Lecture 9

Augmenting Trees (contd.)

Finding the Element with ith Rank

Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T.root, i)$.

Finding the Element with ith Rank

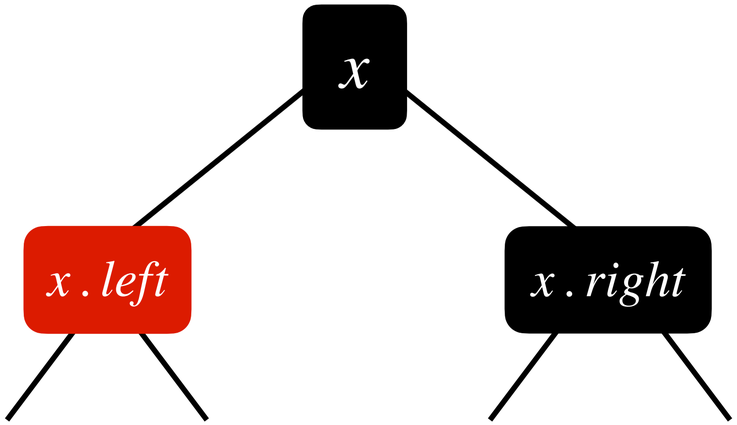
To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Select(x, i)**:**

Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T.root, i)$.

Select(x, i)**:**

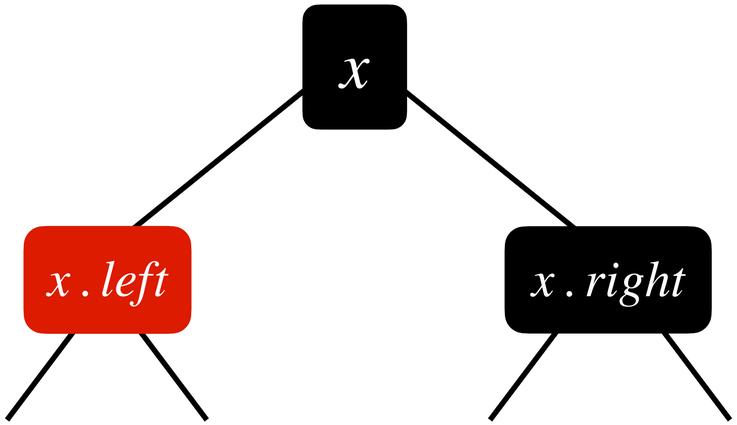


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T.root, i)$.

Select(x, i)**:**

1. $r =$ _____



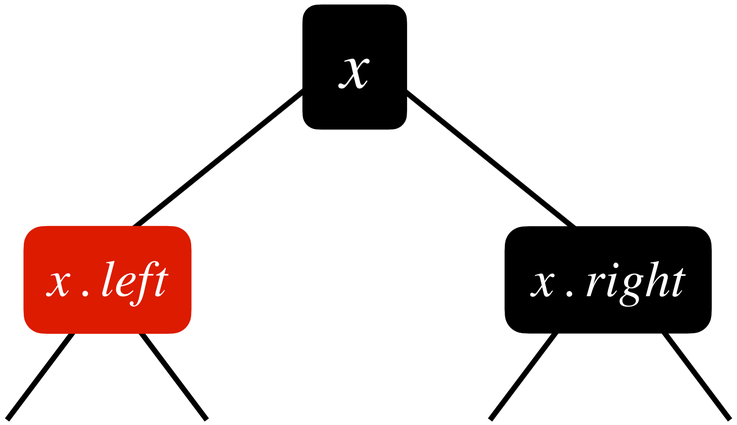
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T.root, i)$.

Calculate the rank of x in subtree(x)

Select(x, i):

1. $r =$ _____



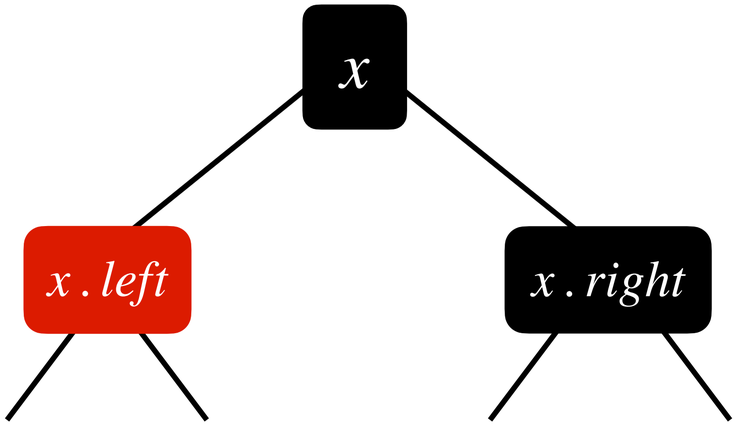
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Calculate the rank of x in subtree(x)

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$

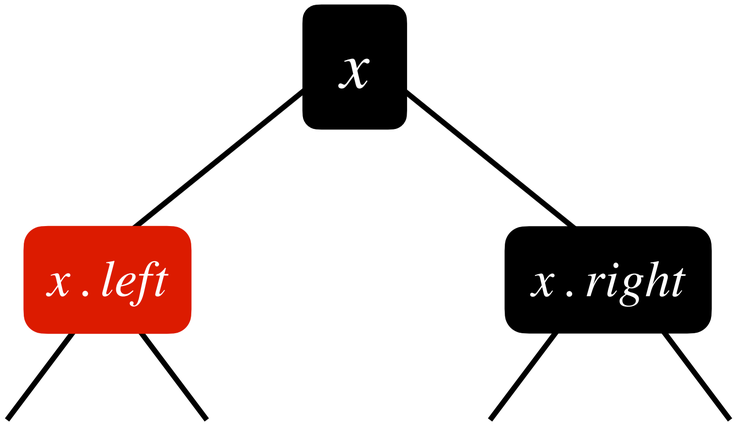


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. **if** $i == r$



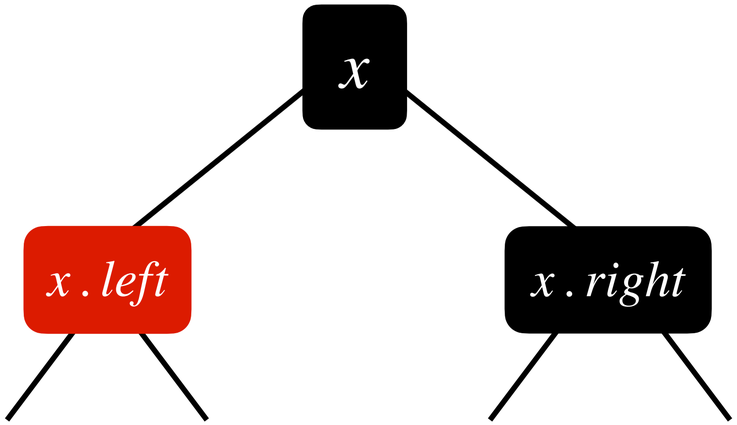
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Element with ith rank must be x

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. **if** $i == r$



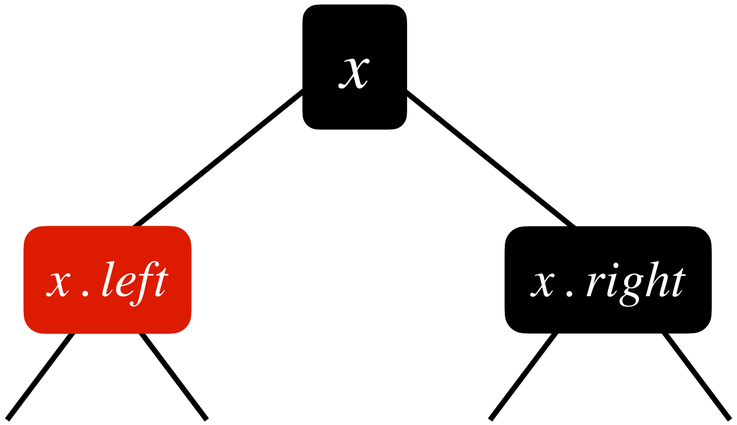
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Element with ith rank must be x

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. **if** $i == r$
3. **return** x

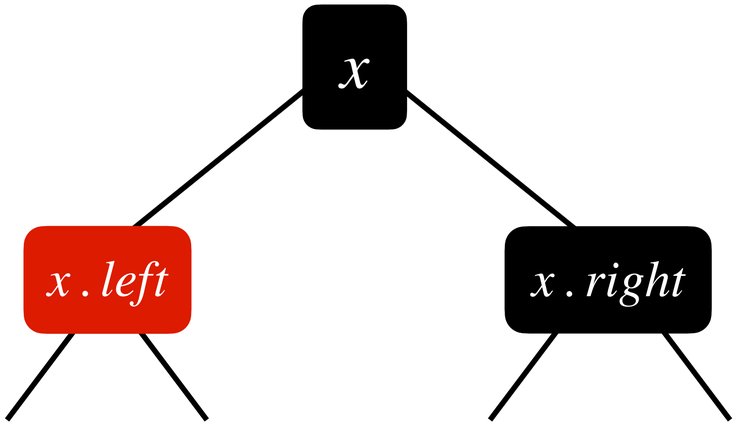


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. **if** $i == r$
3. **return** x
4. **else if** $i < r$



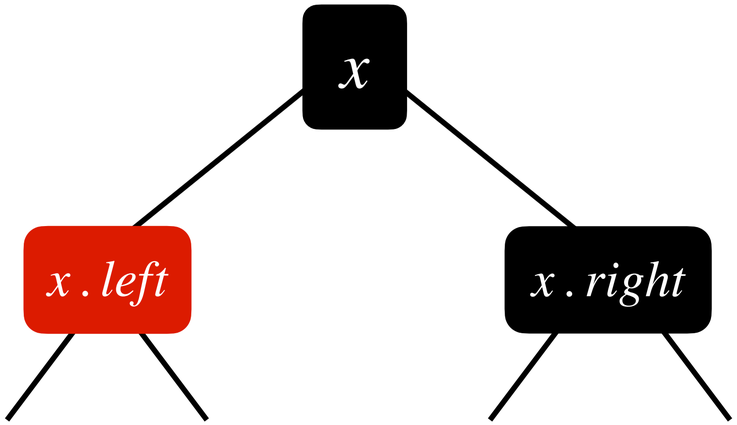
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T.root, i)$.

Element with ith rank must fall in the left-subtree of x

Select(x, i)**:**

```
1.  r = x.left.size + 1
2.  if i == r
3.      return x
4.  else if i < r
```



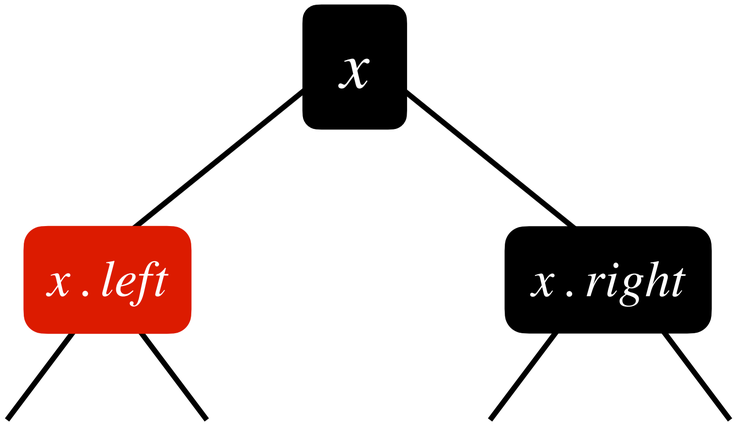
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T.root, i)$.

Element with ith rank must fall in the left-subtree of x

Select(x, i):

1. $r = x.left.size + 1$
2. **if** $i == r$
3. **return** x
4. **else if** $i < r$
5. **return** _____



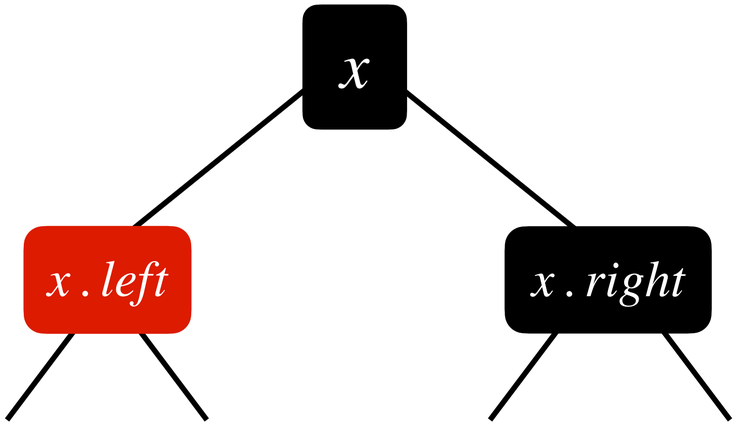
Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Element with ith rank must fall in the left-subtree of x

Select(x, i)**:**

```
1.  r = x.left.size + 1
2.  if i == r
3.      return x
4.  else if i < r
5.      return Select(x.left, i)
```

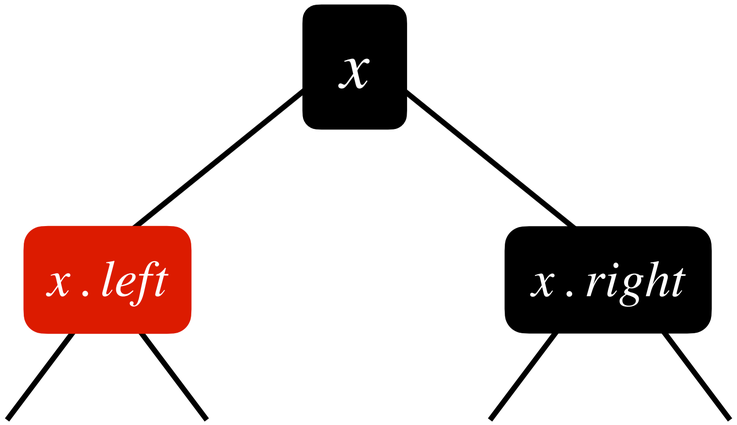


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. **if** $i == r$
3. **return** x
4. **else if** $i < r$
5. **return** **Select**$(x\,.\,left, i)$
6. **else**

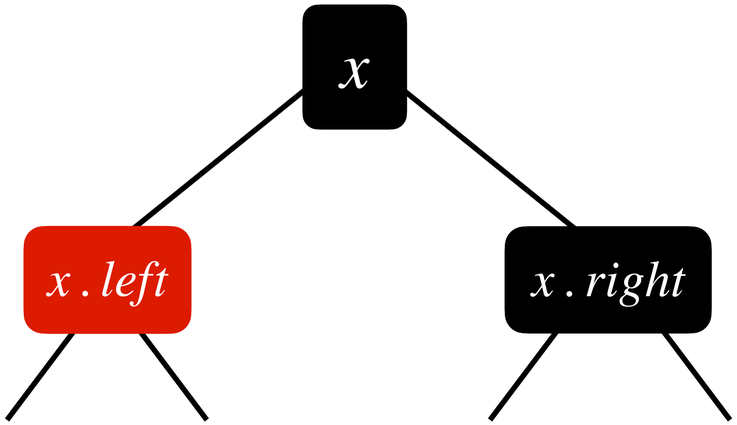


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

```
Select(x, i):
1.   r = x.left.size + 1
2.   if i == r
3.       return x
4.   else if i < r
5.       return Select(x.left, i)
6.   else
```

Element with ith rank must fall in the right-subtree of x

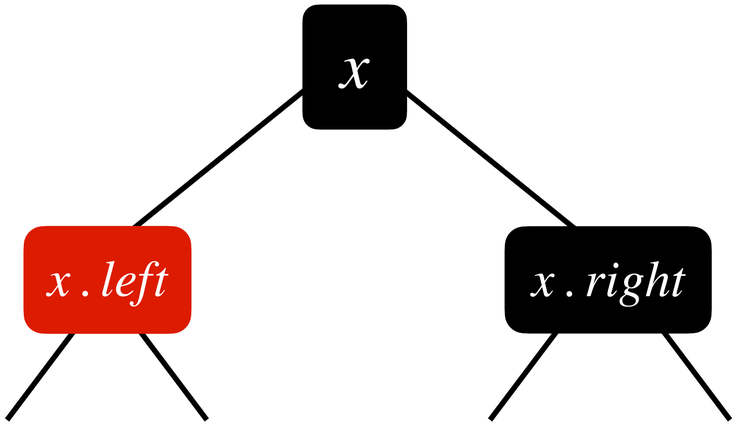


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

```
Select(x, i):
1.   r = x.left.size + 1
2.   if i == r
3.       return x
4.   else if i < r
5.       return Select(x.left, i)
6.   else
7.       return _____
```

