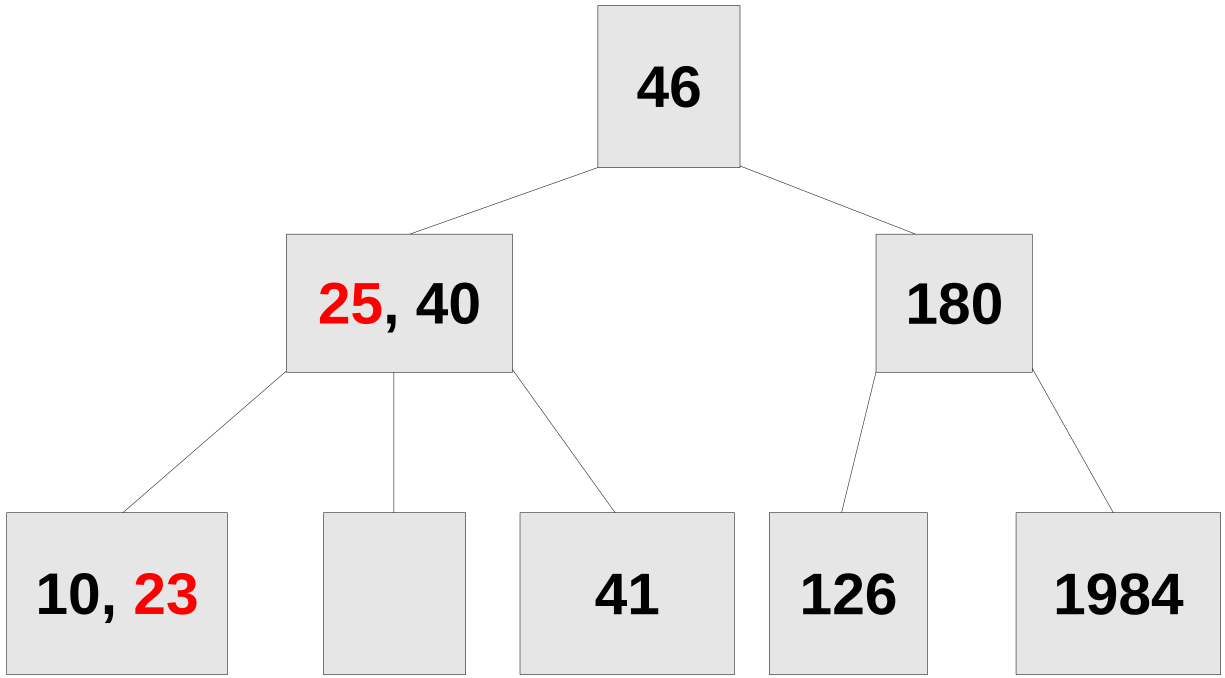

Delete 32

We can’t just move 23 over. That would violate our ordering property.

Instead, 23 moves up and knocks its parent down.

# 2-4 Tree Deletion

(this is going to get crazy)



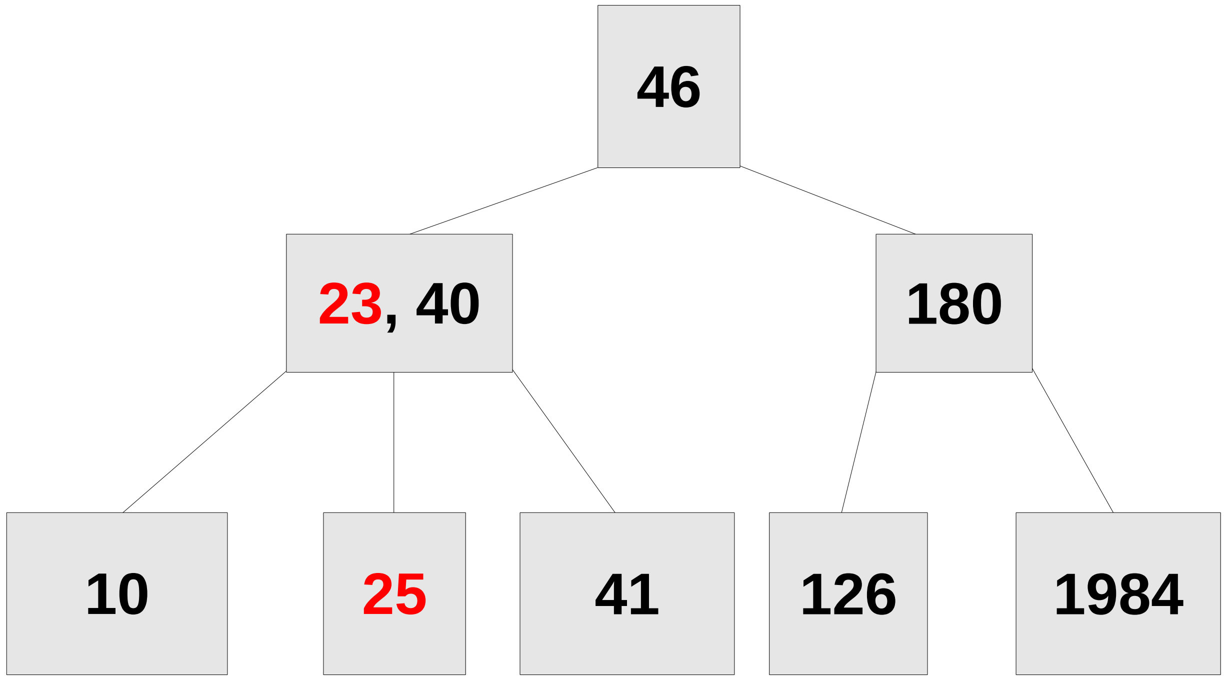

Delete 32

We can't just move 23 over. That would violate our ordering property.

Instead, 23 moves up and knocks its parent down.

# 2-4 Tree Deletion

(this is going to get crazy)



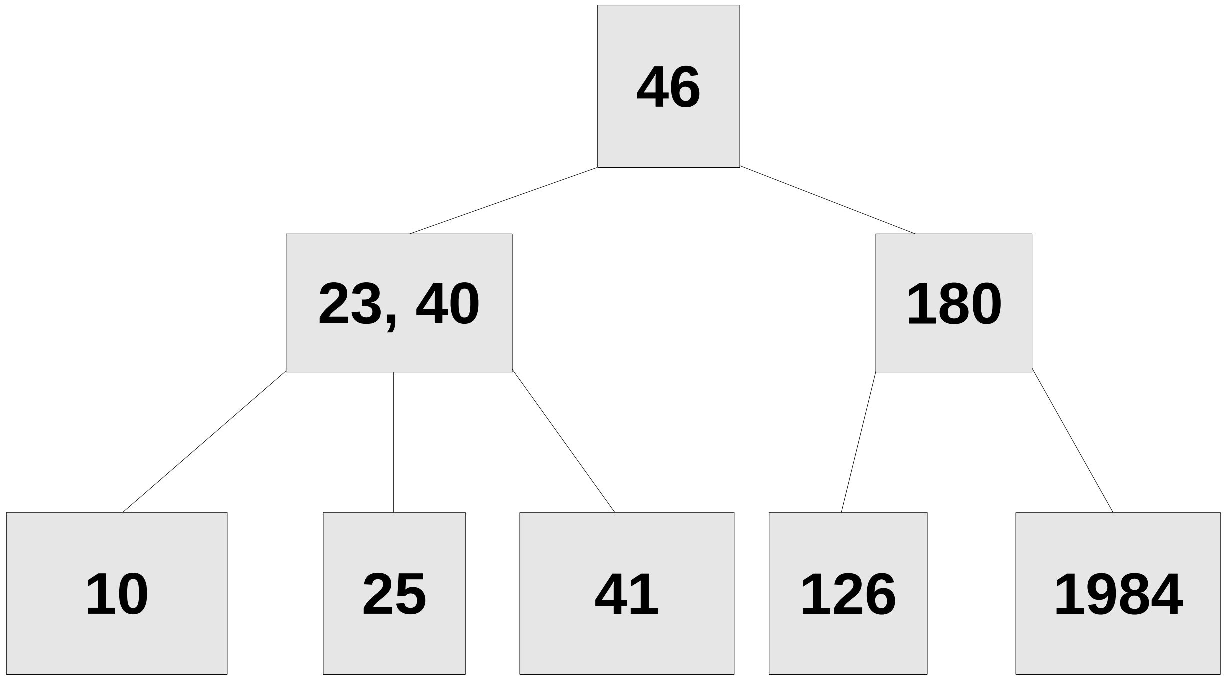

Delete 32

We can’t just move 23 over. That would violate our ordering property.

Instead, 23 moves up and knocks its parent down.

