Lecture 10

Interval Trees

Source: Introduction to Algorithms, CLRS

Interval

Interval

Defn: An interval $[t_1, t_2]$ is an object i such that:

Interval

Defn: An interval $[t_1, t_2]$ is an object i such that:

- t_1 and t_2 are integers such that $t_1 \leq t_2$.

Interval

Defn: An interval $[t_1, t_2]$ is an object i such that:

- t_1 and t_2 are integers such that $t_1 \leq t_2$.
- $i.low = t_1$ and $i.high = t_2$.

Interval

Defn: An interval $[t_1, t_2]$ is an object i such that:

- t_1 and t_2 are integers such that $t_1 \leq t_2$.
- $i.low = t_1$ and $i.high = t_2$.

Defn: We say intervals i and i' overlap if $i \cap i' \neq \phi$.

Interval

Defn: An interval $[t_1, t_2]$ is an object i such that:

- t_1 and t_2 are integers such that $t_1 \leq t_2$.
- $i.low = t_1$ and $i.high = t_2$.

Defn: We say intervals i and i' overlap if $i \cap i' \neq \phi$.

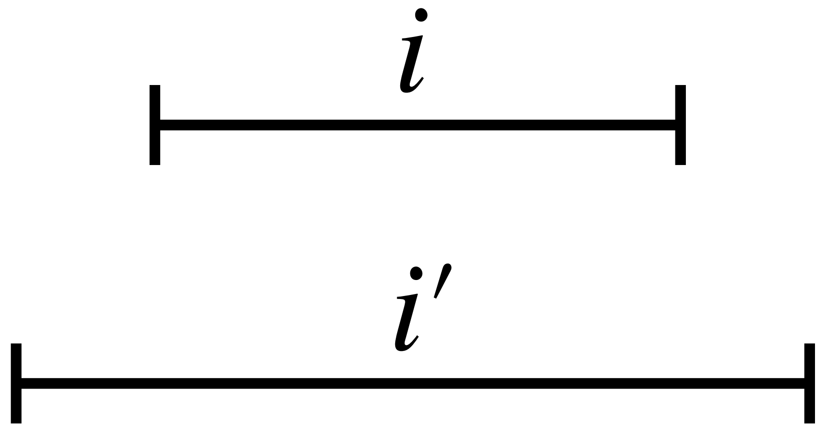
Example: $[5,8]$ and $[6,9]$ are overlapping. $[3,5]$ and $[7,10]$ are non-overlapping.