Element with ith rank must fall in the right-subtree of x

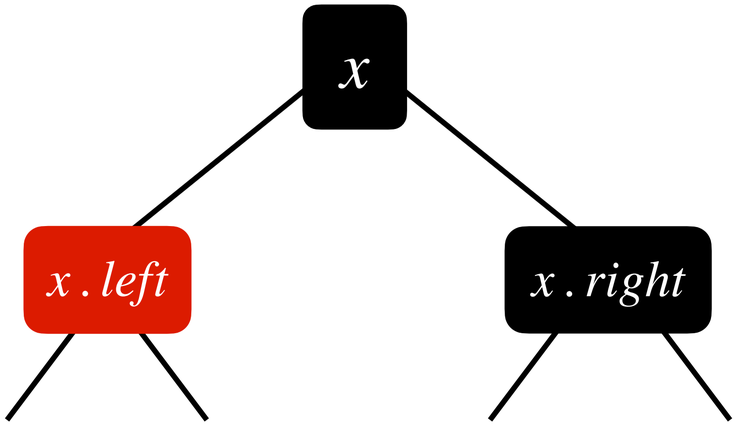


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

```
Select(x, i):
1.   r = x.left.size + 1
2.   if i == r
3.       return x
4.   else if i < r
5.       return Select(x.left, i)
6.   else
7.       return Select(x.right, i − r)
```

Element with ith rank must fall in the right-subtree of x

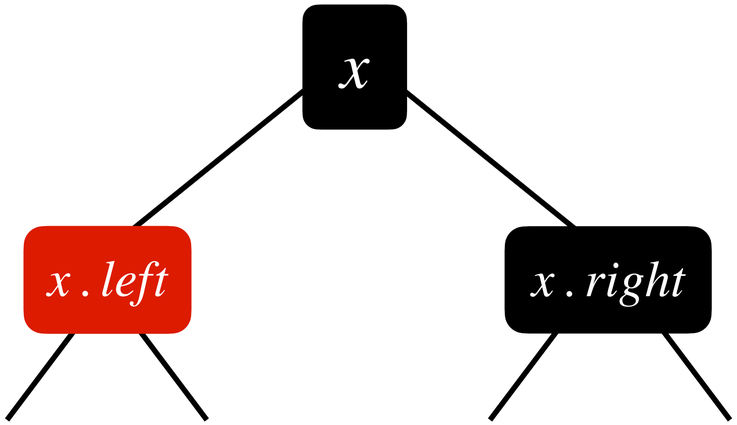


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

Select(x, i)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. **if** $i == r$
3. **return** x
4. **else if** $i < r$
5. **return** **Select**$(x\,.\,left, i)$
6. **else**
7. **return** **Select**$(x\,.\,right, i - r)$

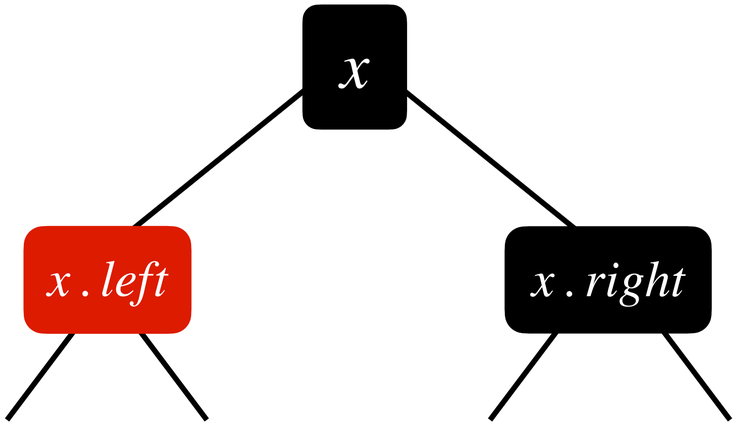


Finding the Element with ith Rank

To find the element with ith rank in T call **Select**$(T\,.\,root, i)$.

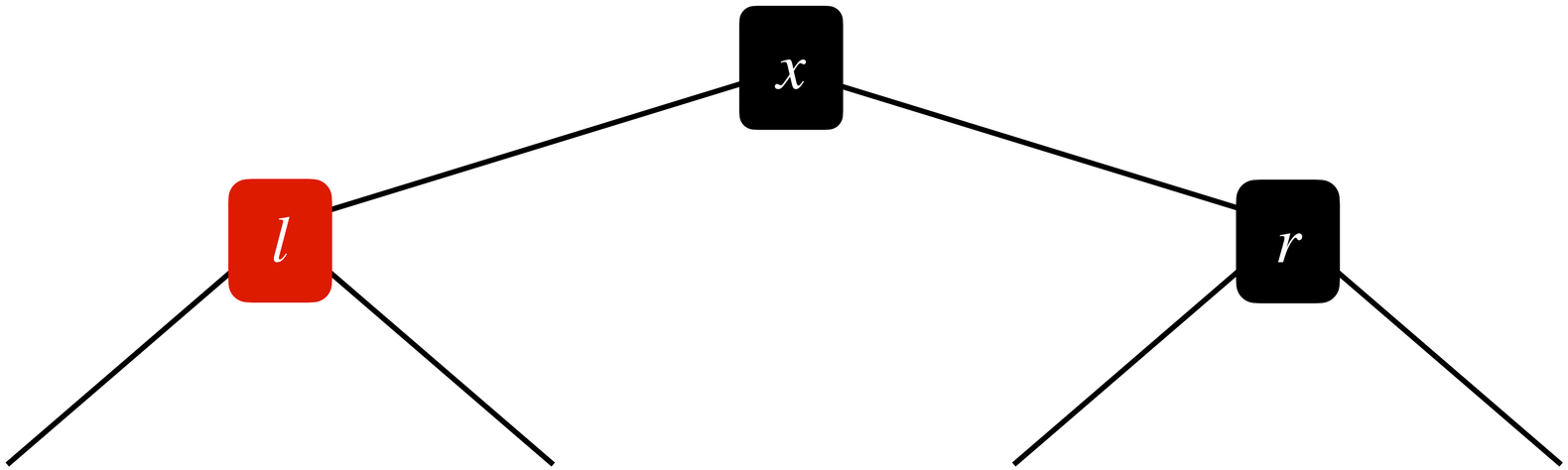
Select(x, i)**:**

```
1.  r = x.left.size + 1
2.  if i == r
3.      return x
4.  else if i < r
5.      return Select(x.left, i)
6.  else
7.      return Select(x.right, i − r)
```

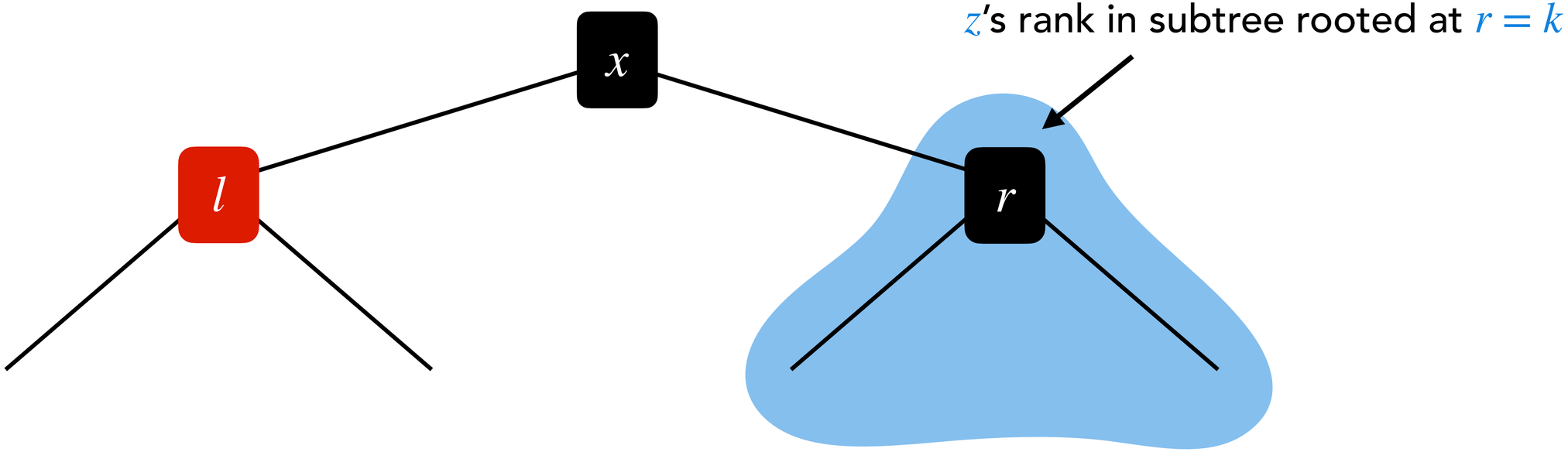


Time Complexity: $\Theta(h) = \Theta(\log n)$ as with every recursive call algorithm goes one level down.