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 32

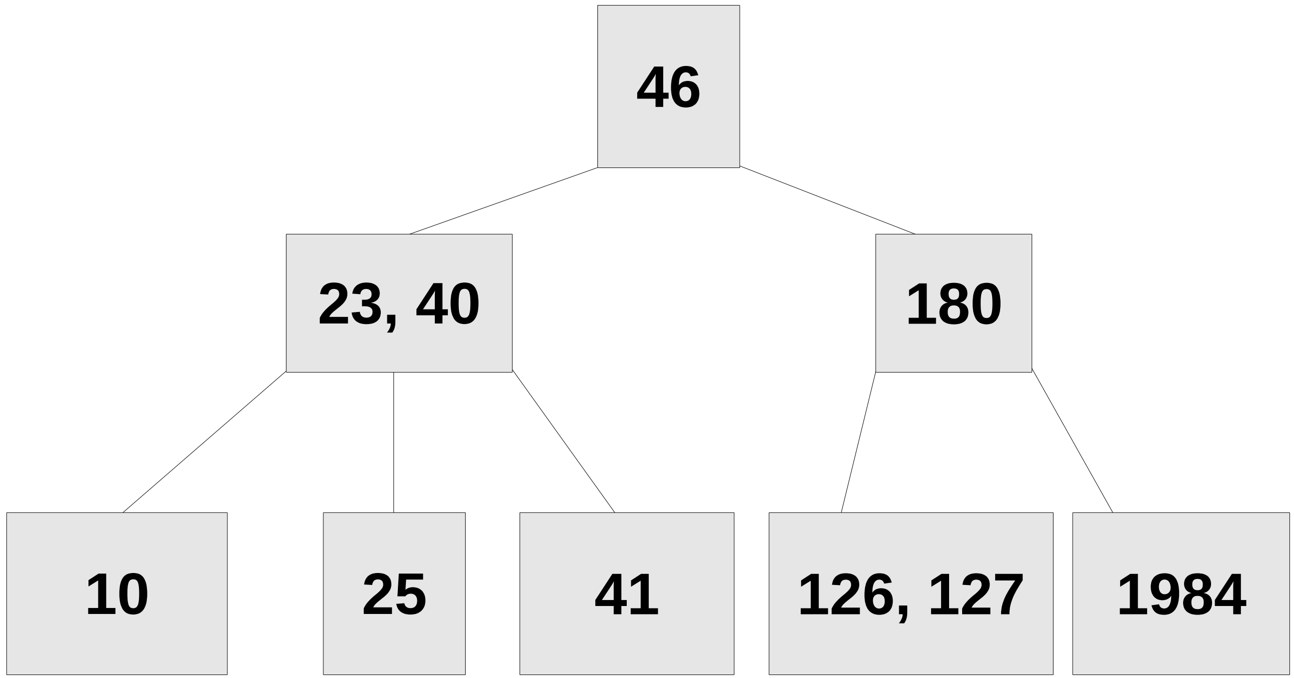
We can’t just move 23 over. That would violate our ordering property.

Instead, 23 moves up and knocks its parent down.

Tada!

# 2-4 Tree Deletion

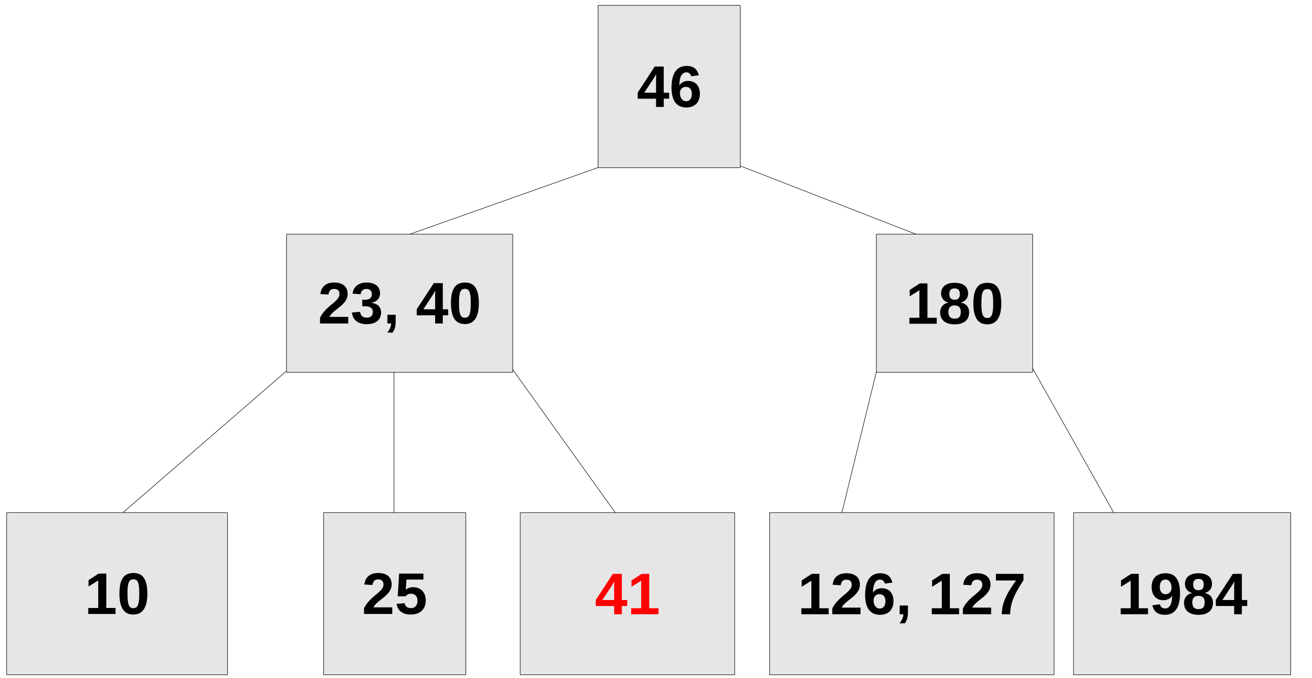
(this is going to get crazy)




Delete 41

# 2-4 Tree Deletion

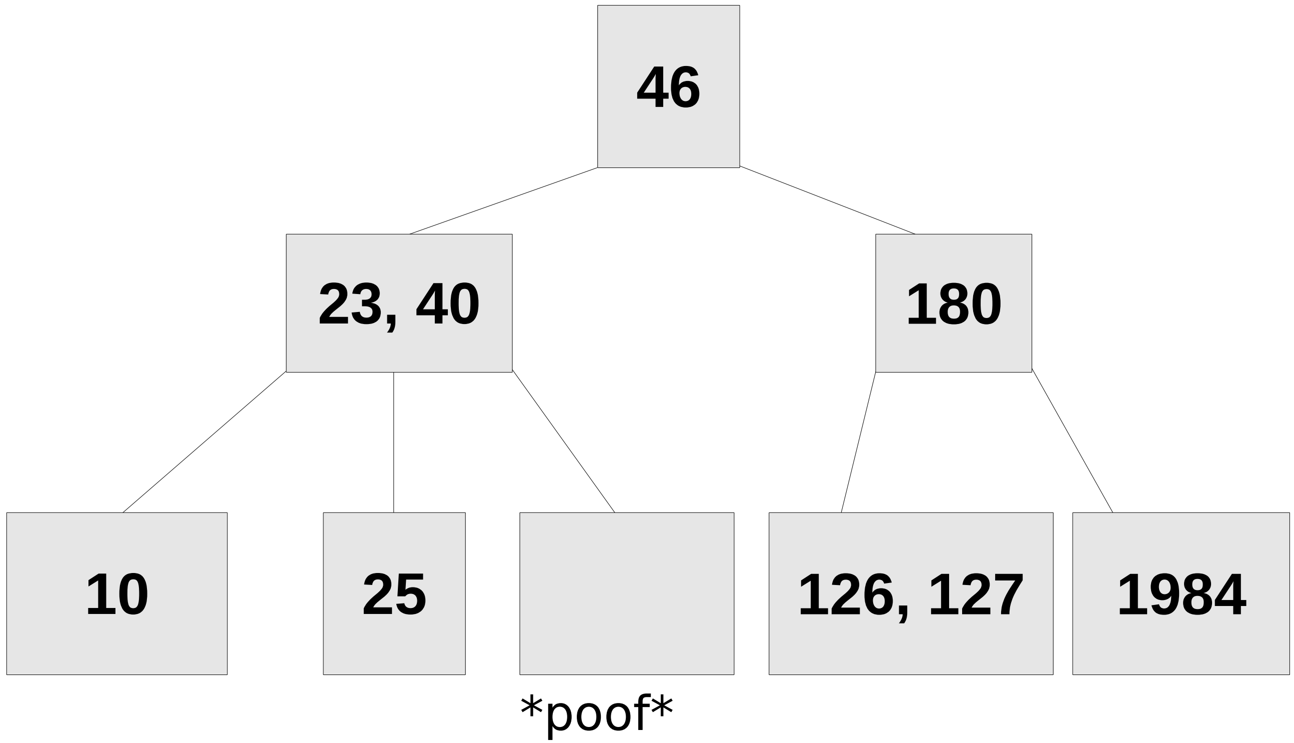
(this is going to get crazy)




Delete 41

# 2-4 Tree Deletion

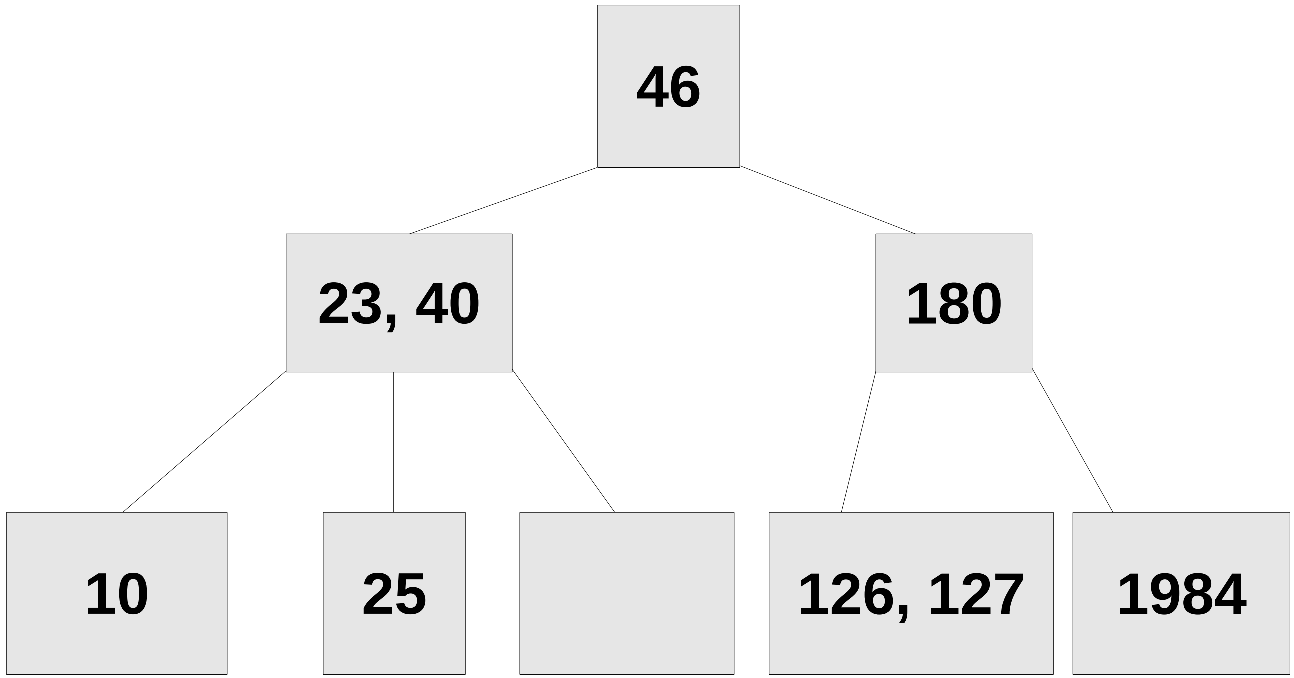
(this is going to get crazy)




Delete 41

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 41

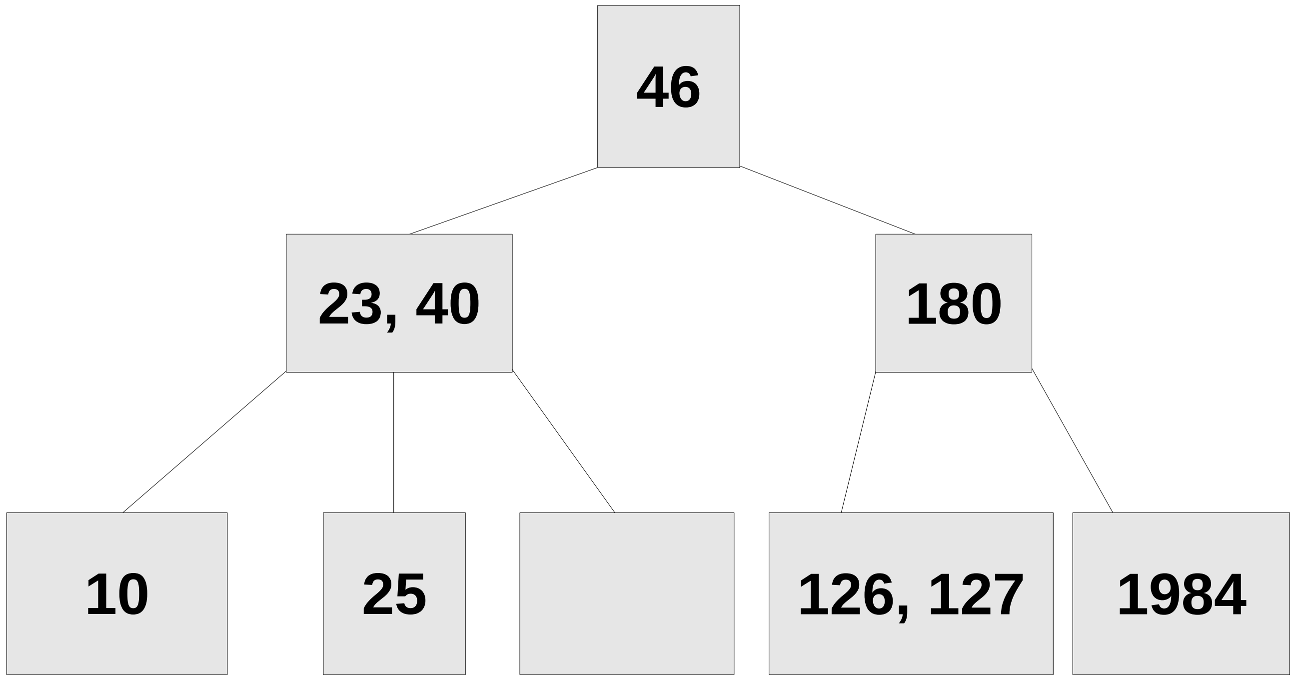
But the empty node has no siblings with extra elements!

Note: (126, 127) is not a sibling of that empty node.

(They have different parents!)

# 2-4 Tree Deletion

(this is going to get crazy)



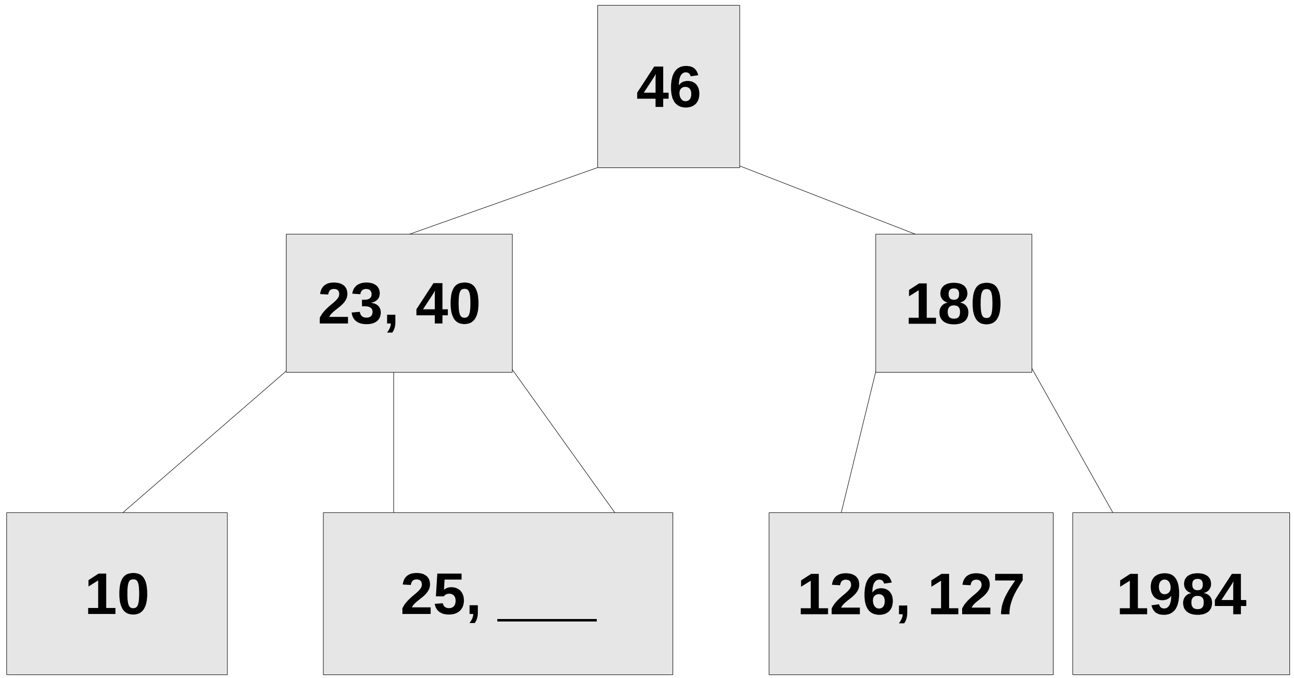

Delete 41

THIS IS A CRISIS. But we have a solution: fuse and drop.

First, fuse the empty node with its left sibling (or right sibling, if it has no left sibling).