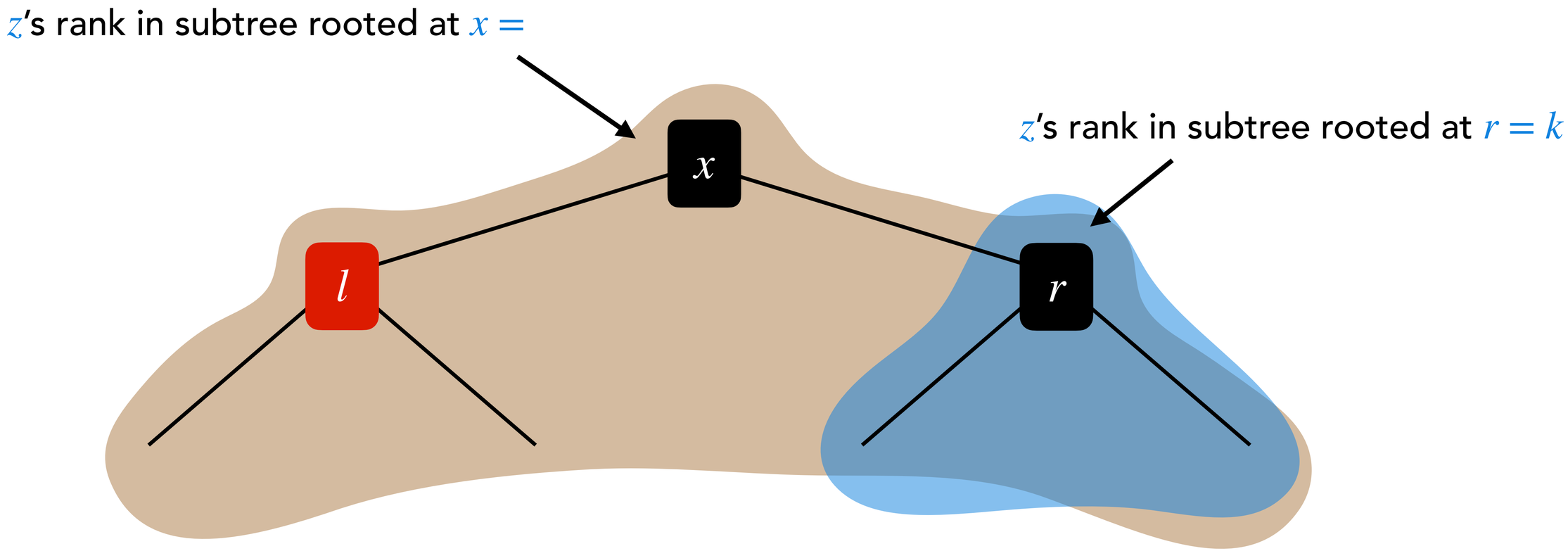
Finding The Rank of an Element



Finding The Rank of an Element

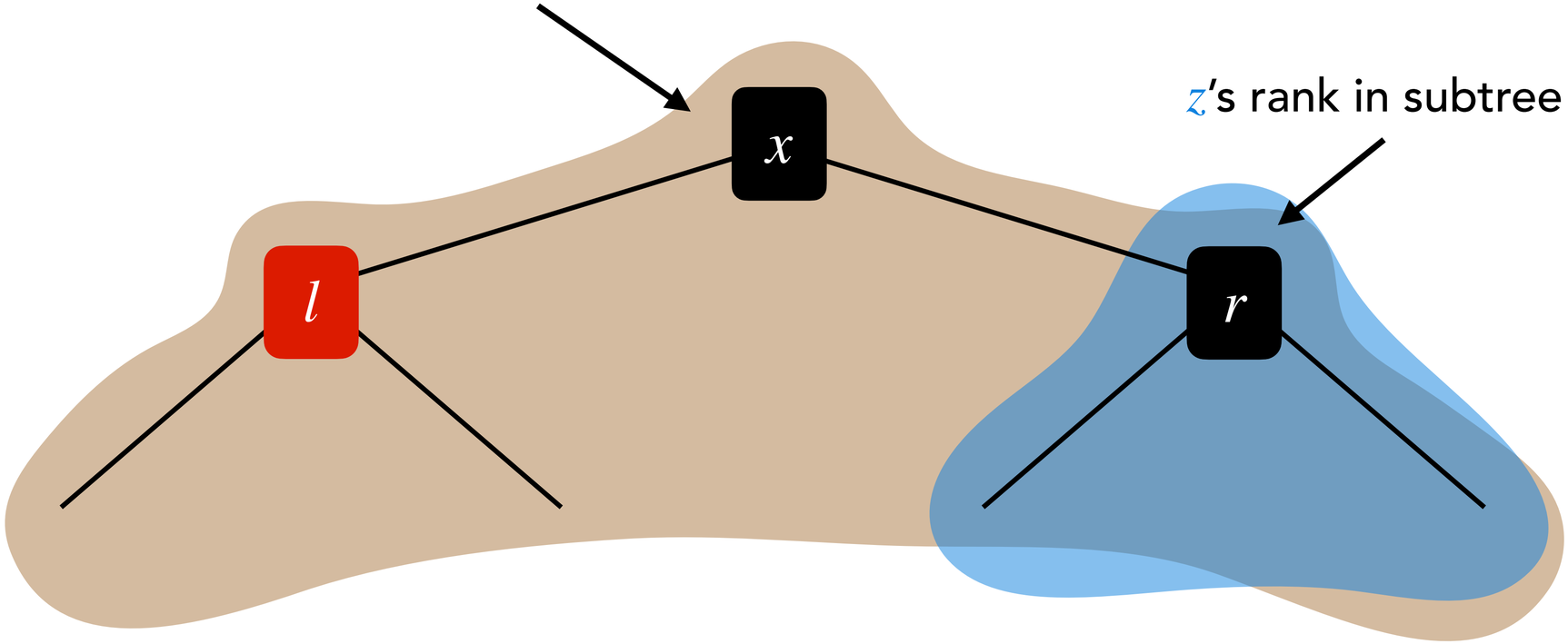


Finding The Rank of an Element



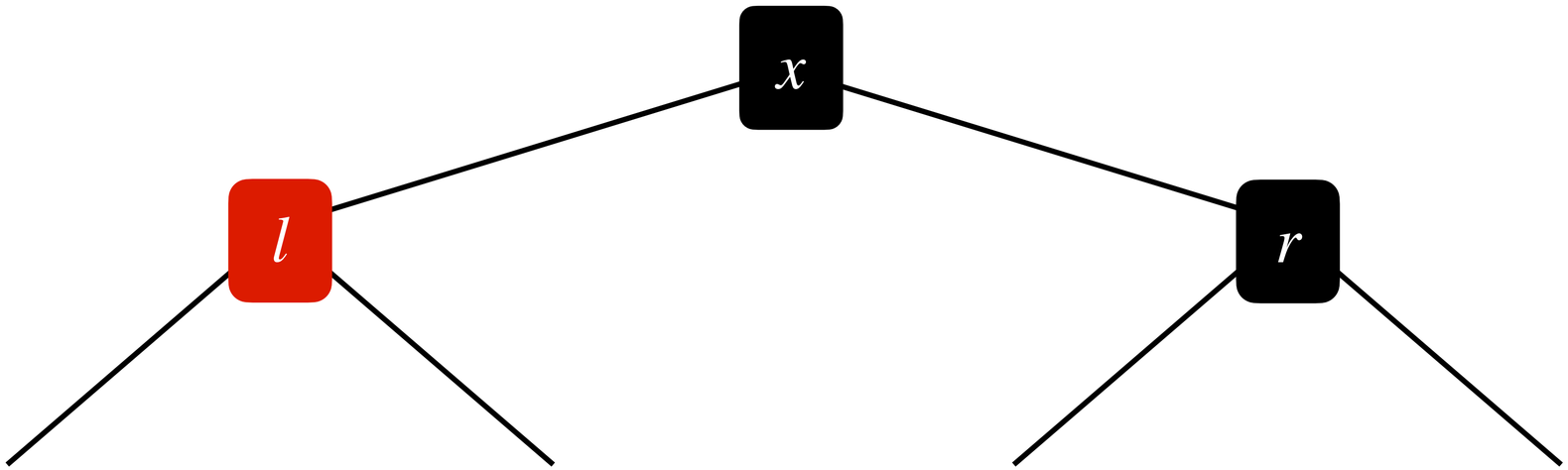
Finding The Rank of an Element

z's rank in subtree rooted at $x = k + l\,.\,size + 1$



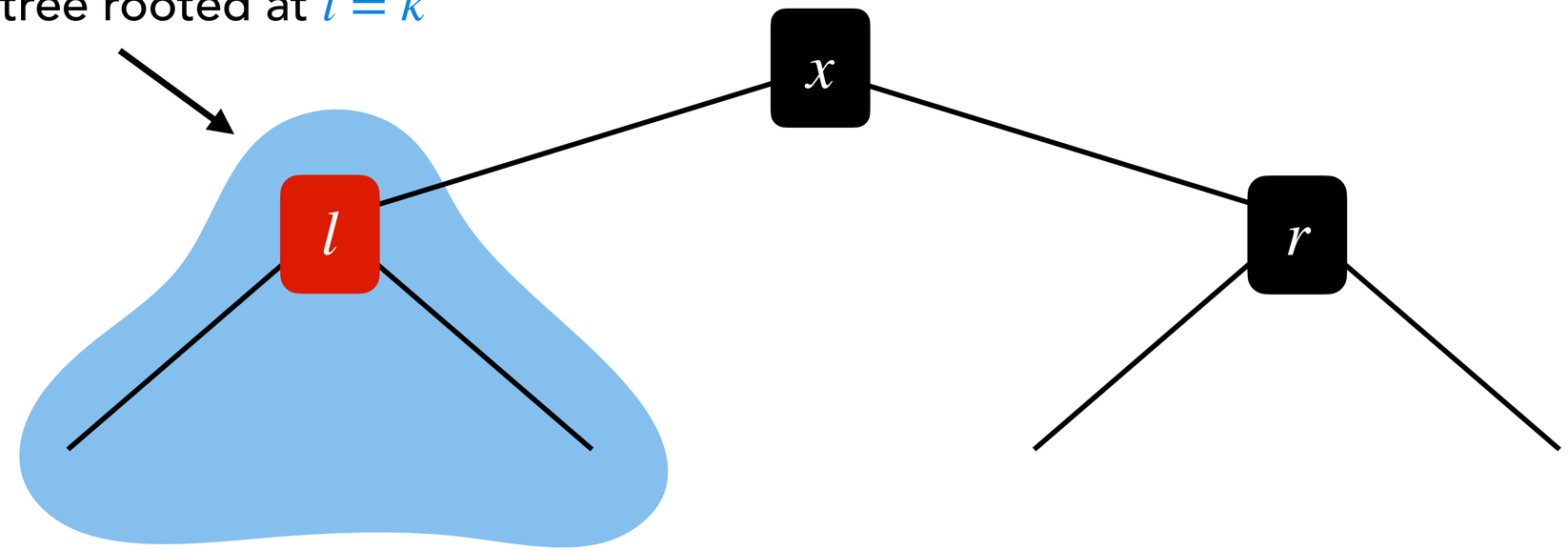
z's rank in subtree rooted at $r = k$

Finding The Rank of an Element

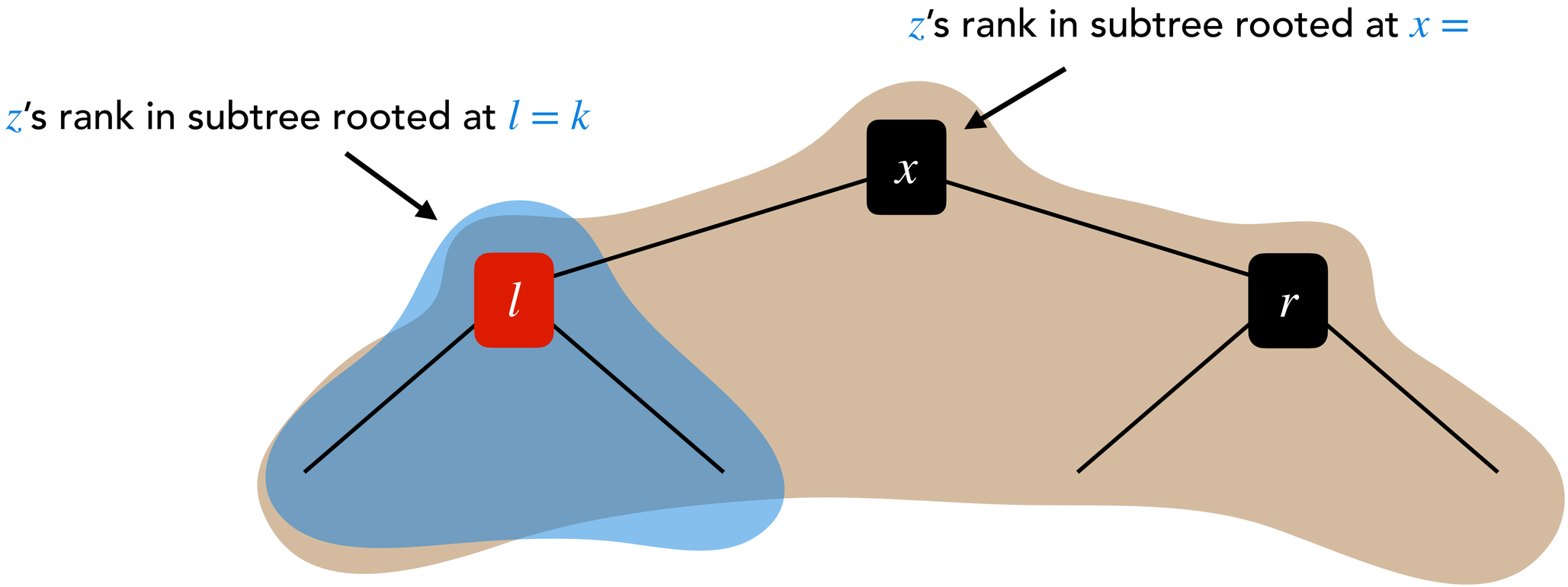


Finding The Rank of an Element

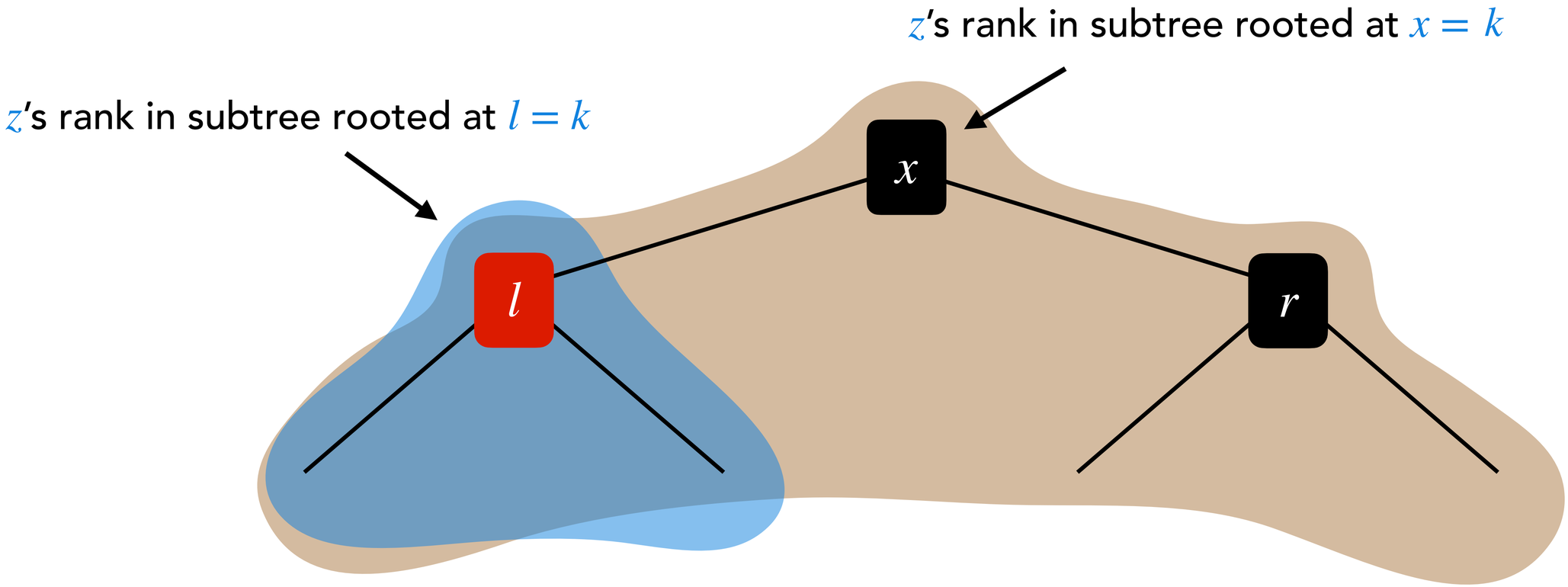
z's rank in subtree rooted at $l = k$



Finding The Rank of an Element



Finding The Rank of an Element



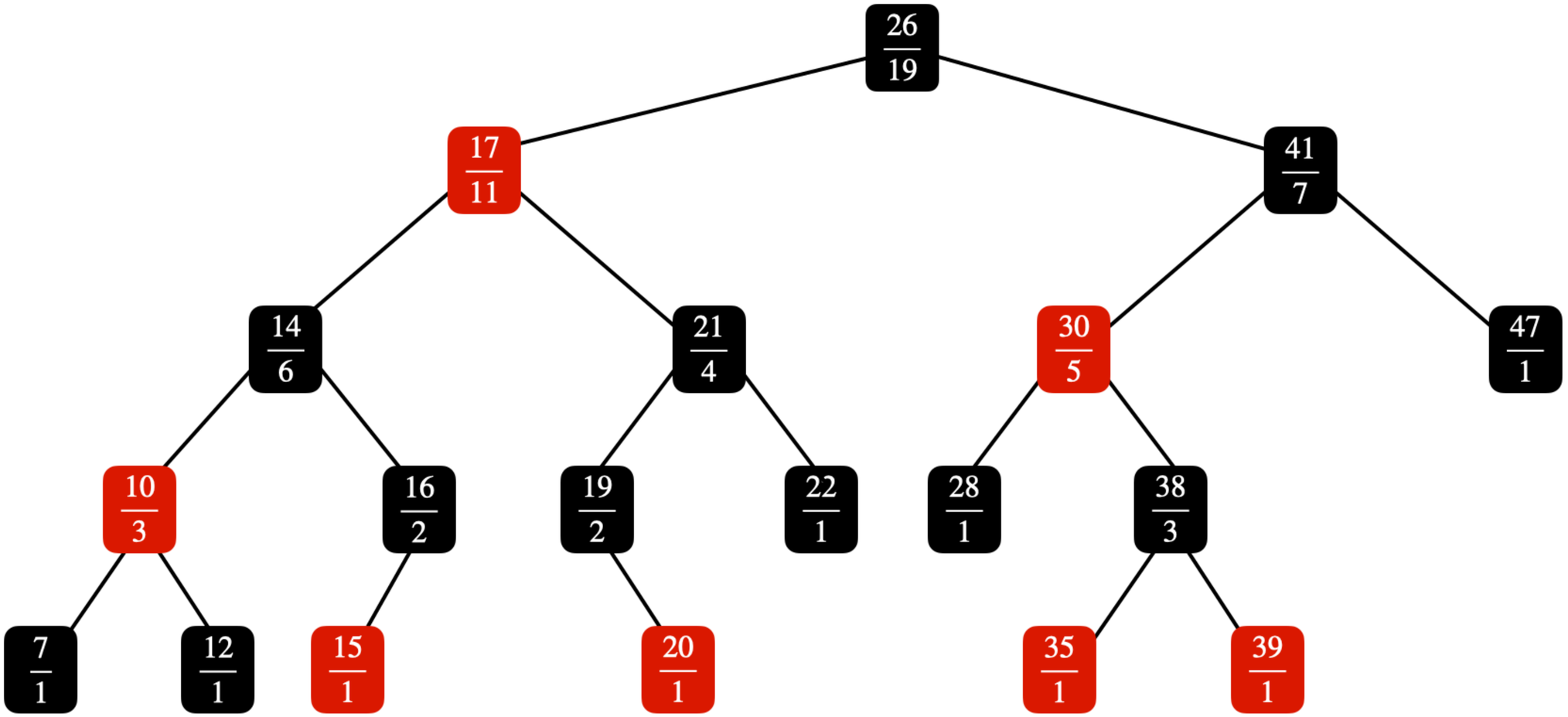
Finding The Rank of an Element

Finding The Rank of an Element

Example: Find the rank of 38 in the below **set** or RB-tree.

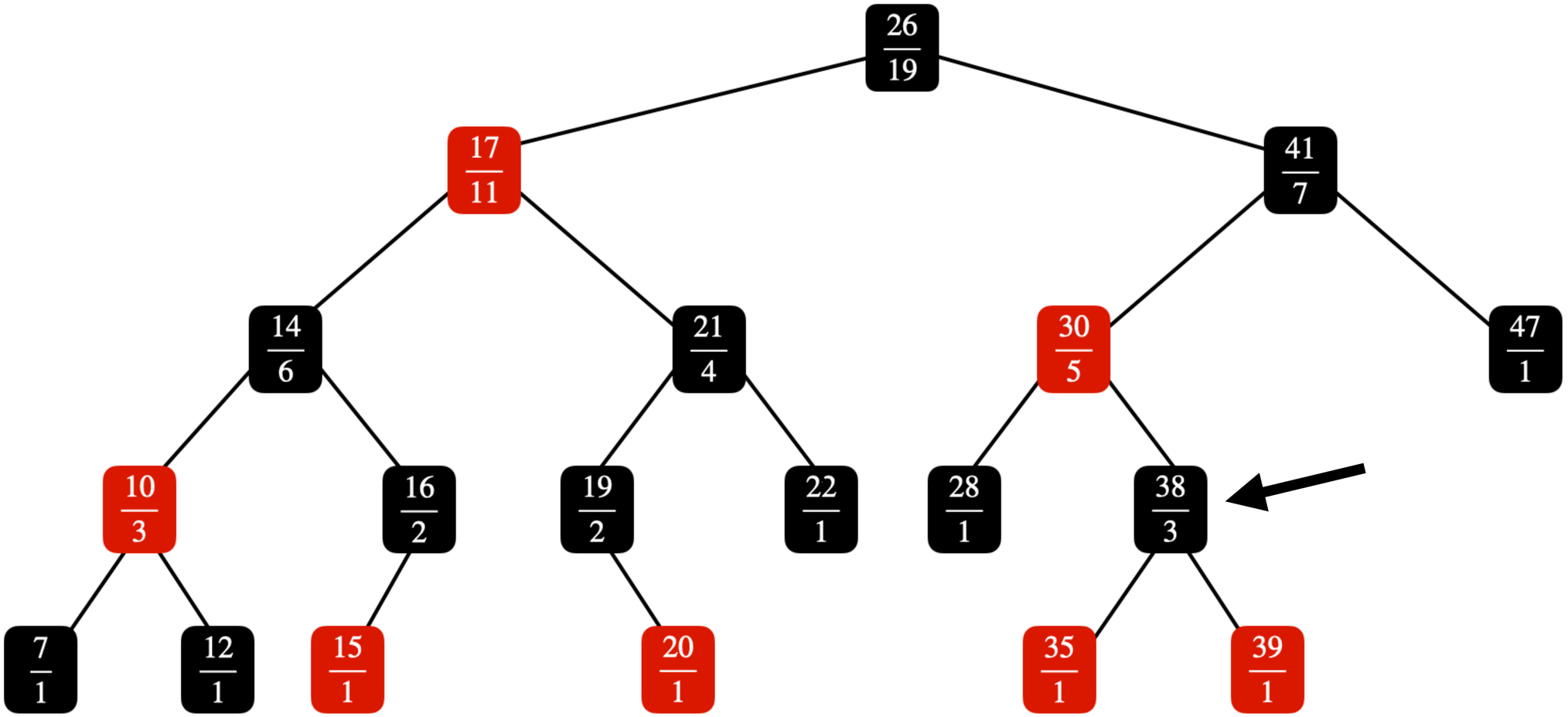
Finding The Rank of an Element

Example: Find the rank of 38 in the below **set** or RB-tree.