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 41

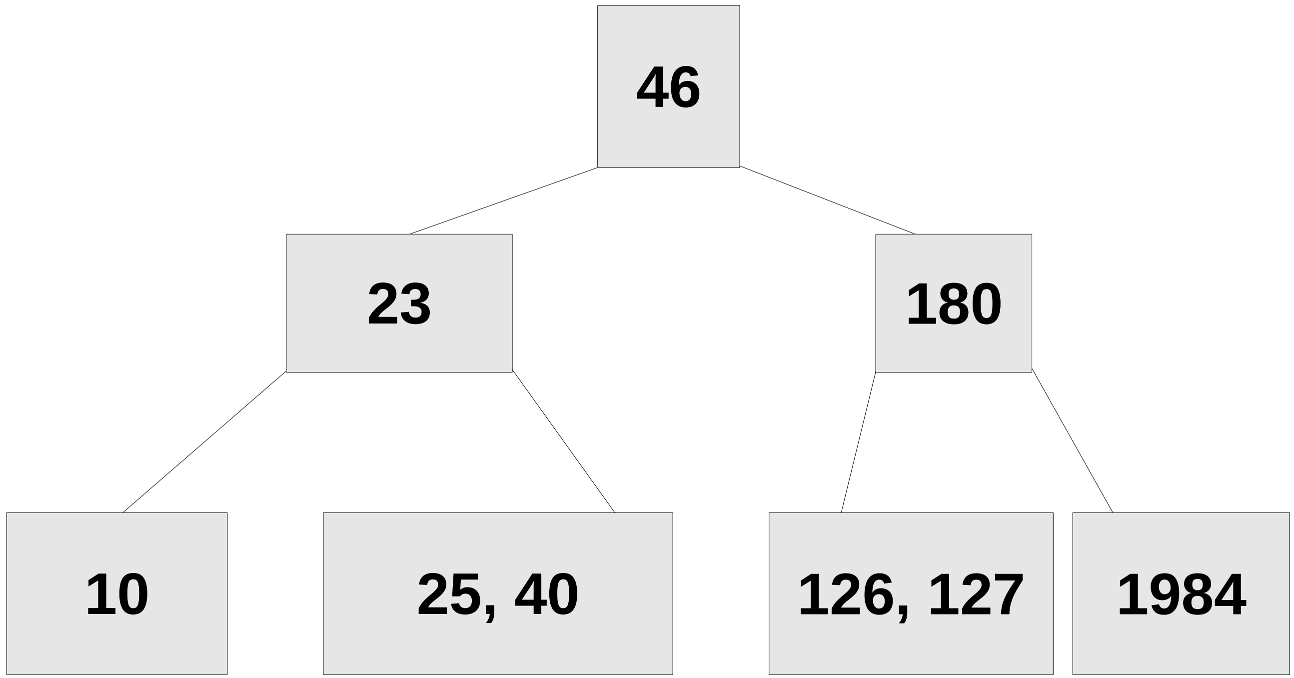
THIS IS A CRISIS. But we have a solution: fuse and drop.

First, fuse the empty node with its left sibling (or right sibling, if it has no left sibling).

Now drop down the parent that was in between the newly fused nodes.

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 41

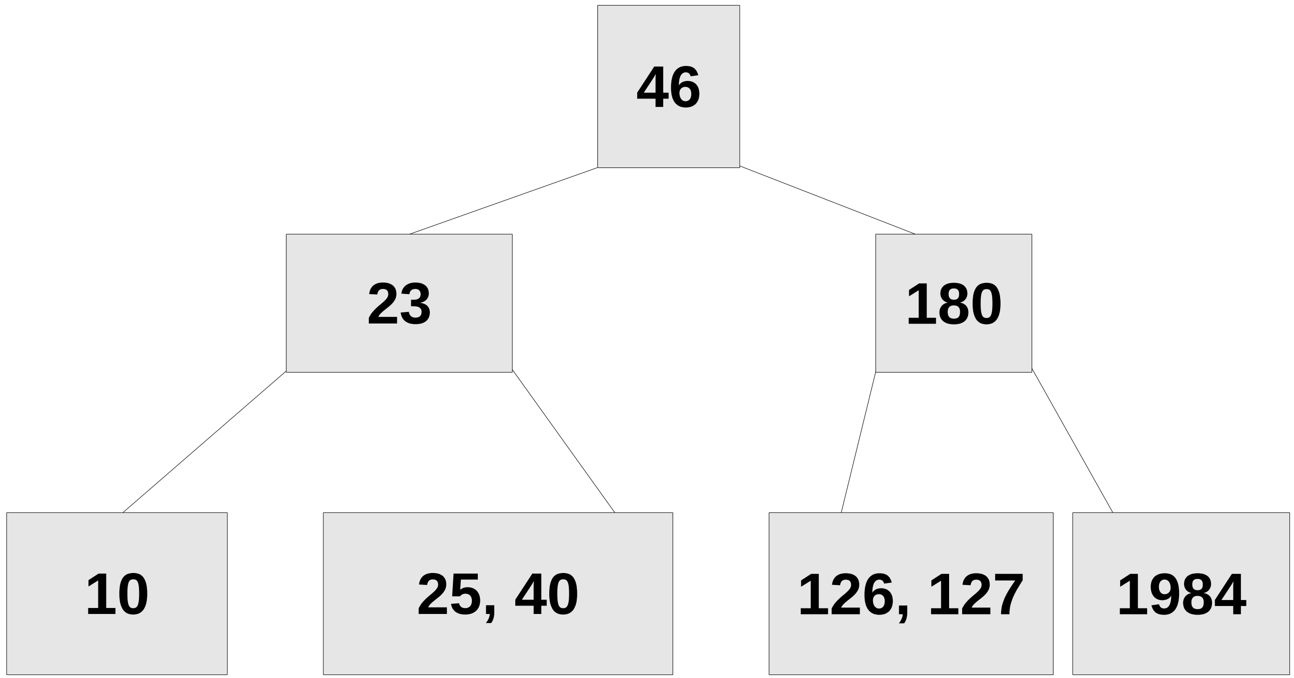
THIS IS A CRISIS. But we have a solution: fuse and drop.

First, fuse the empty node with its left sibling (or right sibling, if it has no left sibling).

Now drop down the parent that was in between the newly fused nodes.

# 2-4 Tree Deletion

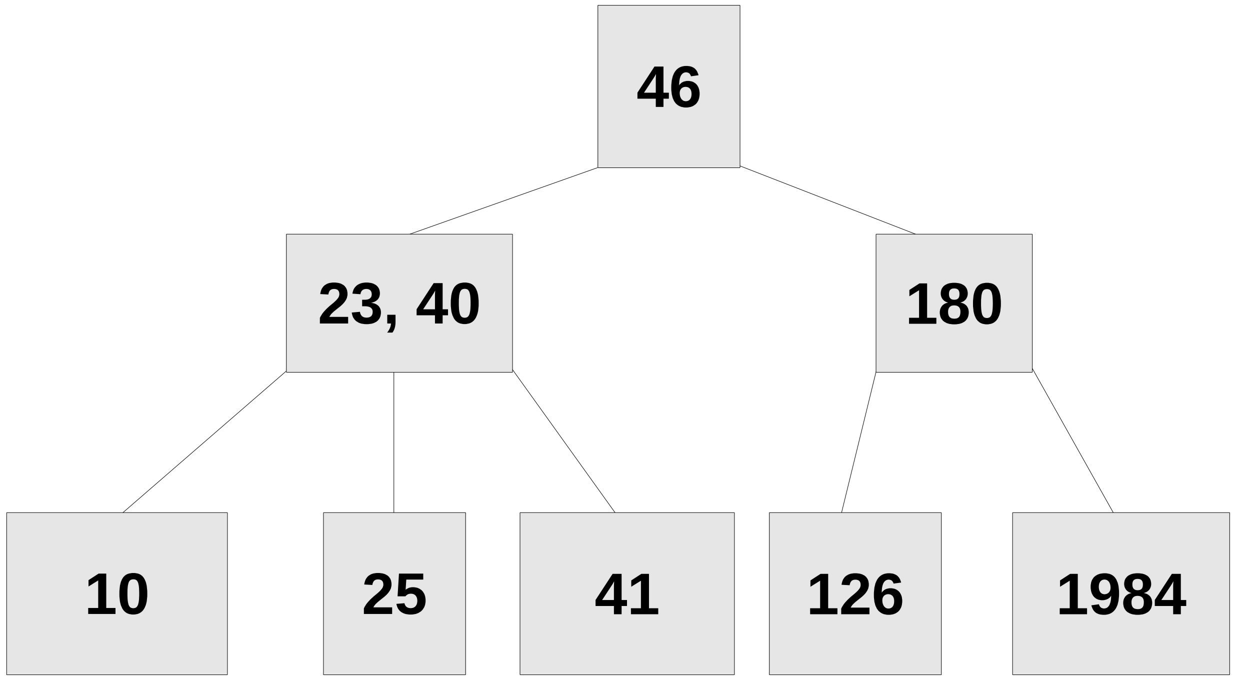
(this is going to get crazy)




Tada!

# 2-4 Tree Deletion

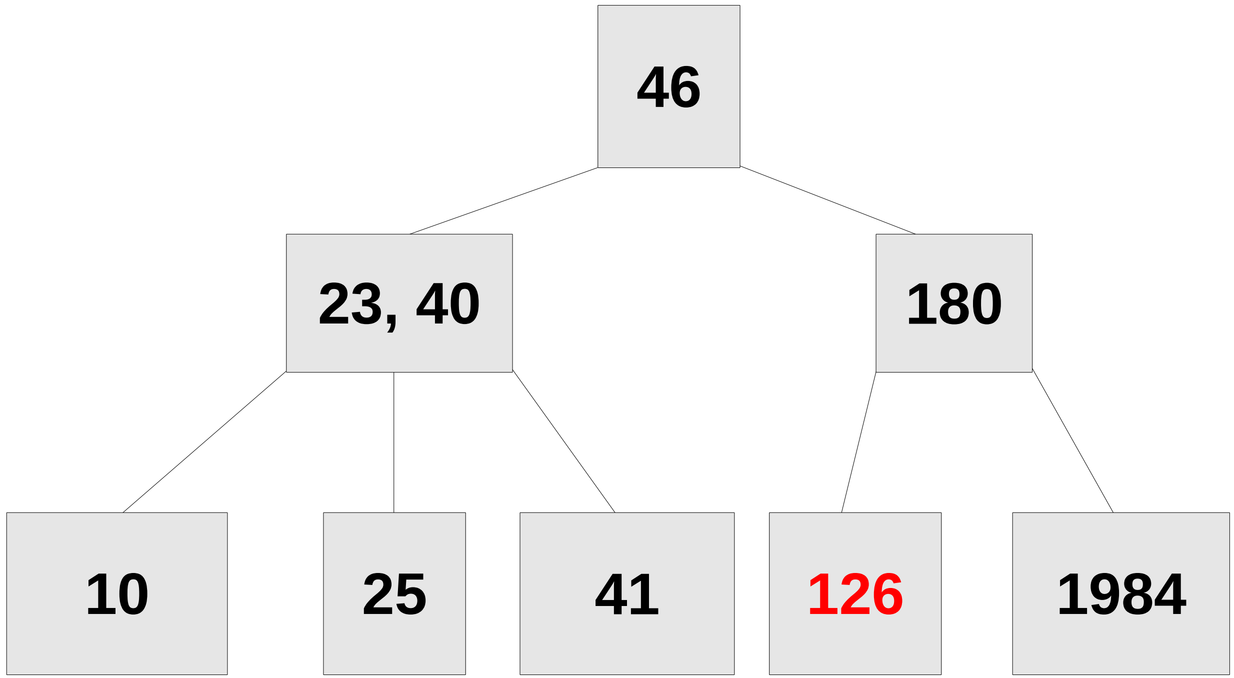
(this is going to get crazy)




Delete 126

# 2-4 Tree Deletion

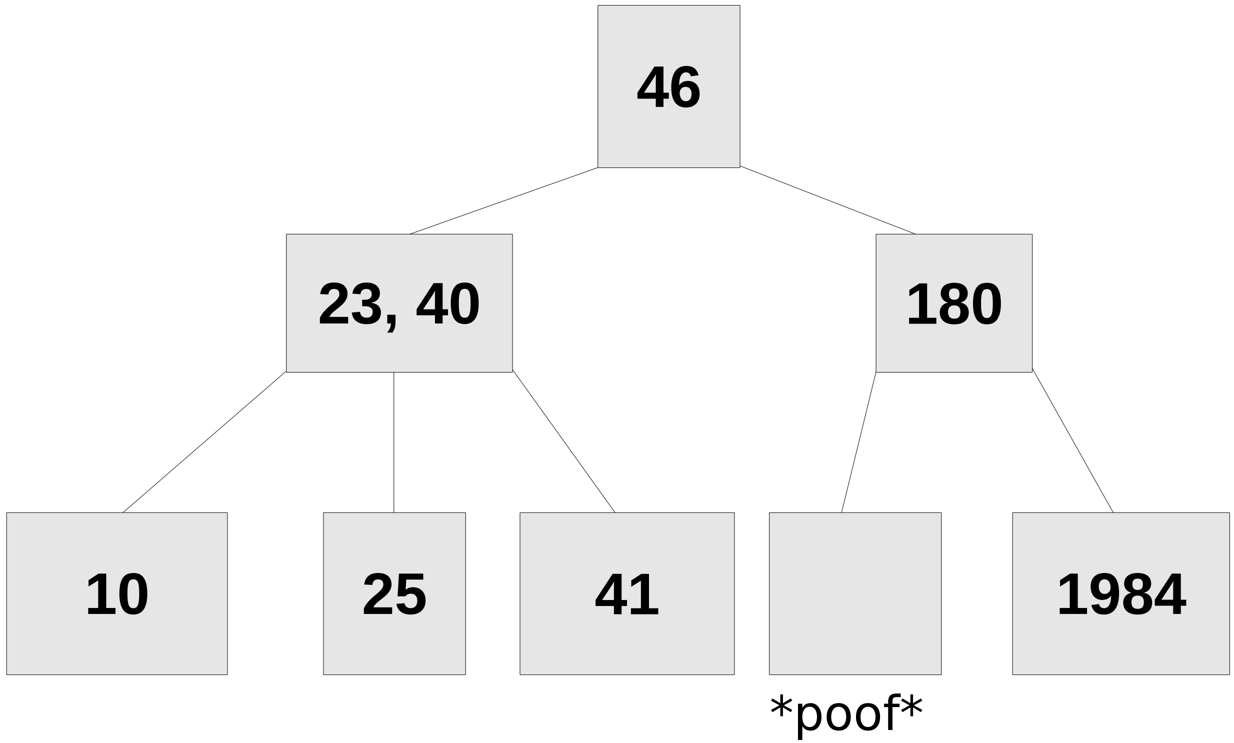
(this is going to get crazy)




Delete 126

# 2-4 Tree Deletion

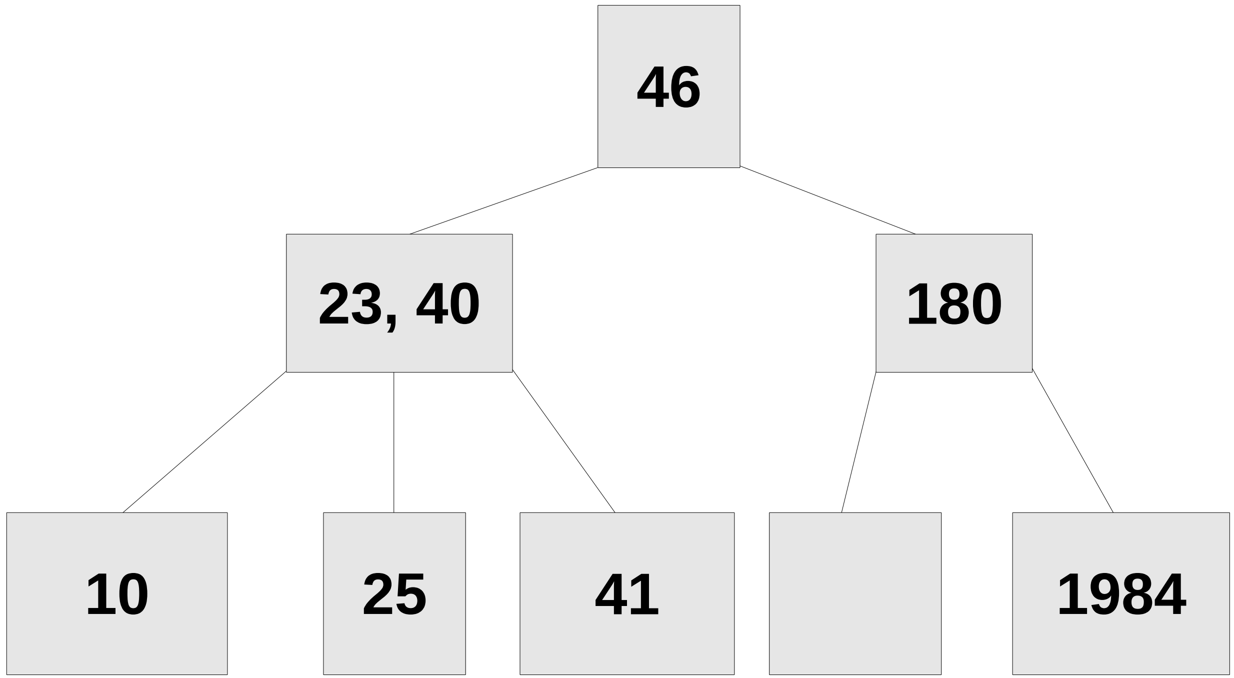
(this is going to get crazy)




Delete 126

# 2-4 Tree Deletion

(this is going to get crazy)