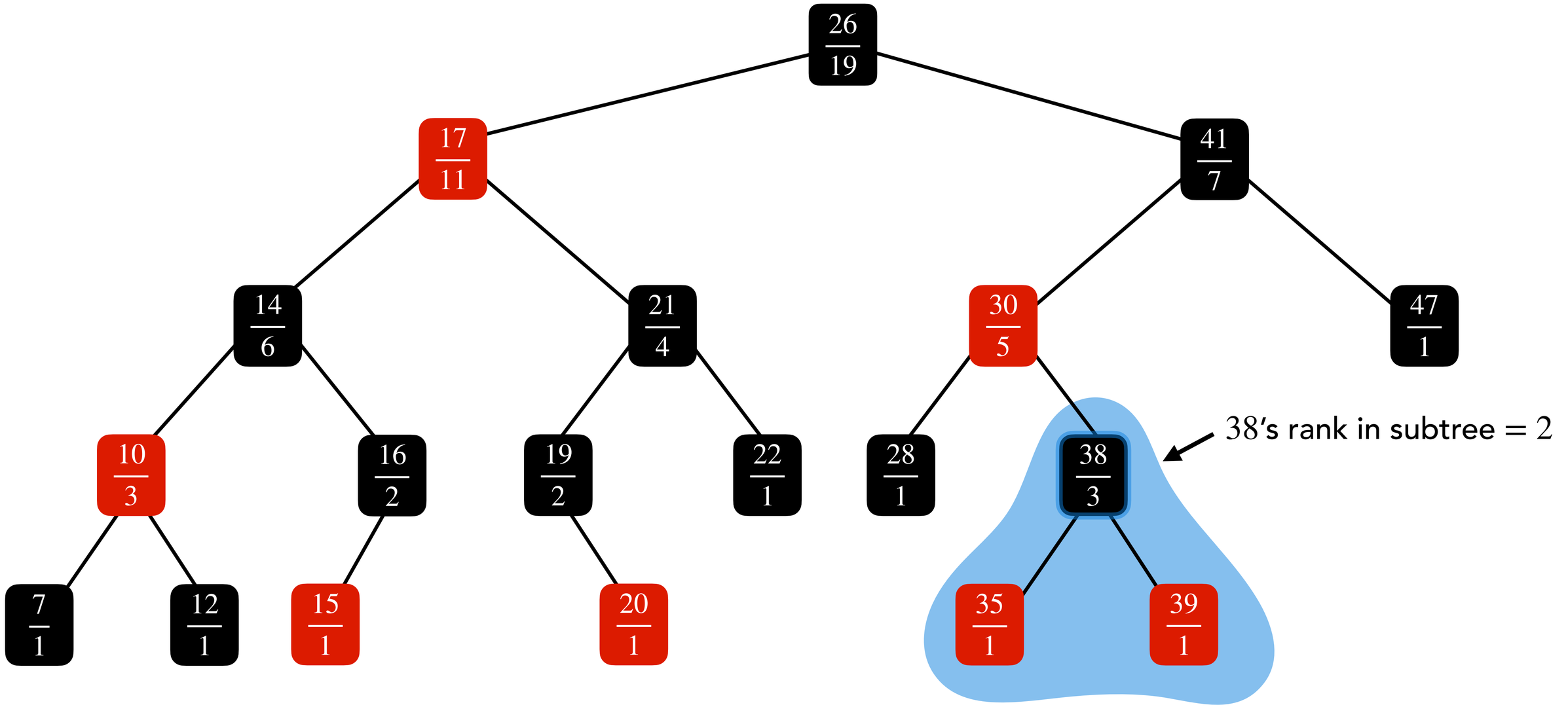


Finding The Rank of an Element

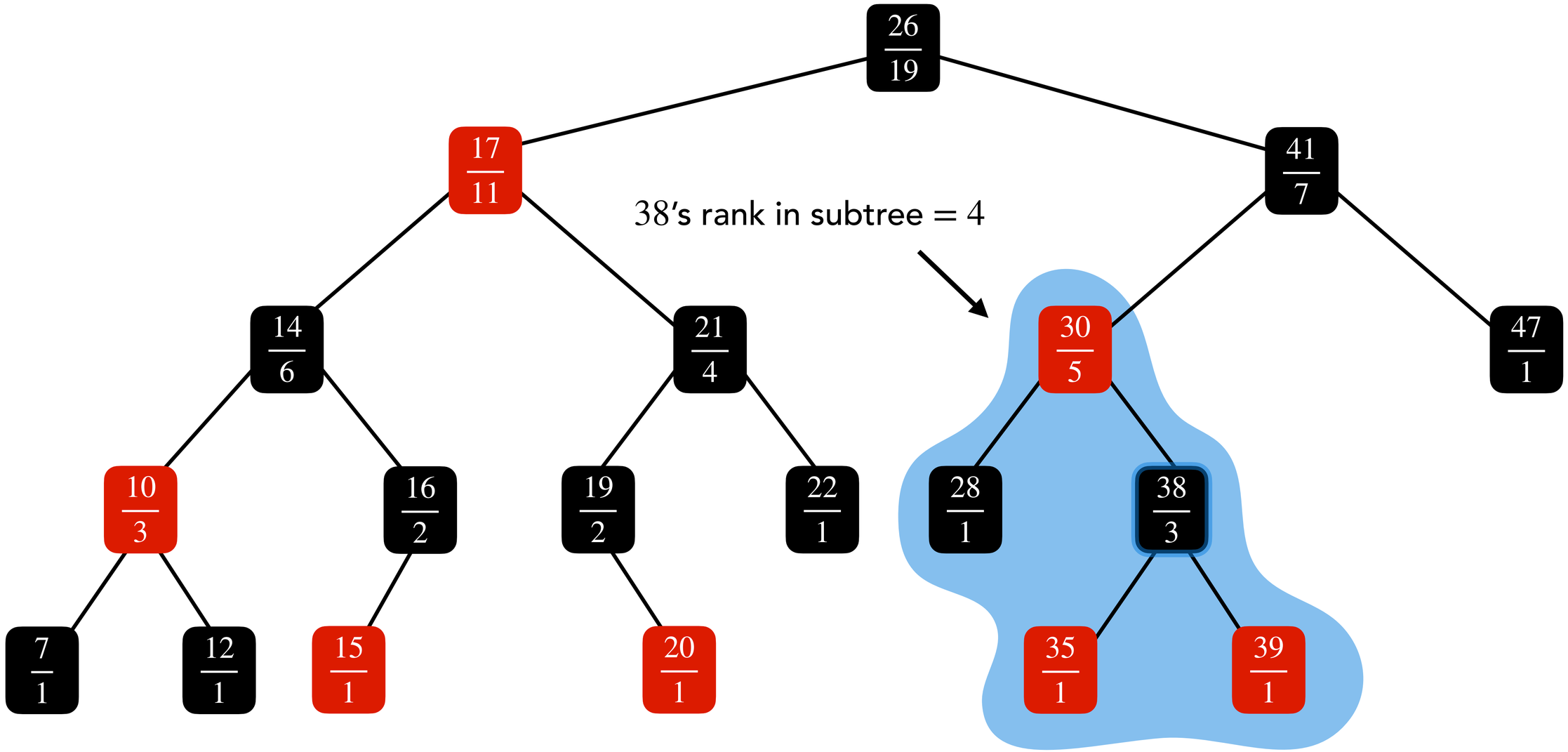
Example: Find the rank of 38 in the below **set** or RB-tree.



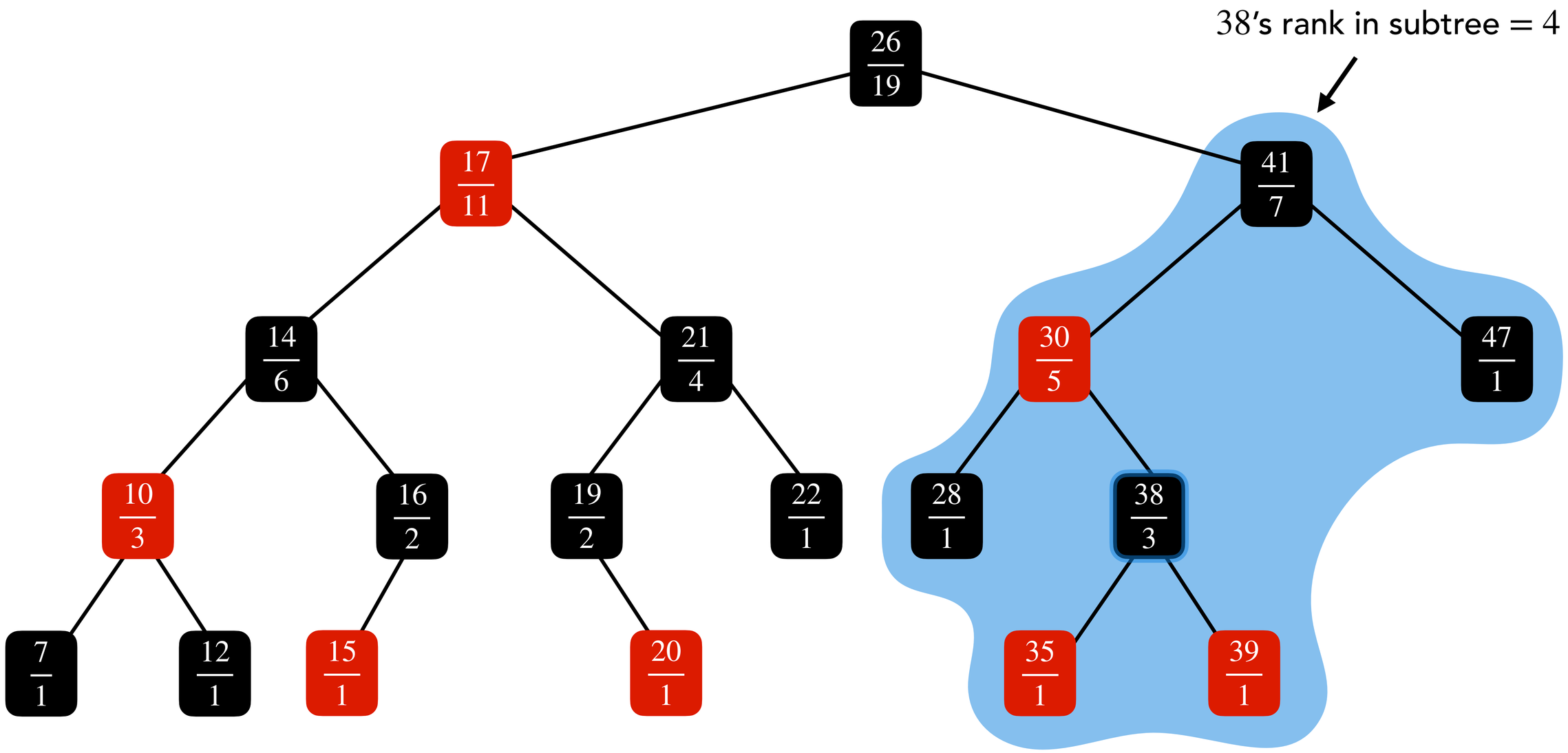
Finding The Rank of an Element



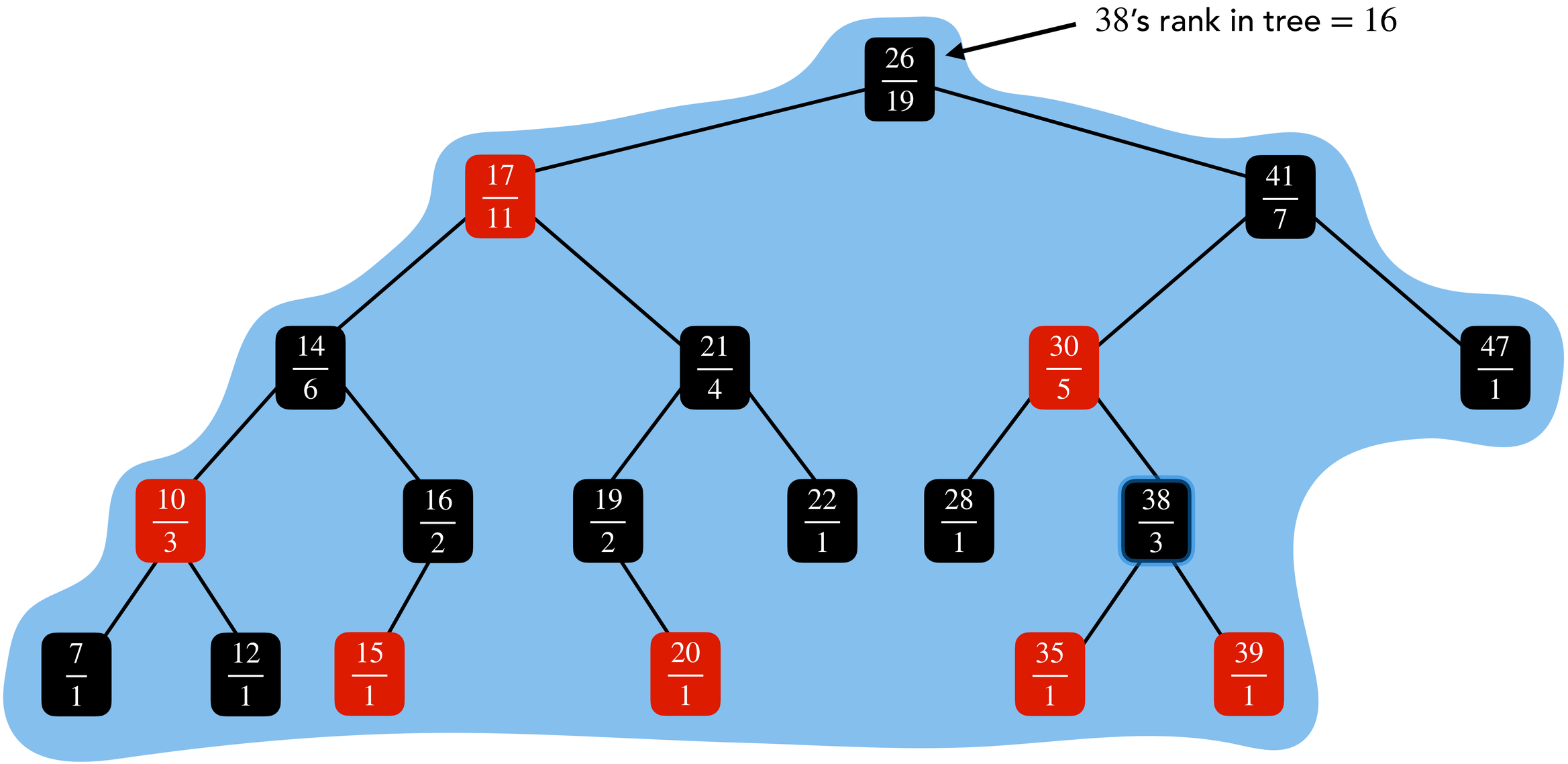
Finding The Rank of an Element



Finding The Rank of an Element



Finding The Rank of an Element



Finding The Rank of an Element

Finding The Rank of an Element

To find the rank of an element x in T call $\mathbf{Rank}(T, x)$.

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

$\textbf{Rank}(T, x)$:

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

Rank(T, x)**:**

1. $r = x\,.\,left\,.\,size + 1$

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

Rank(T, x)**:**

1. $r = x\,.\,left\,.\,size + 1$

x's rank within the subtree rooted at x

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

Rank(T, x)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. $y = x$

x's rank within the subtree rooted at x

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

Rank(T, x)**:**

1. $r = x\,.\,left\,.\,size + 1$
2. $y = x$
3. **while** $y \neq T\,.\,root$

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.   r = x.left.size + 1
2.   y = x
3.   while y ≠ T.root
4.       if y = y.p.right
```

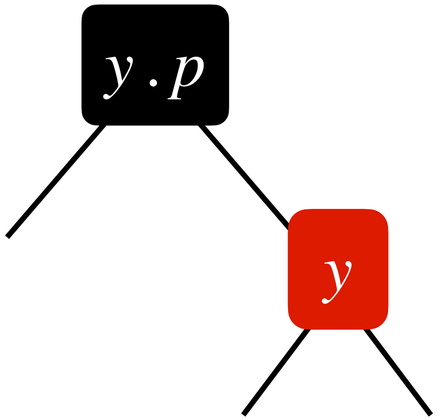
x's rank within the subtree rooted at x

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.   r = x.left.size + 1
2.   y = x
3.   while y ≠ T.root
4.       if y = y.p.right
```

x's rank within the subtree rooted at x

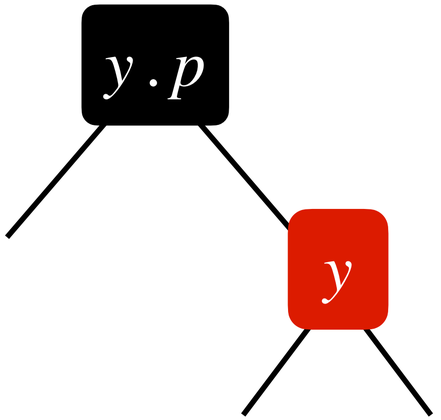


Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.  r = x.left.size + 1
2.  y = x
3.  while y ≠ T.root
4.      if y = y.p.right
```

x's rank within the subtree rooted at x



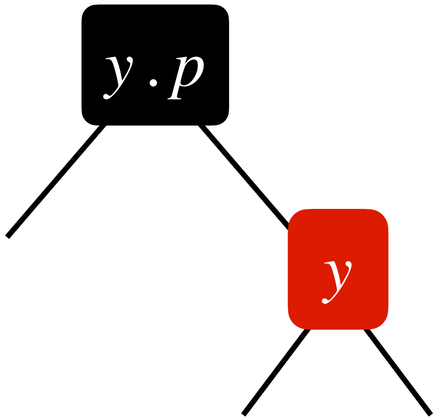
Updating y's rank when y is the right child of its parent.

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.   r = x.left.size + 1
2.   y = x
3.   while y ≠ T.root
4.       if y = y.p.right
5.           r = _____
```

x's rank within the subtree rooted at x



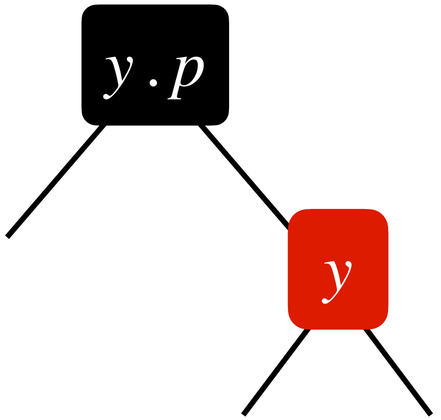
Updating y's rank when y is the right child of its parent.

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.  r = x.left.size + 1
2.  y = x
3.  while y ≠ T.root
4.      if y = y.p.right
5.          r = r + y.p.left.size + 1
```

x's rank within the subtree rooted at x



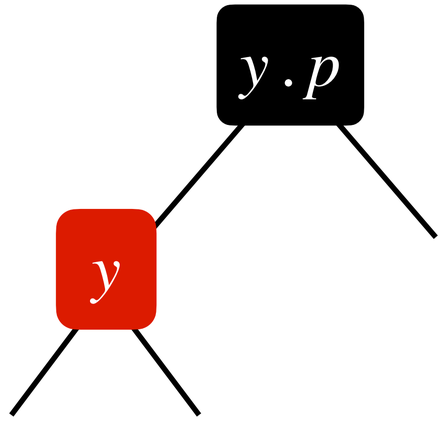
Updating y's rank when y is the right child of its parent.

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.  r = x.left.size + 1
2.  y = x
3.  while y ≠ T.root
4.      if y = y.p.right
5.          r = r + y.p.left.size + 1
```

x's rank within the subtree rooted at x



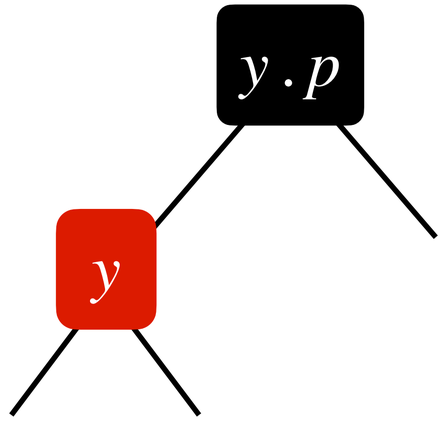
Updating y's rank when y is the right child of its parent.

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.  r = x.left.size + 1
2.  y = x
3.  while y ≠ T.root
4.      if y = y.p.right
5.          r = r + y.p.left.size + 1
6.      y = y.p
```

x's rank within the subtree rooted at x



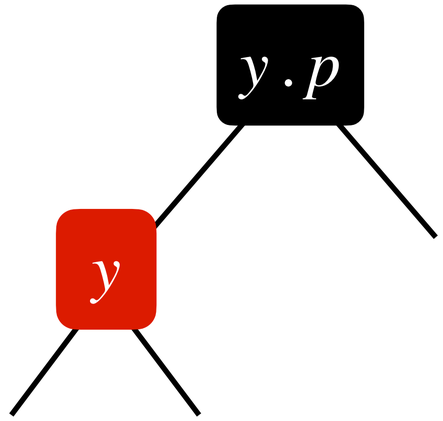
Updating y's rank when y is the right child of its parent.

Finding The Rank of an Element

To find the rank of an element x in T call $\textbf{Rank}(T, x)$.