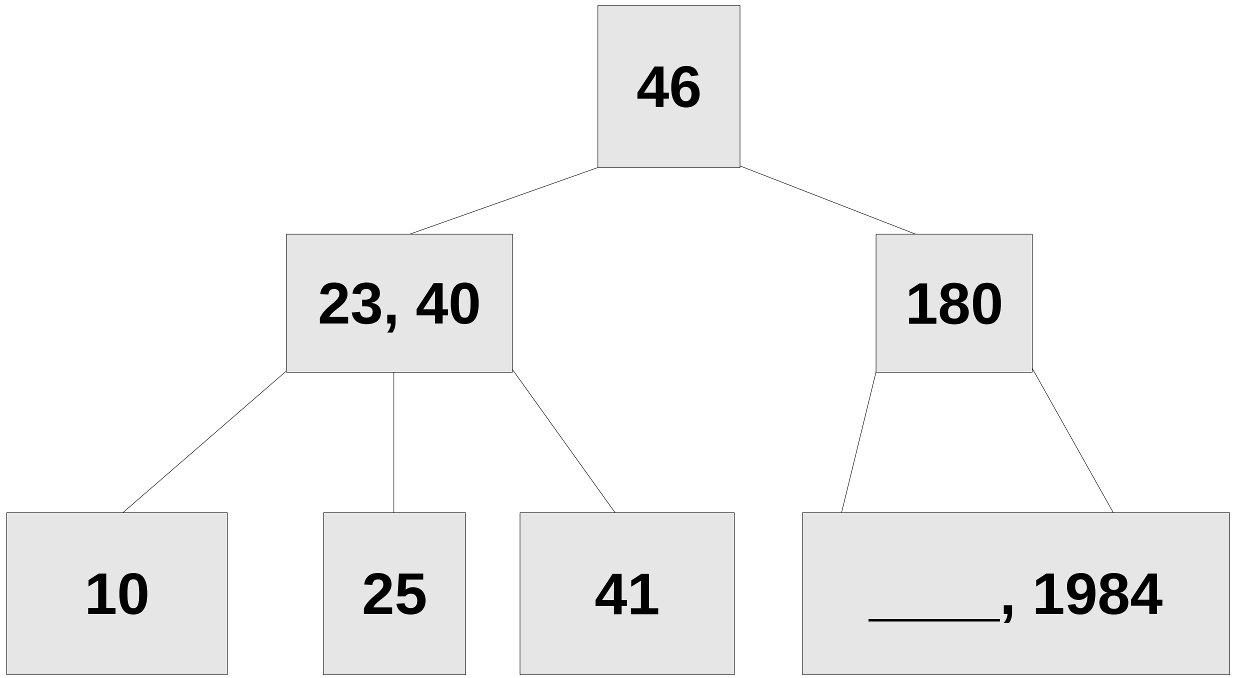

Delete 126

Again, the empty node has no siblings with extra elements! So, fuse and drop!

First, fuse the empty node with its left sibling (or right sibling, if it has no left sibling).

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 126

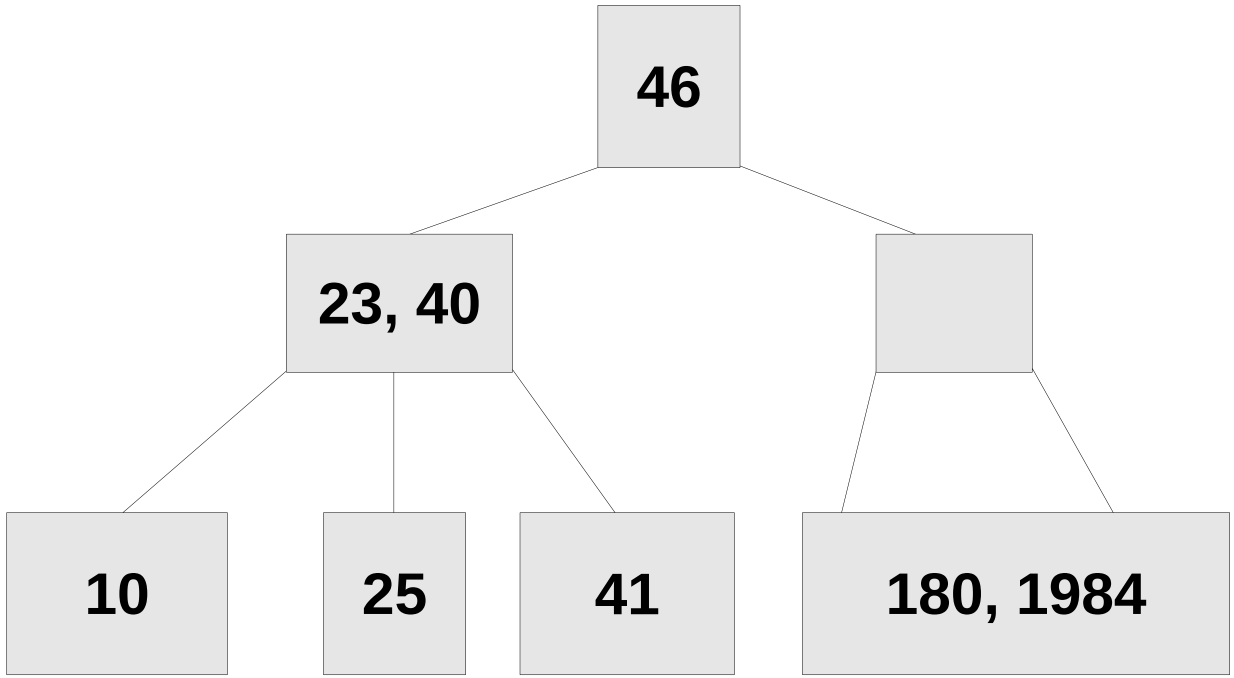
Again, the empty node has no siblings with extra elements! So, fuse and drop!

First, fuse the empty node with its left sibling (or right sibling, if it has no left sibling).

Now drop down the parent that was in between the newly fused nodes.

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 126

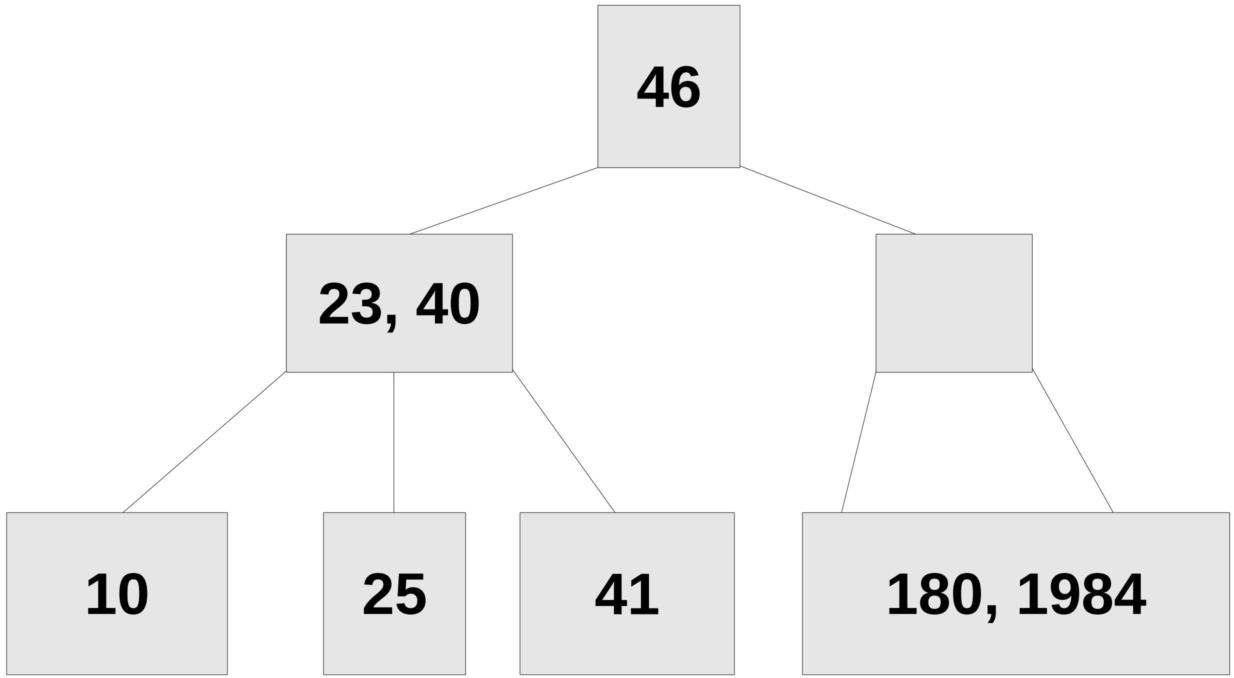
Again, the empty node has no siblings with extra elements! So, fuse and drop!

First, fuse the empty node with its left sibling (or right sibling, if it has no left sibling).

Now drop down the parent that was in between the newly fused nodes.