```
Rank(T, x):
1.  r = x.left.size + 1
2.  y = x
3.  while y ≠ T.root
4.      if y = y.p.right
5.          r = r + y.p.left.size + 1
6.      y = y.p
7.  return r
```

x's rank within the subtree rooted at x



Updating y's rank when y is the right child of its parent.

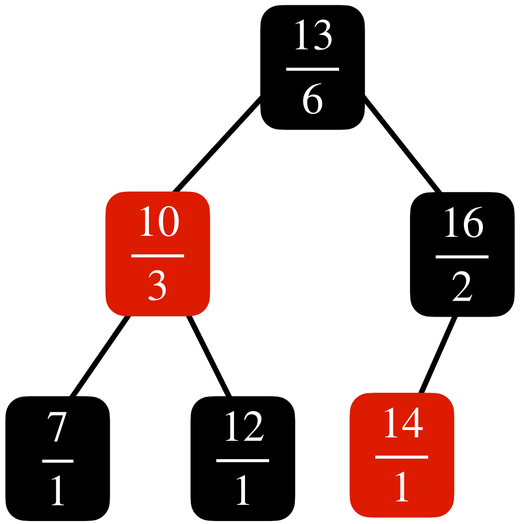
Maintaining Subtree Sizes: Insertion

Maintaining Subtree Sizes: Insertion

Insert 15 in this tree

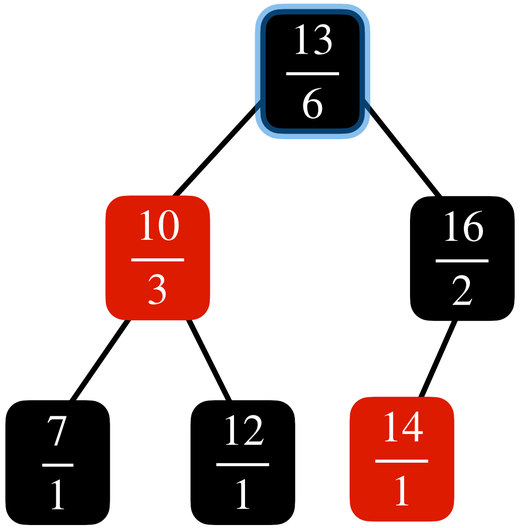
Maintaining Subtree Sizes: Insertion

Insert 15 in this tree



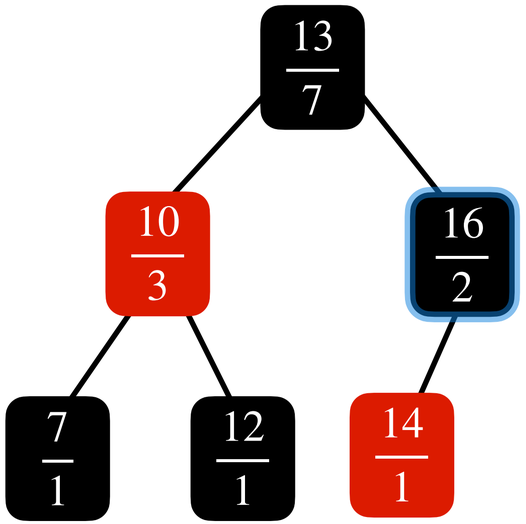
Maintaining Subtree Sizes: Insertion

Insert 15 in this tree



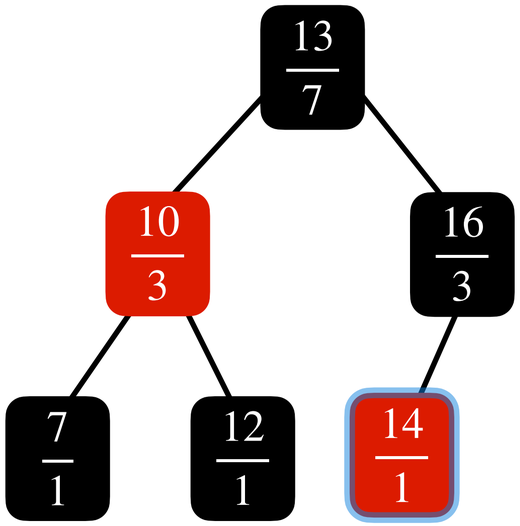
Maintaining Subtree Sizes: Insertion

Insert 15 in this tree



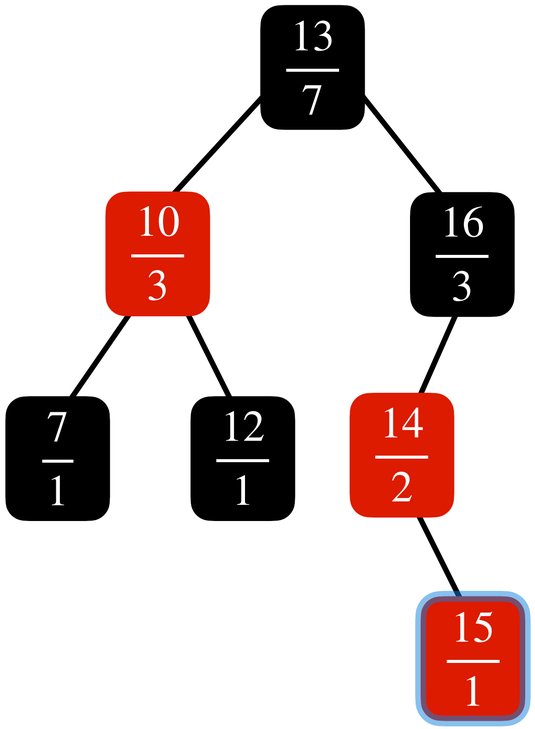
Maintaining Subtree Sizes: Insertion

Insert 15 in this tree



Maintaining Subtree Sizes: Insertion

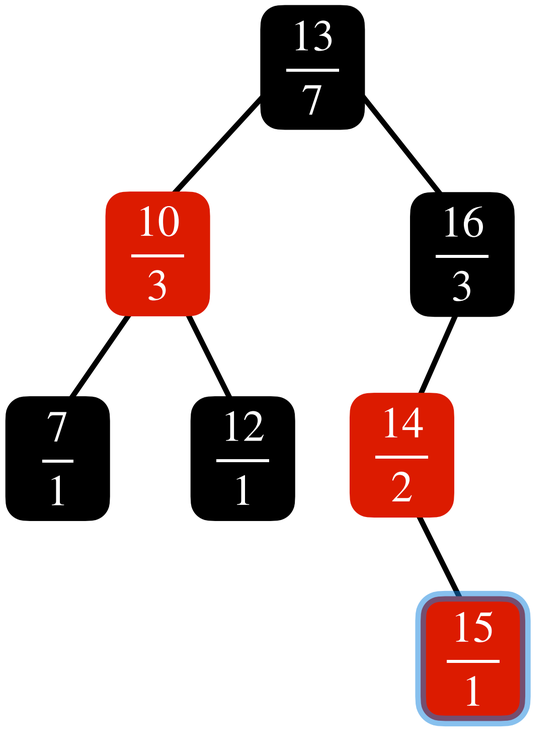
Insert 15 in this tree



Maintaining Subtree Sizes: Insertion

Insertion phase:

Insert 15 in this tree

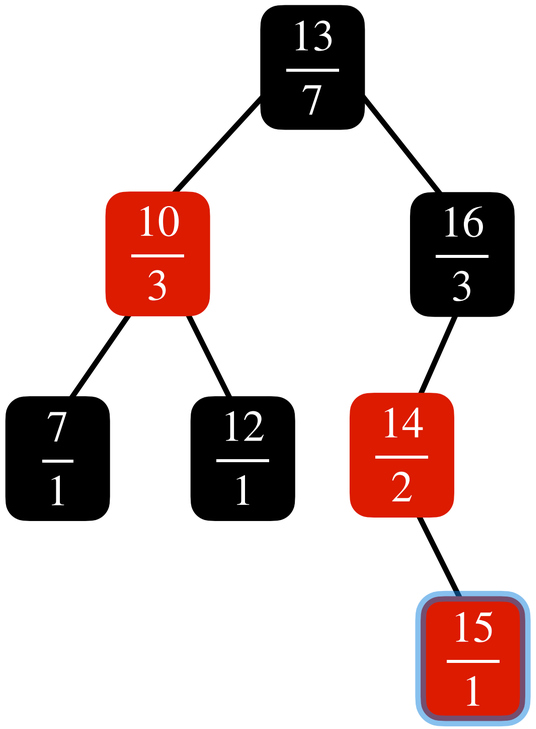


Maintaining Subtree Sizes: Insertion

Insertion phase:

Keep adding 1 to the **sizes** of **every node** we visit

Insert 15 in this tree

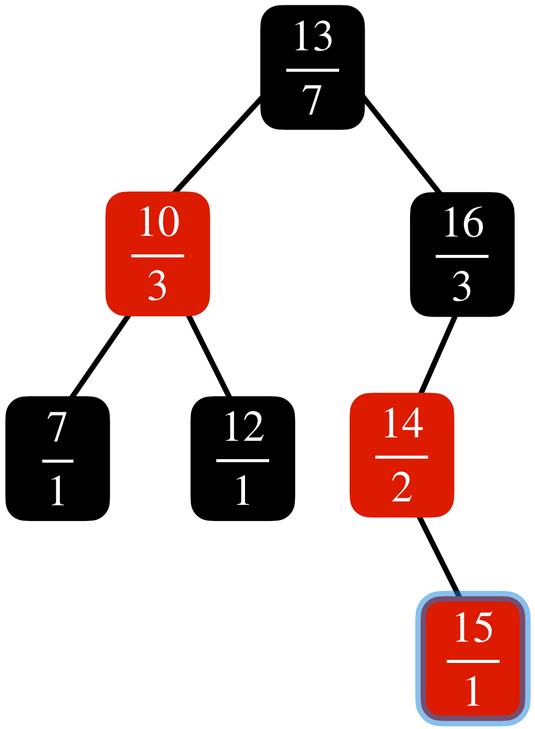


Maintaining Subtree Sizes: Insertion

Insertion phase:

Keep adding 1 to the **sizes** of **every node** we visit while **searching** for the correct leaf

Insert 15 in this tree

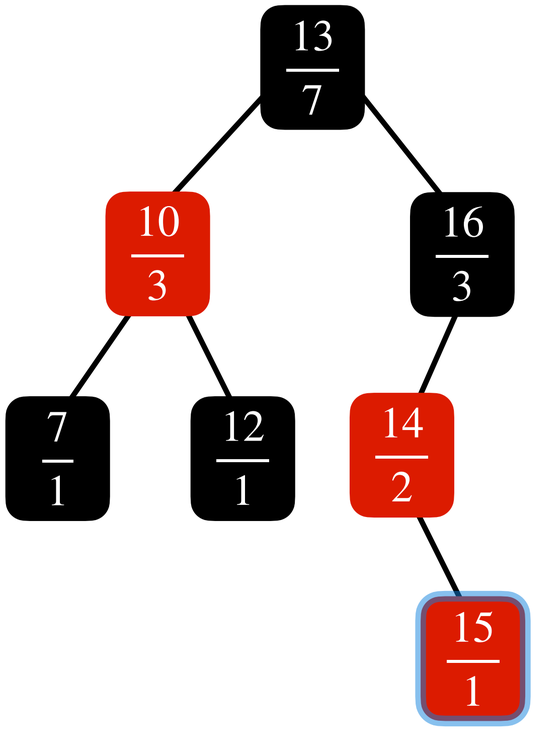


Maintaining Subtree Sizes: Insertion

Insertion phase:

Keep adding 1 to the **sizes** of **every node** we visit while **searching** for the correct leaf where new node can be inserted.

Insert 15 in this tree



Maintaining Subtree Sizes: Insertion

Maintaining Subtree Sizes: Insertion

Fix-up phase:

Maintaining Subtree Sizes: Insertion

Fix-up phase:

- Fix-ups involve only **rotations** and **recolouring**.

Maintaining Subtree Sizes: Insertion

Fix-up phase:

- Fix-ups involve only **rotations** and **recolouring**.
- **Recolouring** doesn't require changing sizes.

Maintaining Subtree Sizes: Insertion

Fix-up phase:

- Fix-ups involve only **rotations** and **recolouring**.
- **Recolouring** doesn't require changing sizes.
- During **rotations** size changes are doable in constant time.

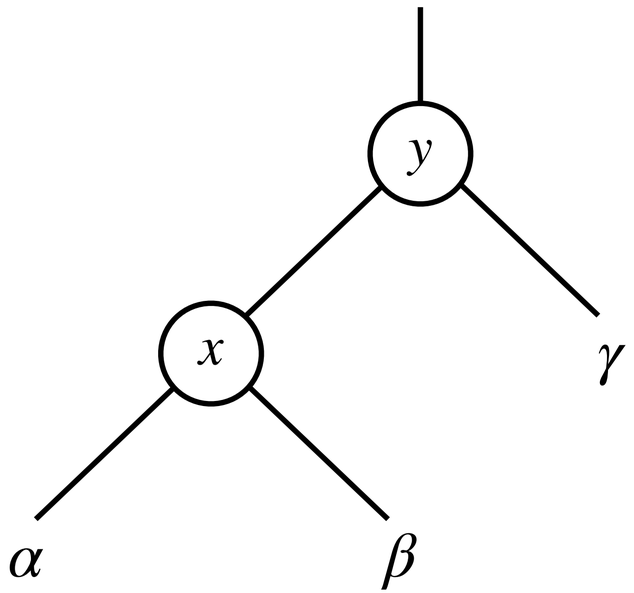
Maintaining Subtree Sizes: Rotations

Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.

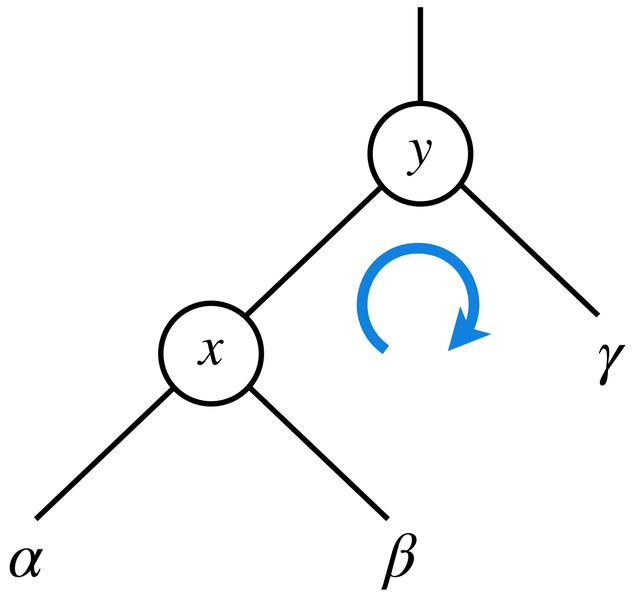
Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



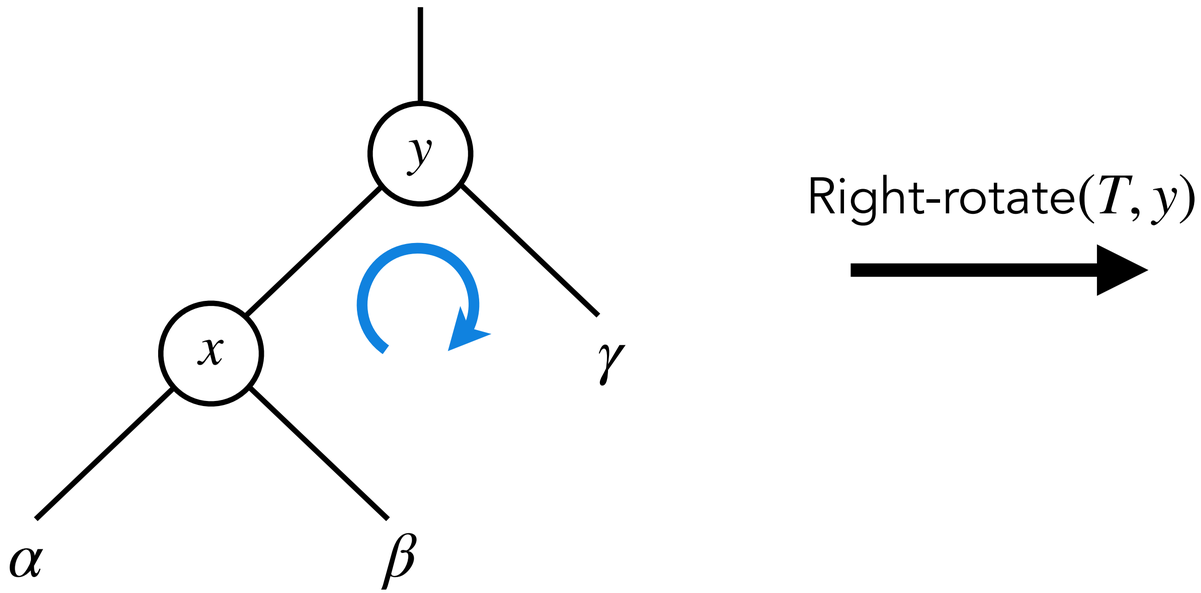
Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



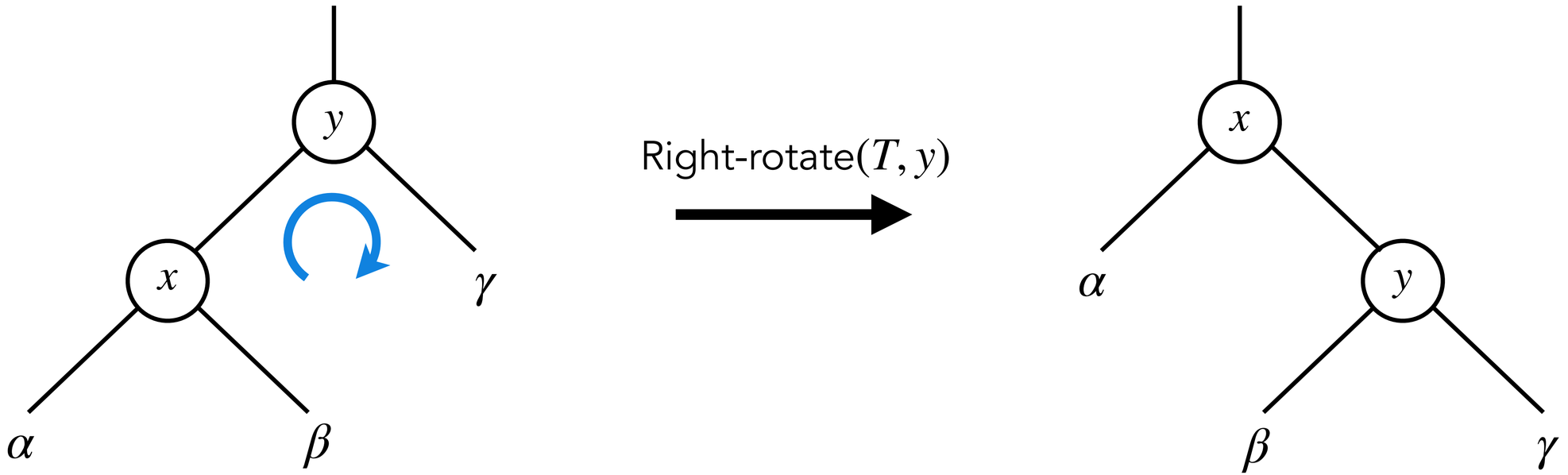
Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



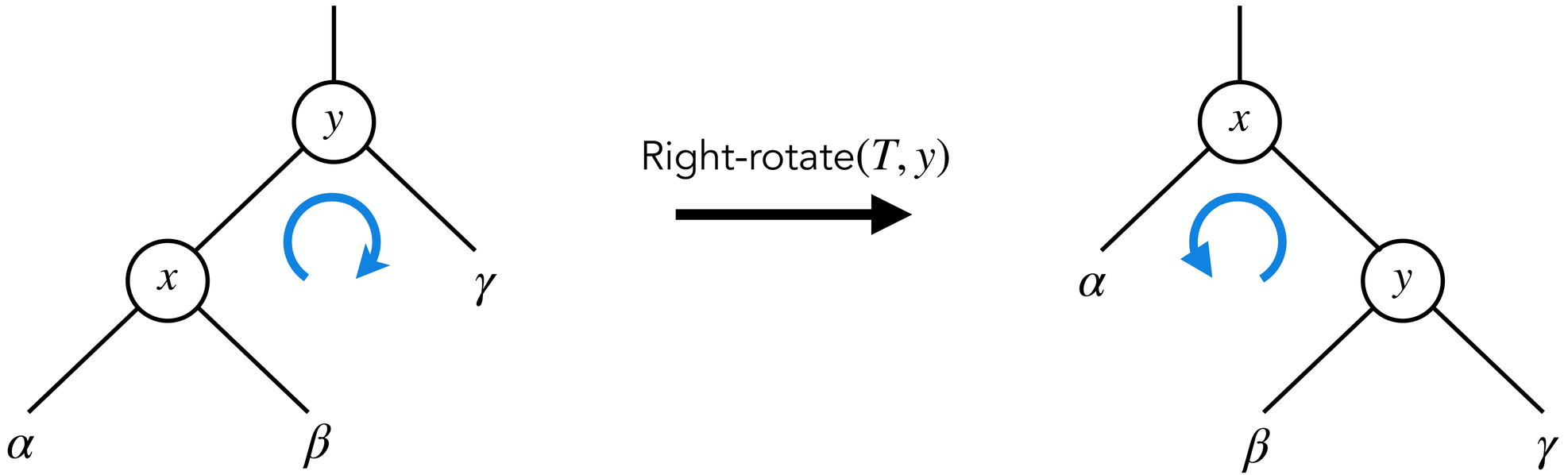
Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



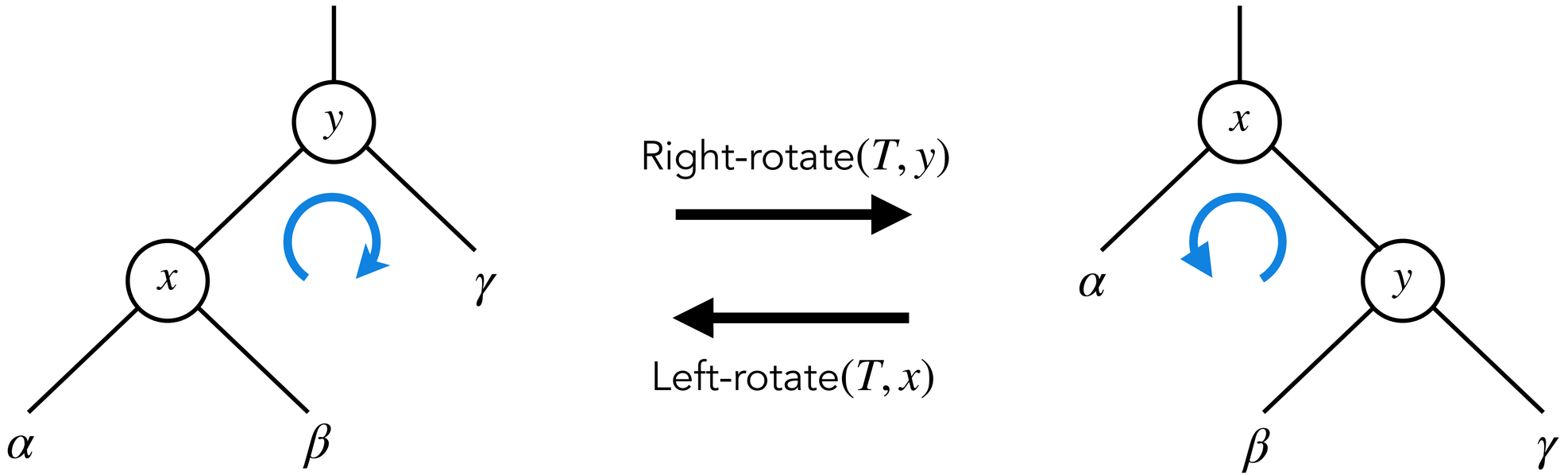
Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



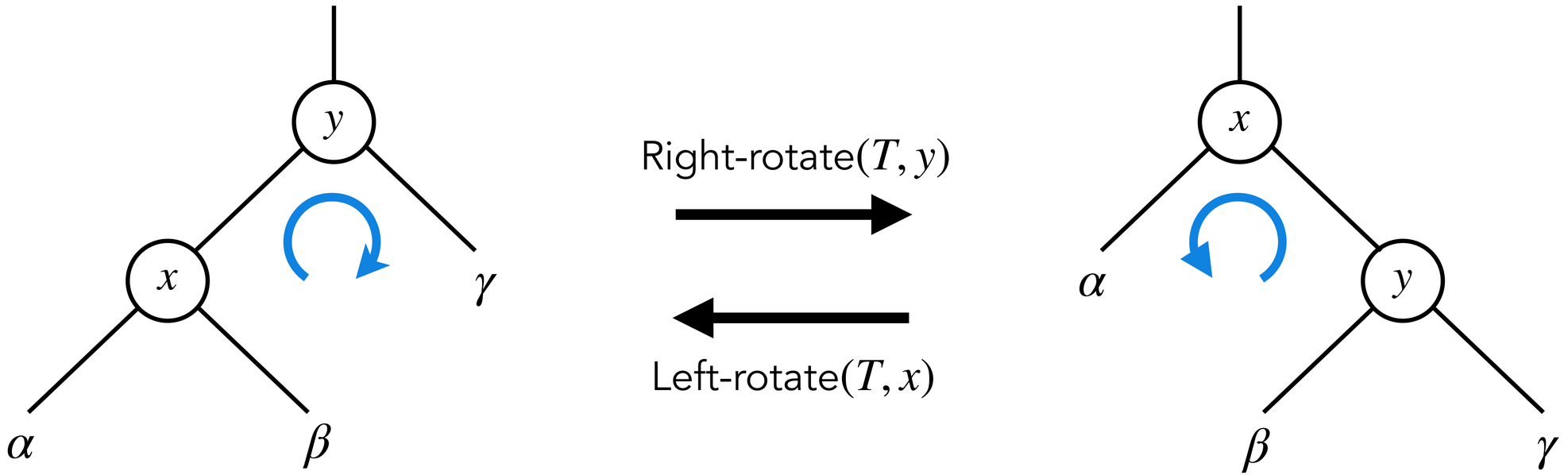
Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



Maintaining Subtree Sizes: Rotations

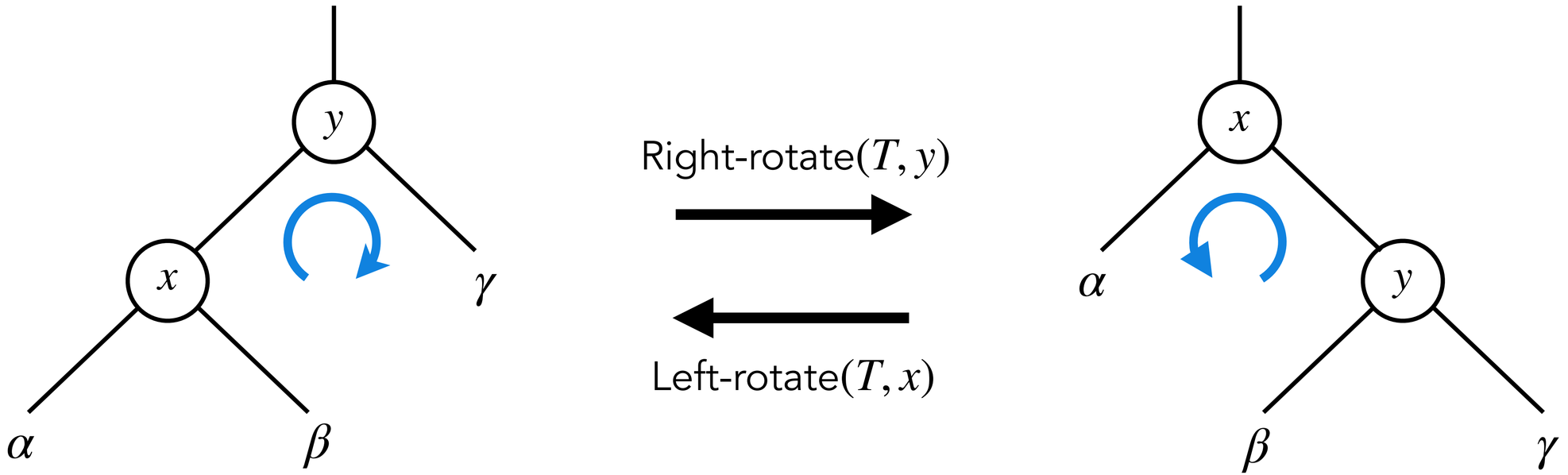
Maintaining subtree sizes during rotations is easy.



After **Left-rotate**(T, x), do the following:

Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.

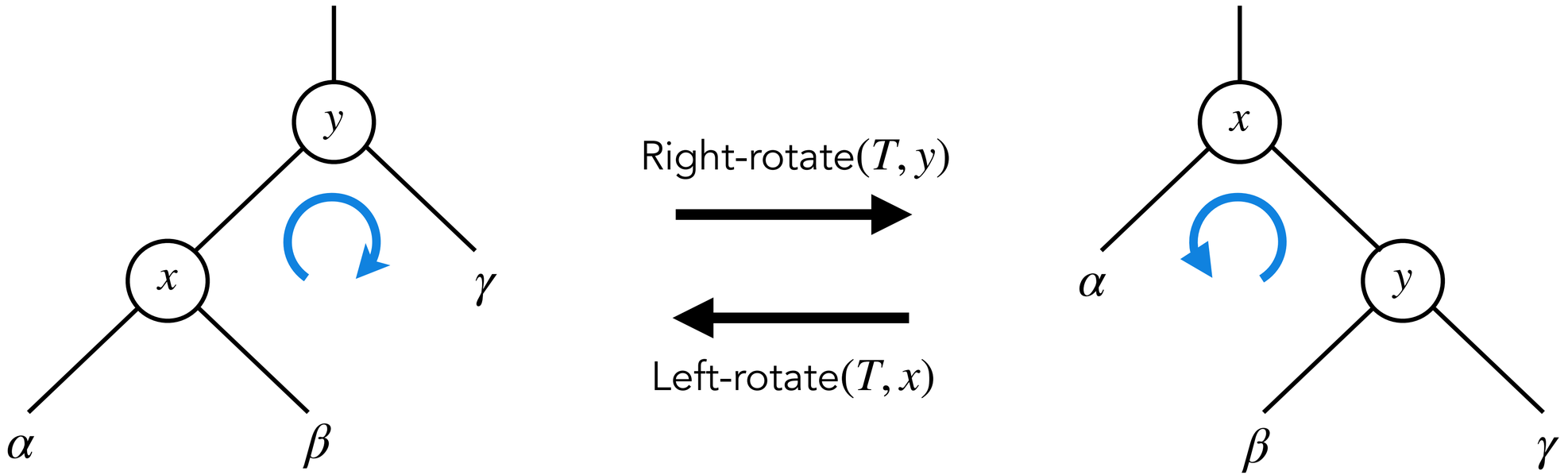


After **Left-rotate**(T, x), do the following:

1) $y\,.\,size =$

Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



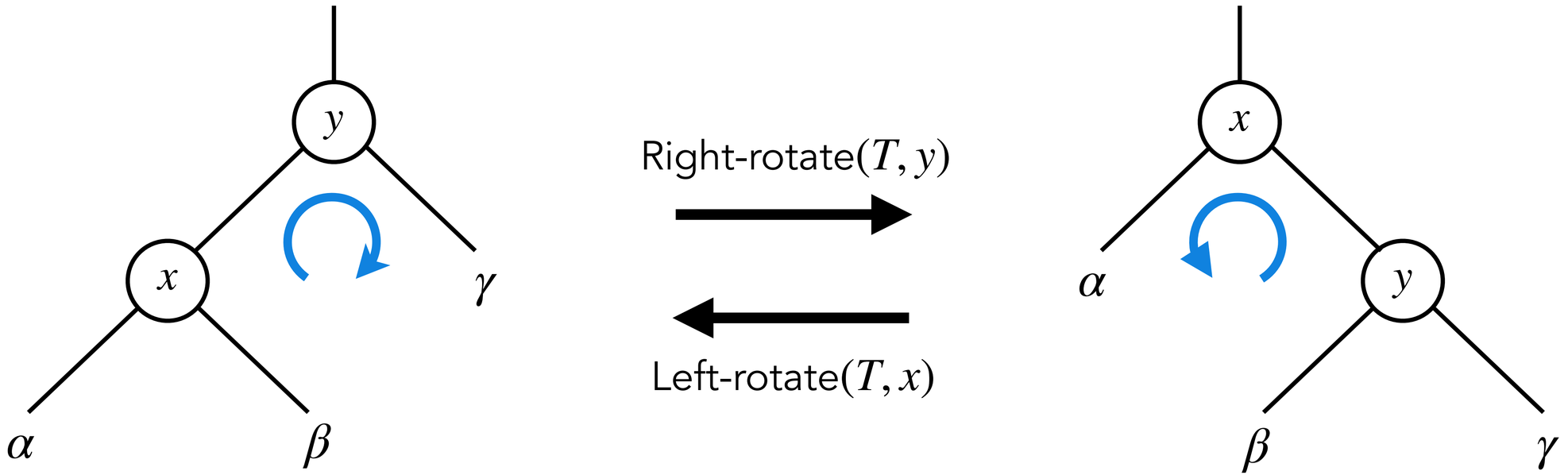
After **Left-rotate**(T, x), do the following:

1) $y\,.\,size =$

2) $x\,.\,size =$

Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.



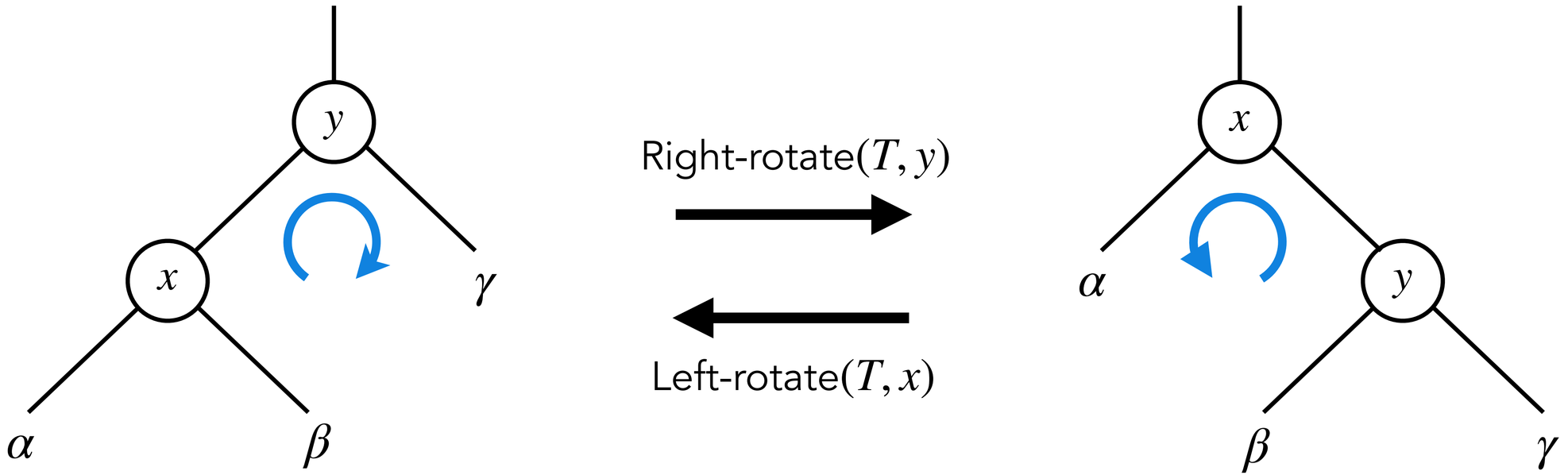
After **Left-rotate**(T, x), do the following:

1) $y\,.\,size = x\,.\,size$

2) $x\,.\,size =$

Maintaining Subtree Sizes: Rotations

Maintaining subtree sizes during rotations is easy.

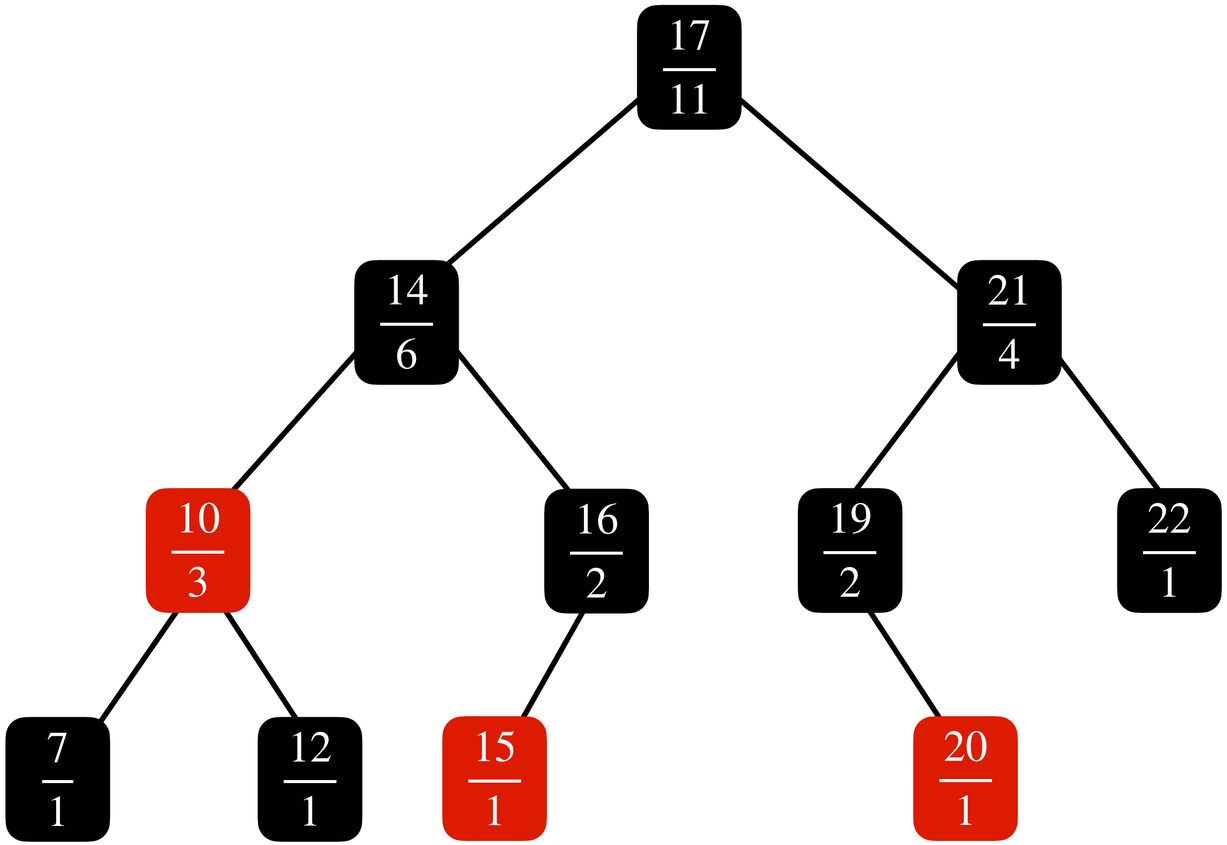


After **Left-rotate**(T, x), do the following:

1) $y\,.\,size = x\,.\,size$

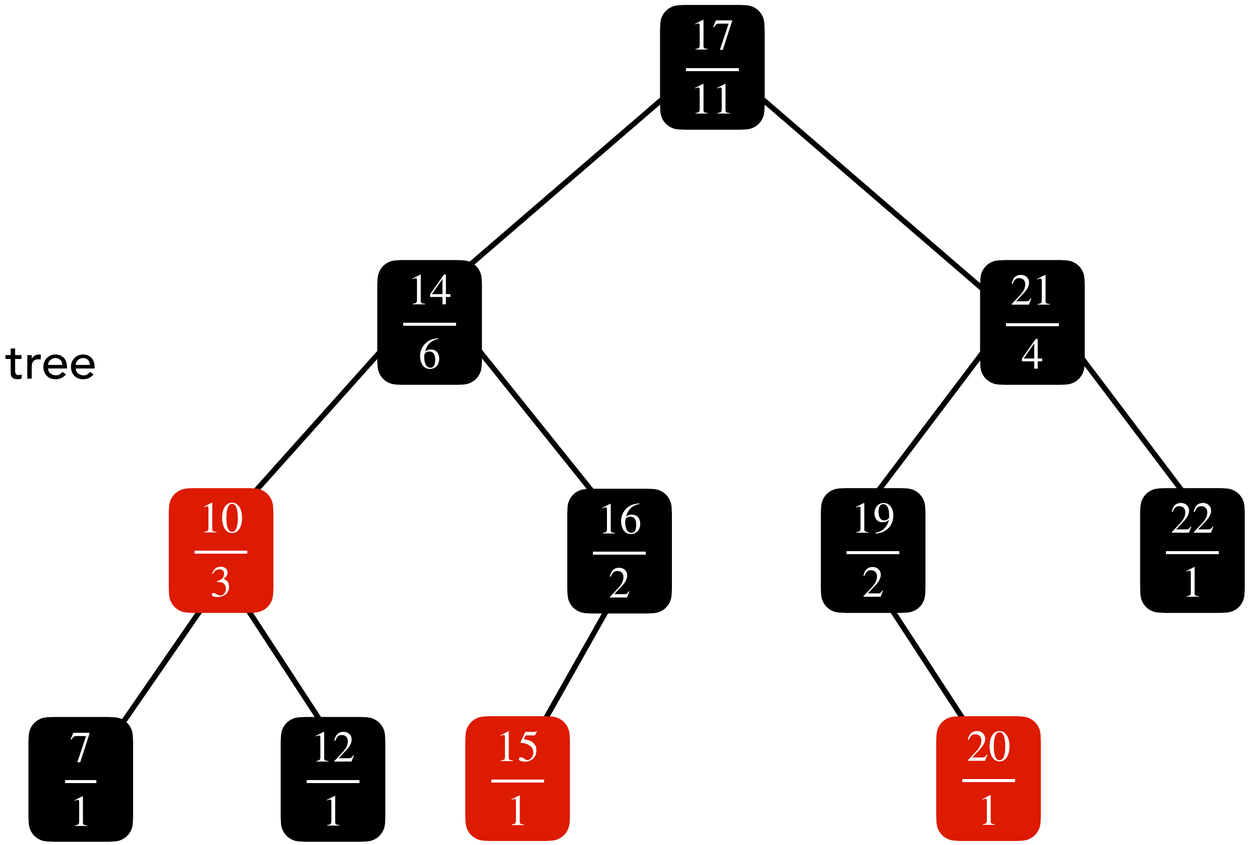