# 2-4 Tree Deletion

(this is going to get crazy)



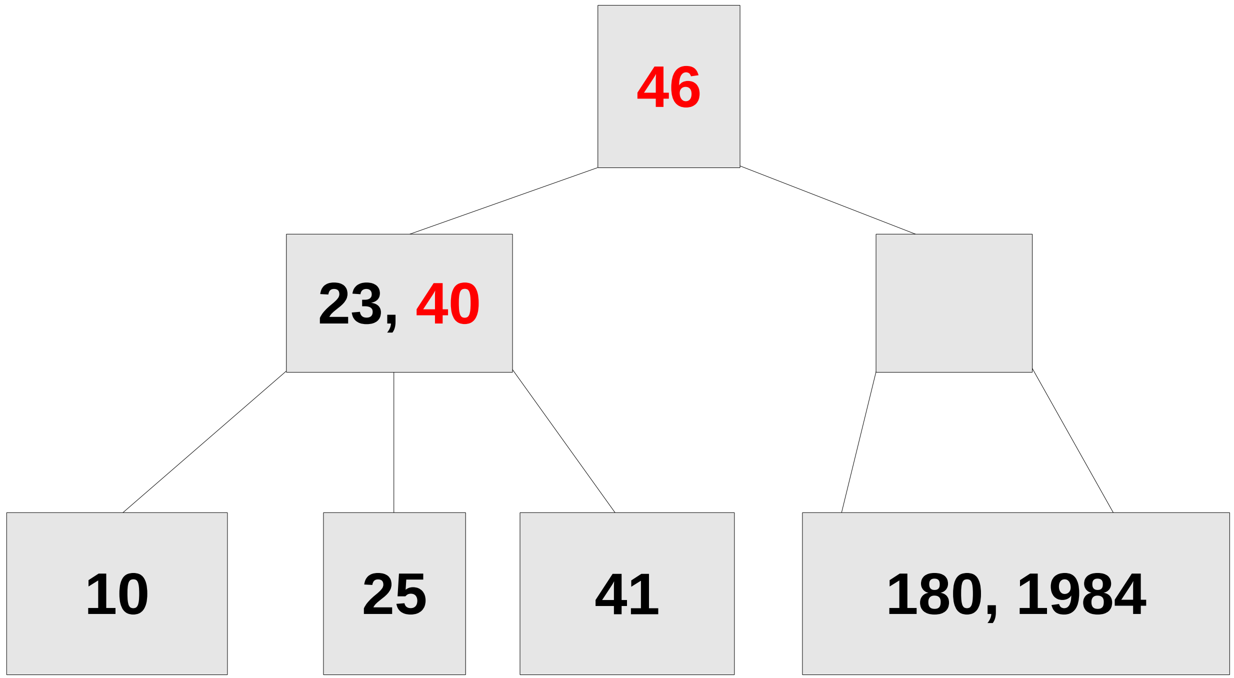

Delete 126

There’s now this empty node that we can fix with a transfer.

40 moves up. 46 moves down.

# 2-4 Tree Deletion

(this is going to get crazy)



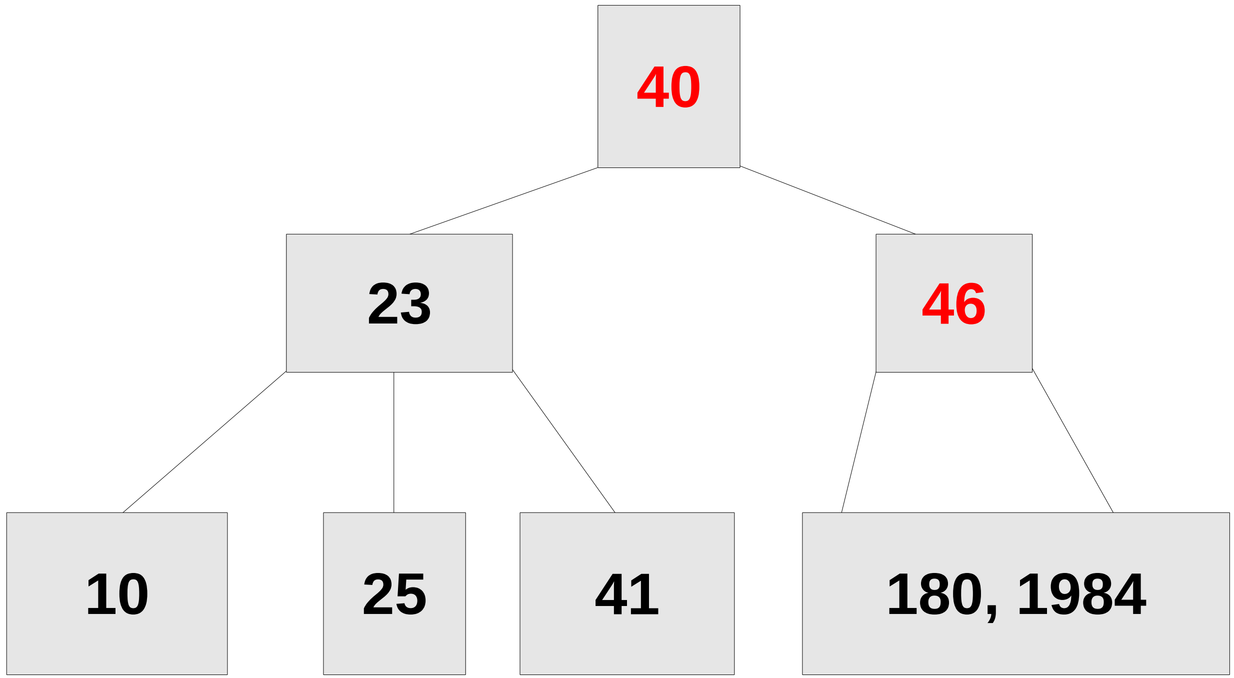

Delete 126

There's now this empty node that we can fix with a transfer.

40 moves up. 46 moves down.

# 2-4 Tree Deletion

(this is going to get crazy)



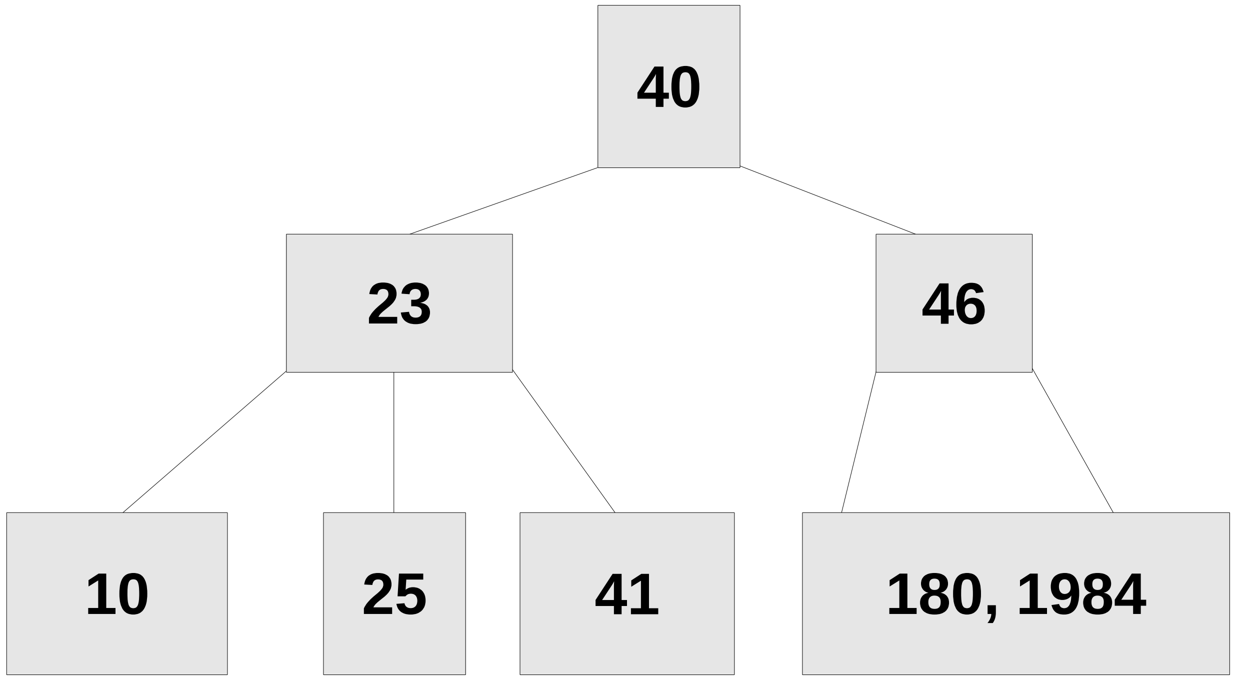

Delete 126

There’s now this empty node that we can fix with a transfer.

40 moves up. 46 moves down.

# 2-4 Tree Deletion

(this is going to get crazy)




Delete 126

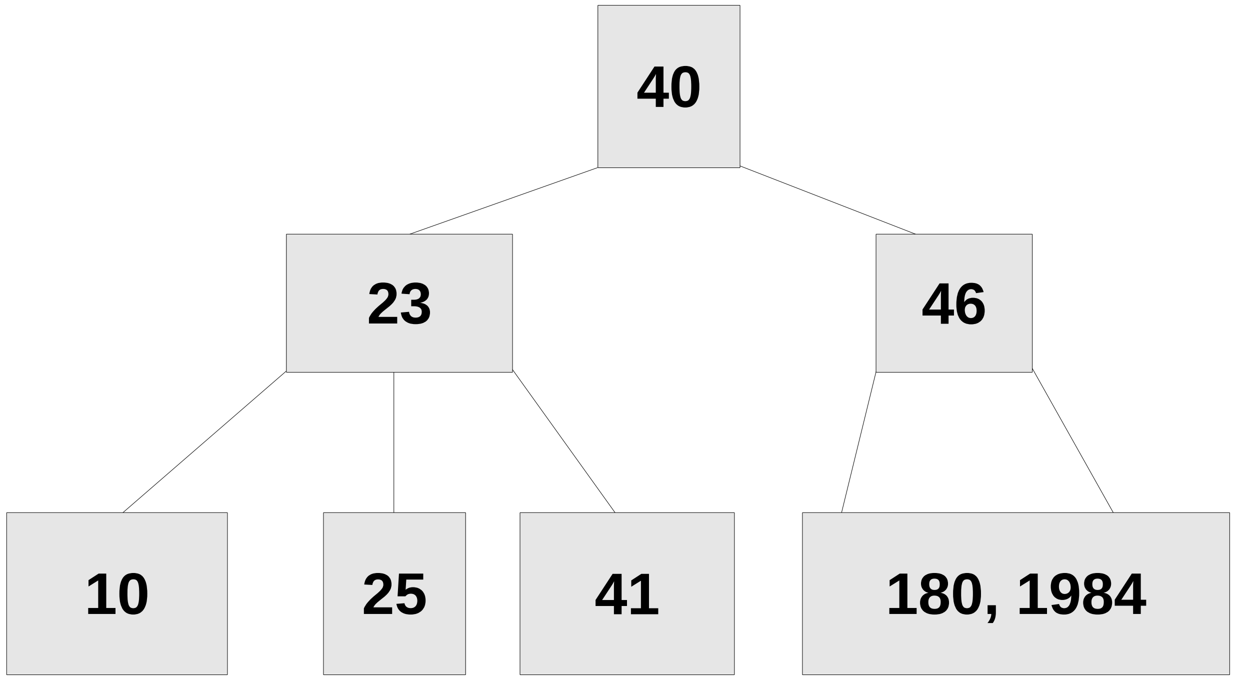
There’s now this empty node that we can fix with a transfer.