2) $x\,.\,size = x\,.\,left\,.\,size + x\,.\,right\,.\,size + 1$

Maintaining Subtree Sizes: Deletion



Maintaining Subtree Sizes: Deletion

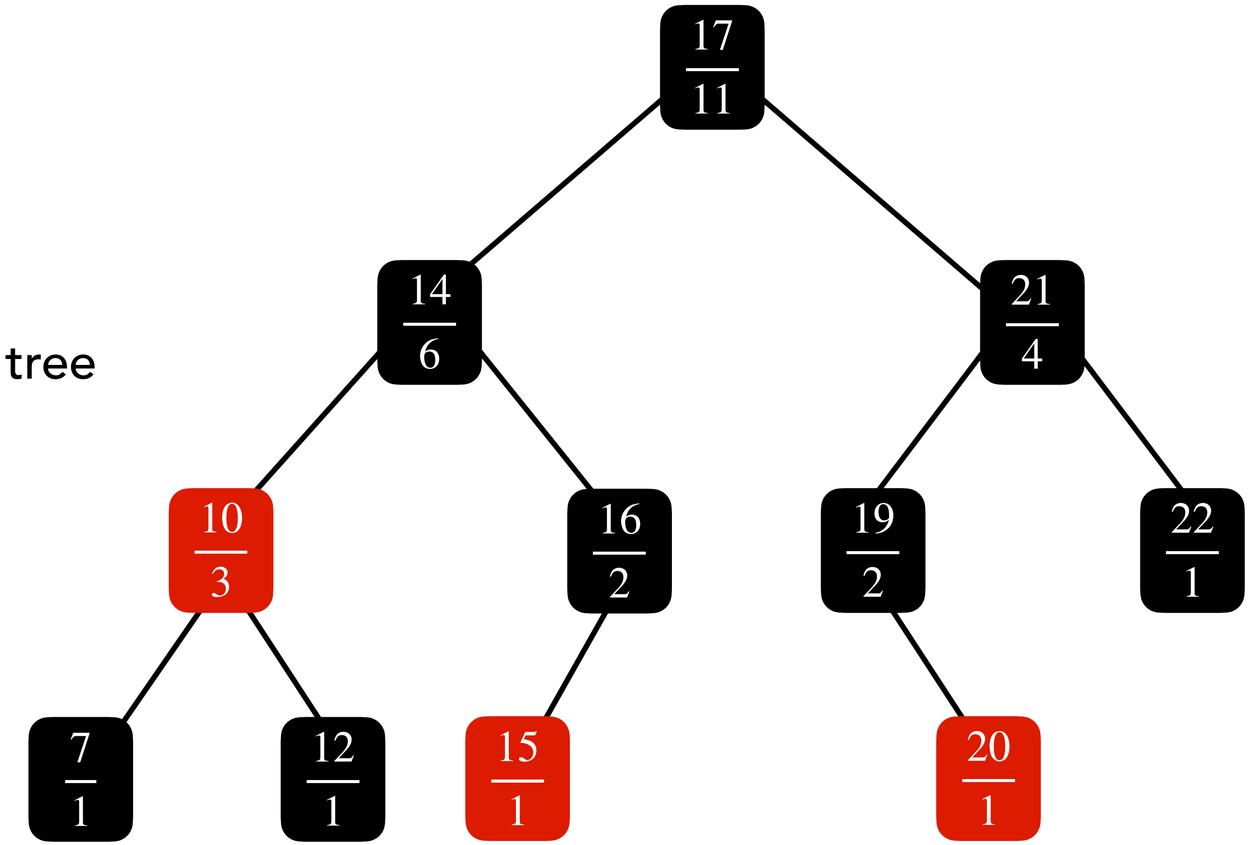
Delete 20 in this tree



Maintaining Subtree Sizes: Deletion

Deletion phase:

Delete 20 in this tree

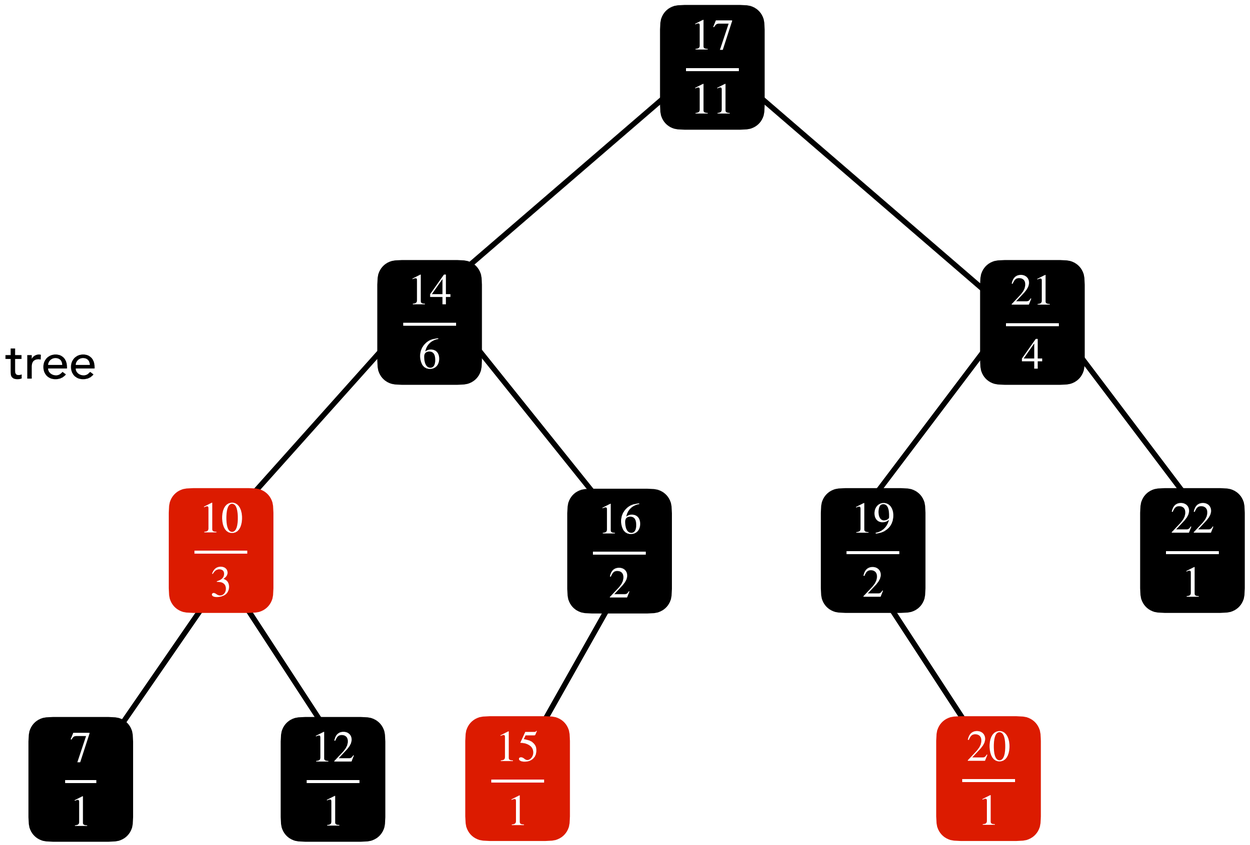


Maintaining Subtree Sizes: Deletion

Deletion phase:

Keep subtracting 1 from the **sizes** of **every node** from the parent of the removed node to

Delete 20 in this tree

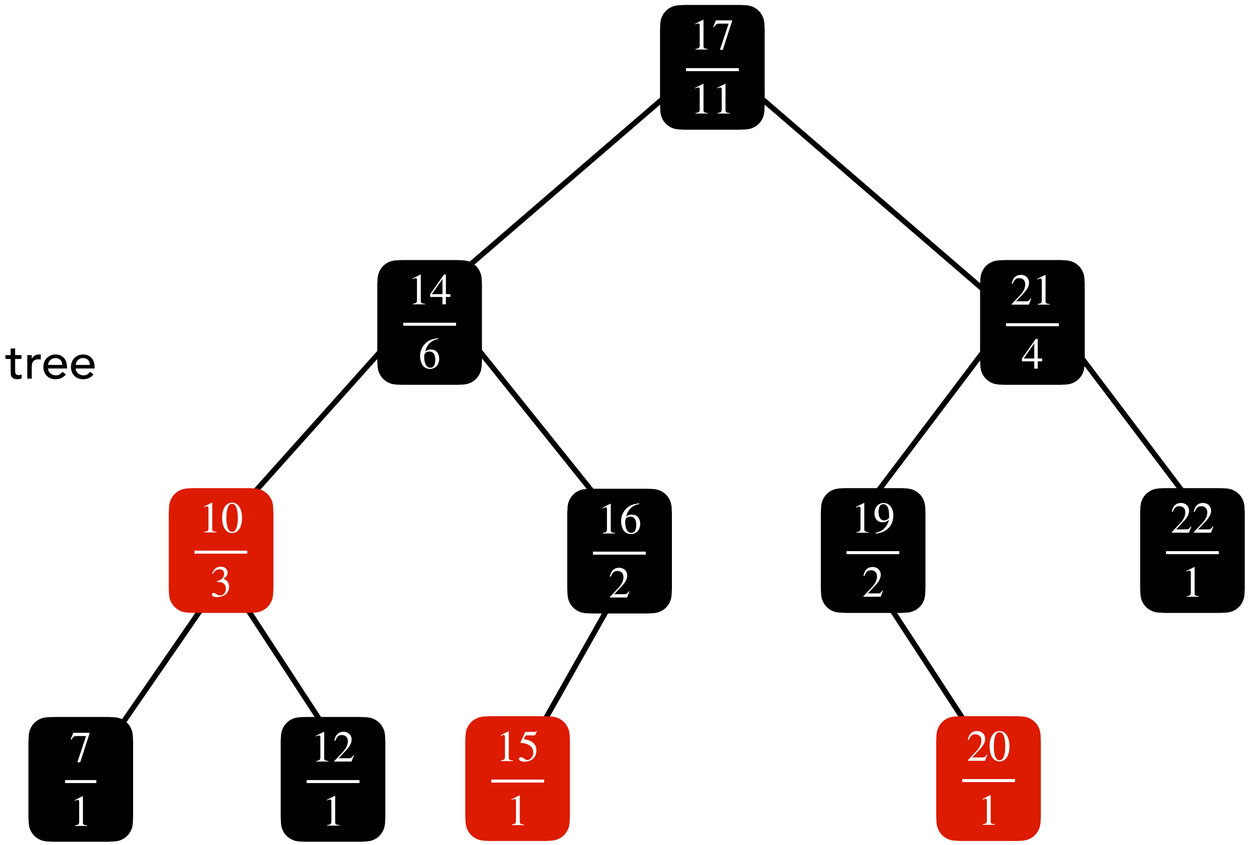


Maintaining Subtree Sizes: Deletion

Deletion phase:

Keep subtracting 1 from the **sizes** of **every node** from the parent of the removed node to the root.

Delete 20 in this tree

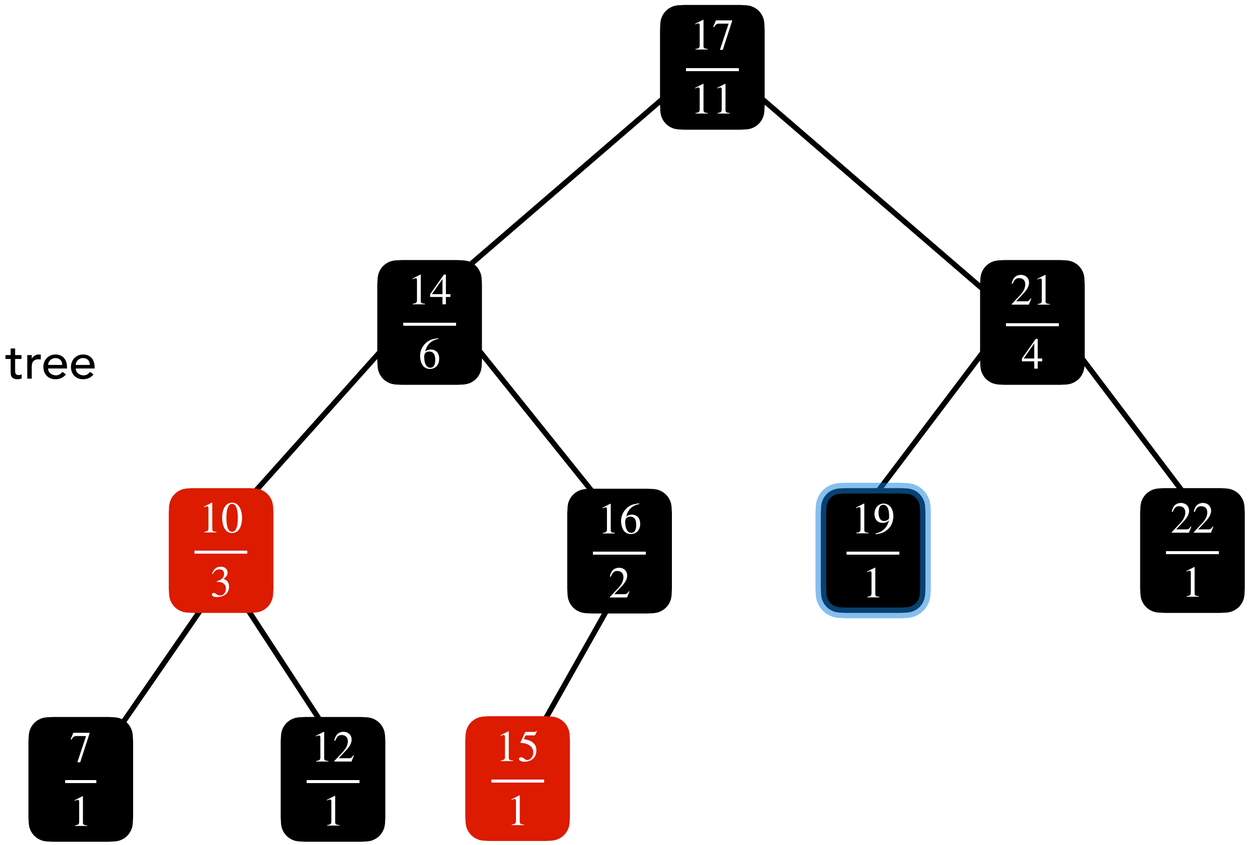


Maintaining Subtree Sizes: Deletion

Deletion phase:

Keep subtracting 1 from the **sizes** of **every node** from the parent of the removed node to the root.

Delete 20 in this tree

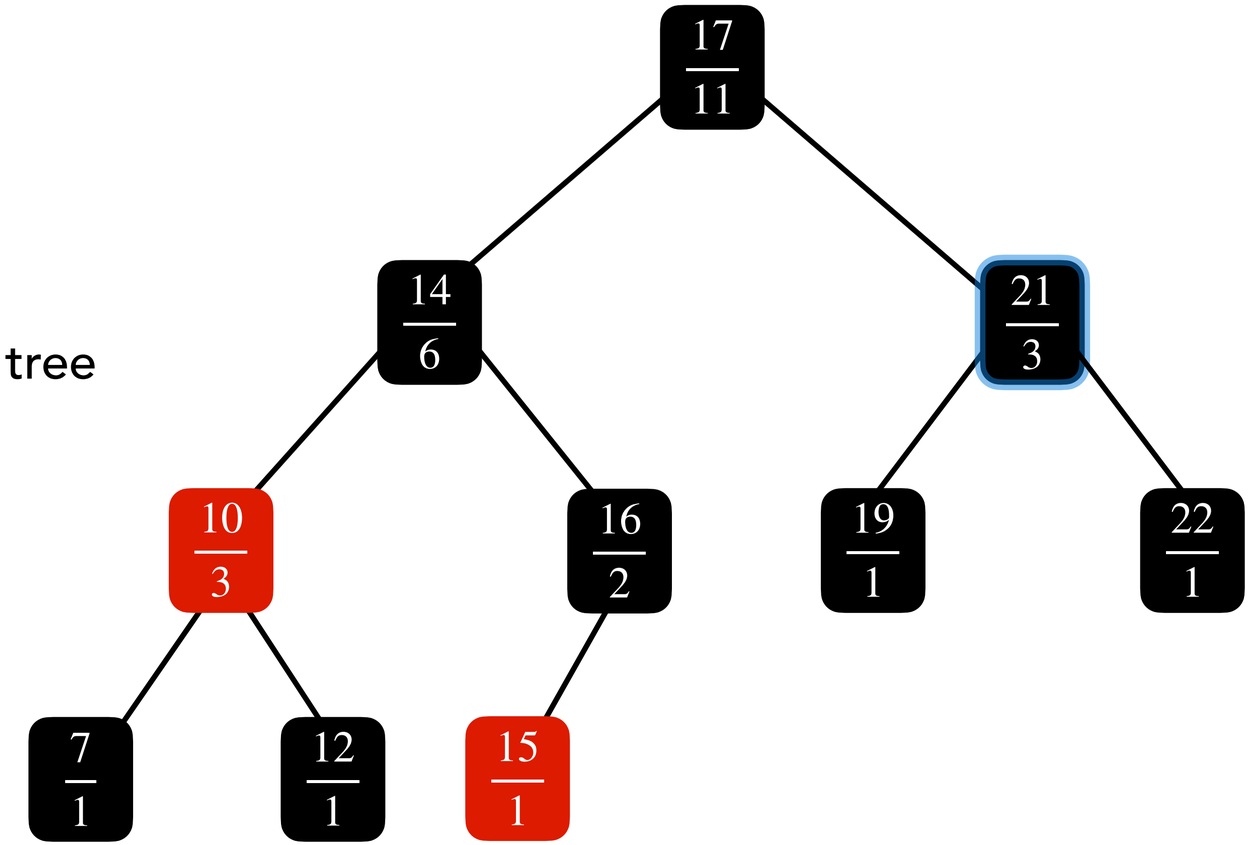


Maintaining Subtree Sizes: Deletion

Deletion phase:

Keep subtracting 1 from the **sizes** of **every node** from the parent of the removed node to the root.

Delete 20 in this tree

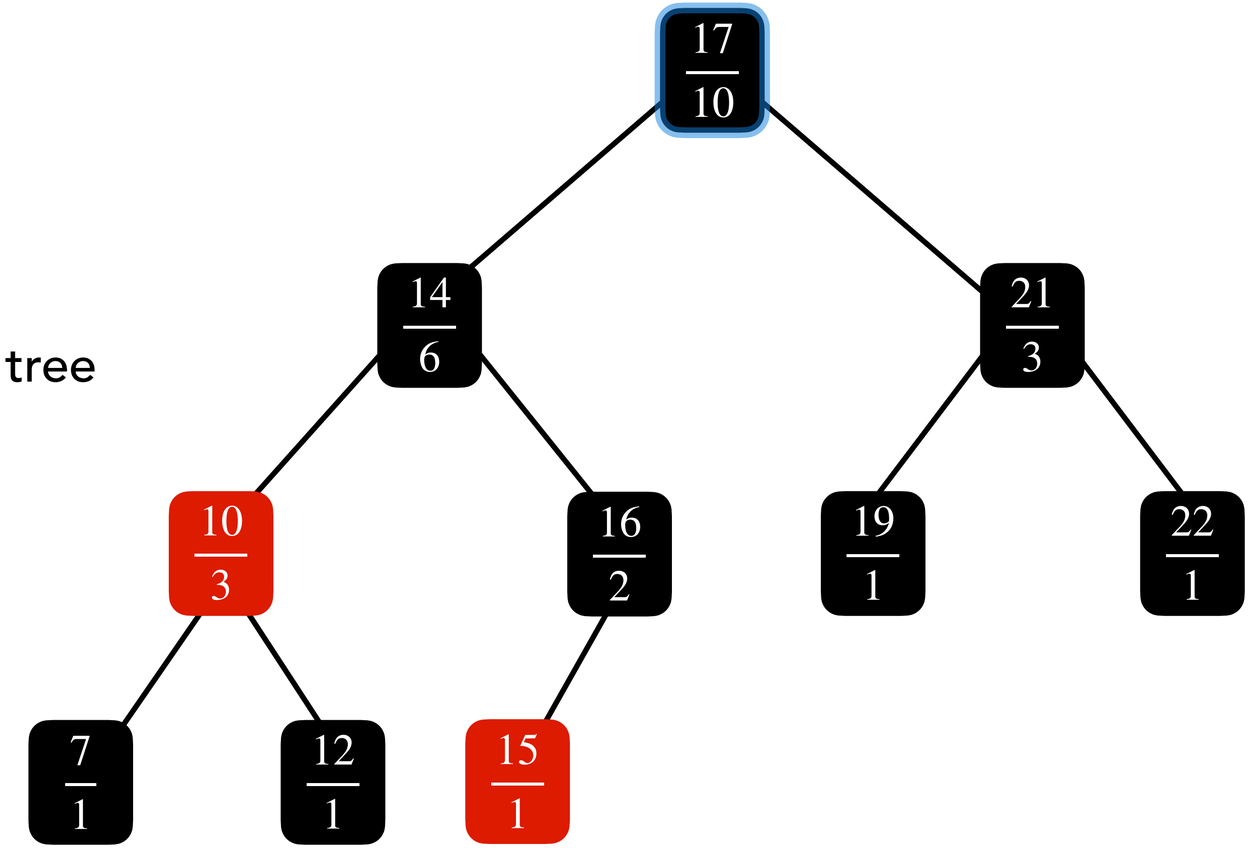


Maintaining Subtree Sizes: Deletion

Deletion phase:

Keep subtracting 1 from the **sizes** of **every node** from the parent of the removed node to the root.

Delete 20 in this tree



Maintaining Subtree Sizes: Deletion

Maintaining Subtree Sizes: Deletion

Fix-up phase:

Maintaining Subtree Sizes: Deletion

Fix-up phase:

- Fix-ups involve only **rotations** and **recolouring**.

Maintaining Subtree Sizes: Deletion

Fix-up phase:

- Fix-ups involve only **rotations** and **recolouring**.
- **Recolouring** doesn't require changing sizes.

Maintaining Subtree Sizes: Deletion

Fix-up phase:

- Fix-ups involve only **rotations** and **recolouring**.
- **Recolouring** doesn't require changing sizes.
- During **rotations** size changes are doable in constant time.