40 moves up. 46 moves down.

Notice that 41 is now orphaned, but 46 really only has one child.

# 2-4 Tree Deletion

(this is going to get crazy)

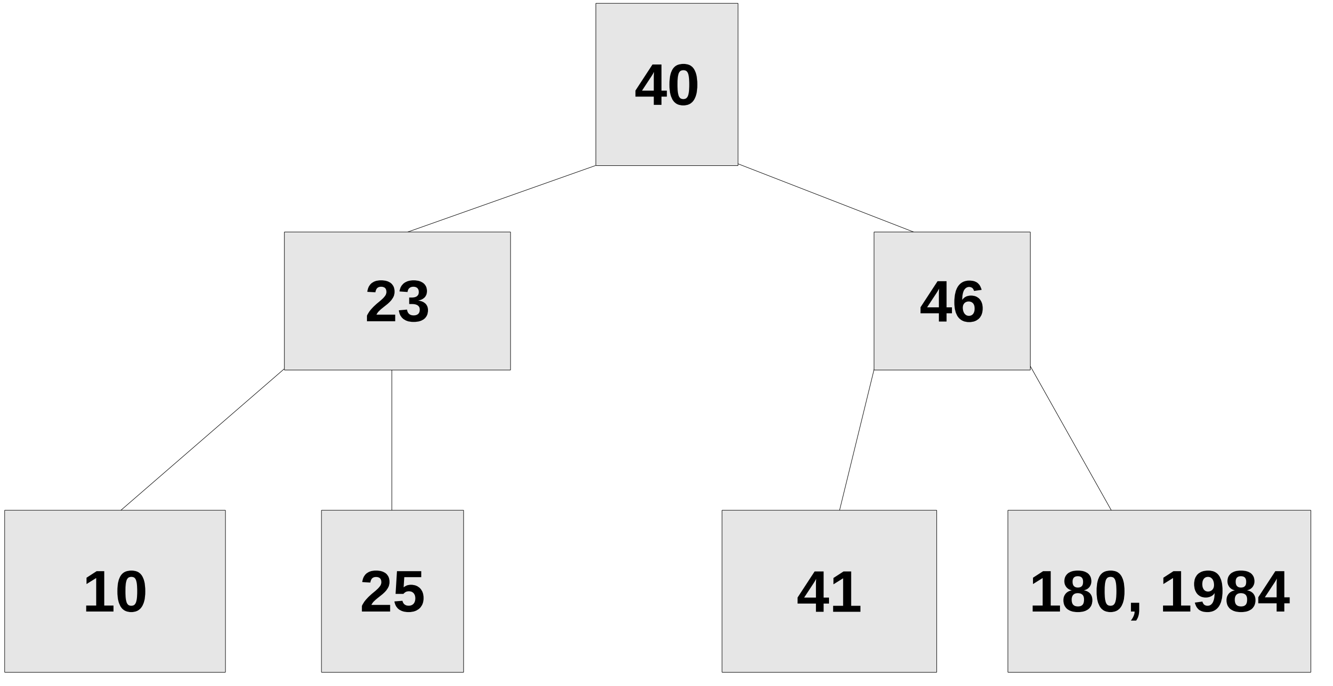



Delete 126

41 moves over to become the left child of 46.

# 2-4 Tree Deletion

(this is going to get crazy)

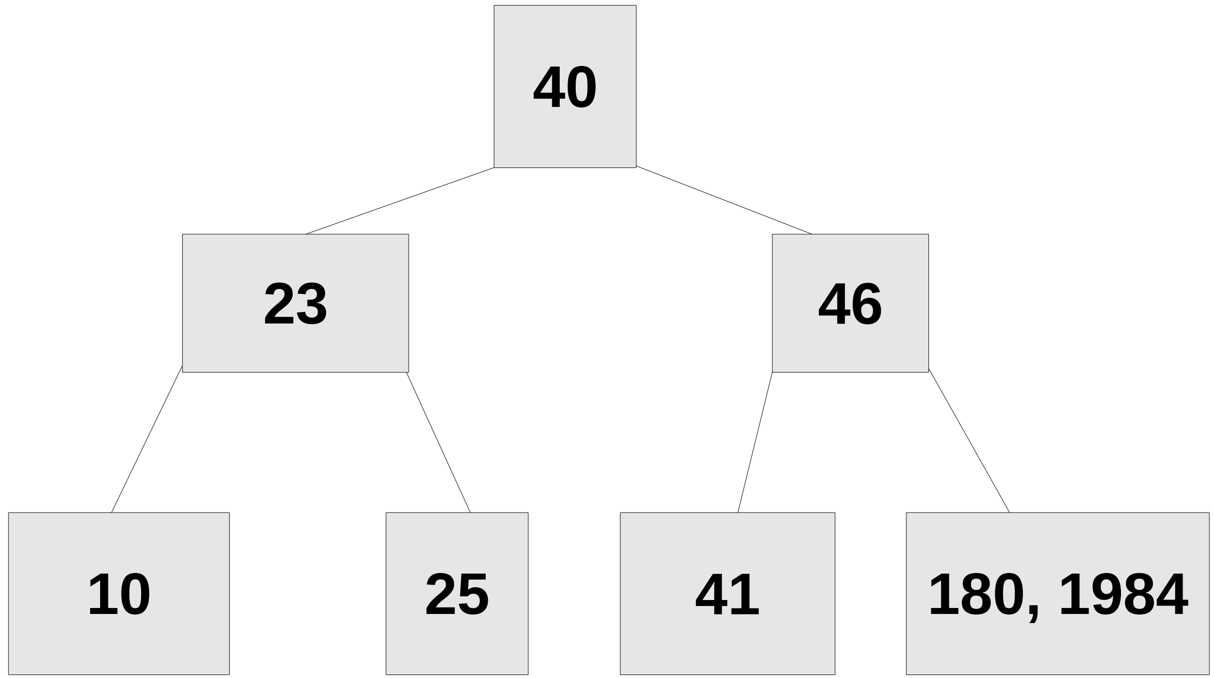



Delete 126

41 moves over to become the left child of 46.

# 2-4 Tree Deletion

(this is going to get crazy)




Tada!

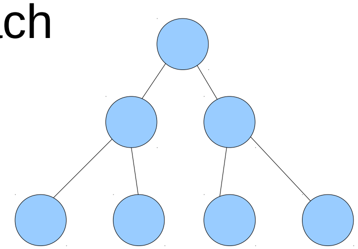
Okay, but what's the `runtime` for insert, delete, and search?

# 2-4 Tree Height

(which also tells us the runtime...)

Consider a tree of height *h*.

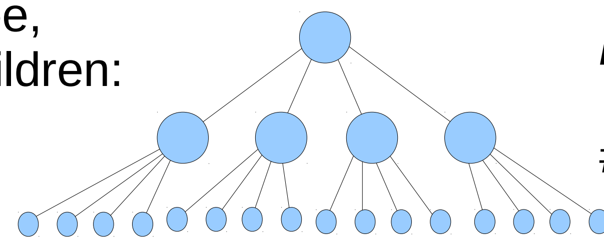
In the smallest such tree, each node has two children:



*h* = 2

# nodes in bottom level: $2^h$

In the largest such tree, each node has four children:



*h* = 2

# nodes in bottom level: $4^h$

Note: If that binary tree has $n$ nodes, it has a layer of $n + 1$ null references. So, we have:

$2^{h+1} \leq n+1 \leq 4^{h+1}$

$(h+1) \leq \log_2(n+1) \leq 2h+2$ *by taking $\log_2$ of the entire expression*

$h \leq \log(n+1) - 1$ *and* $h \geq (1/2)\log(n+1) - 1$ *by solving each side separately for h*

$(1/2)\log(n+1) - 1 \leq h \leq \log(n+1) - 1 \Rightarrow \Theta(\log n)$ *because math and formal defn. of Big-Theta*

Note: Proof: http://www.cs.mcgill.ca/~cs251/ClosestPair/proof6.6.html