Testing Overlapping of Intervals

Testing Overlapping of Intervals

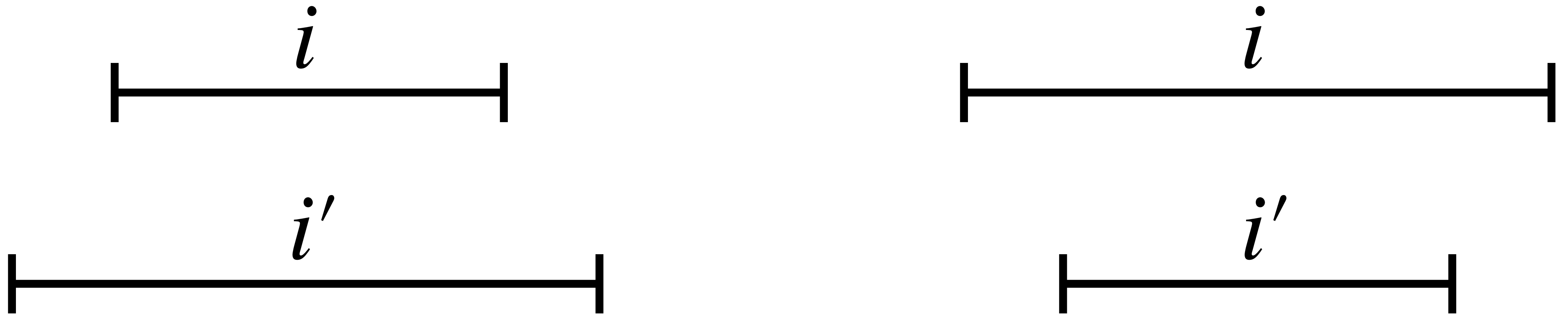
Overlapping intervals

Testing Overlapping of Intervals



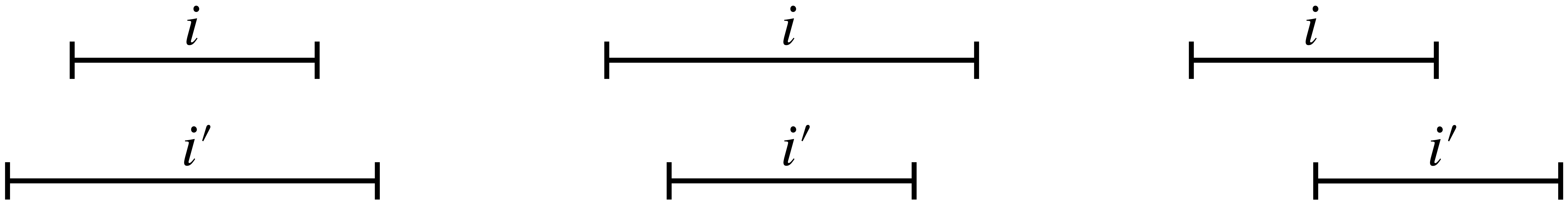
Overlapping intervals

Testing Overlapping of Intervals



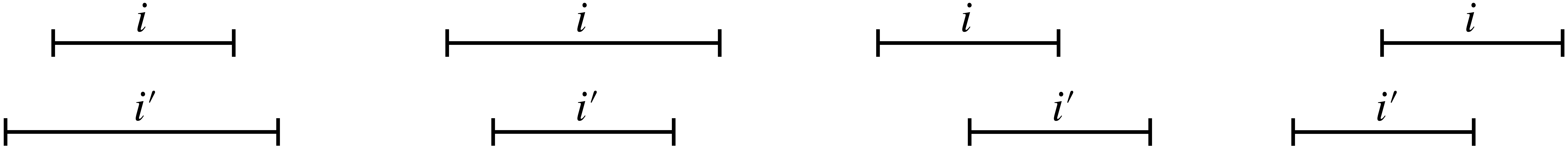
Overlapping intervals

Testing Overlapping of Intervals



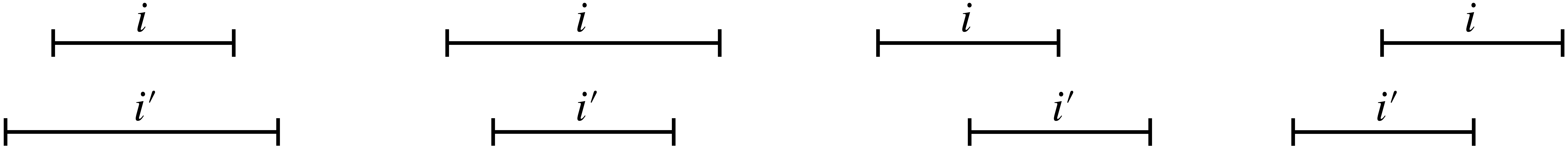
Overlapping intervals

Testing Overlapping of Intervals



Overlapping intervals

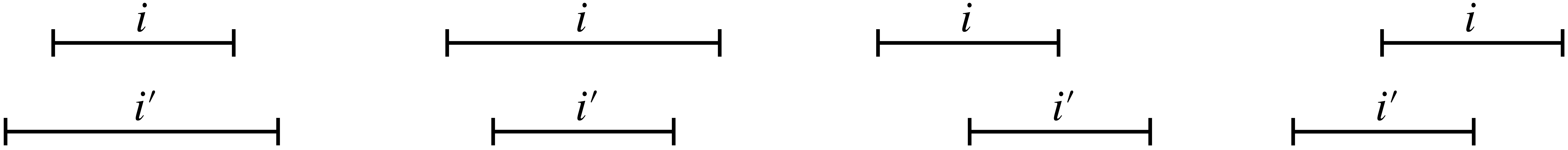
Testing Overlapping of Intervals



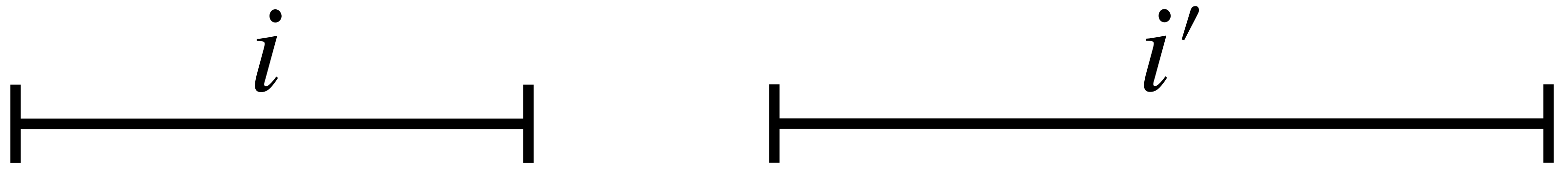
Overlapping intervals

Non-overlapping intervals

Testing Overlapping of Intervals

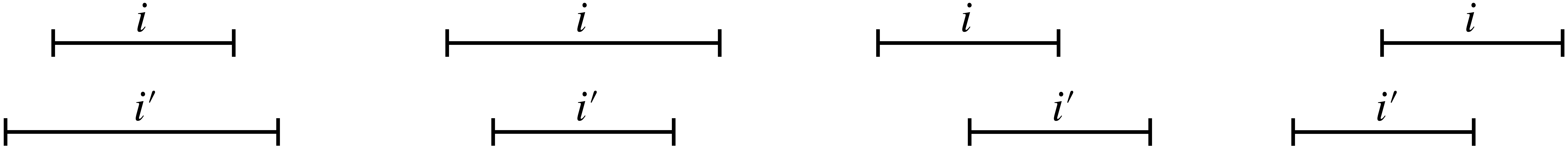


Overlapping intervals

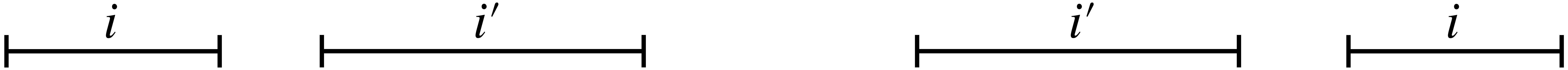


Non-overlapping intervals

Testing Overlapping of Intervals



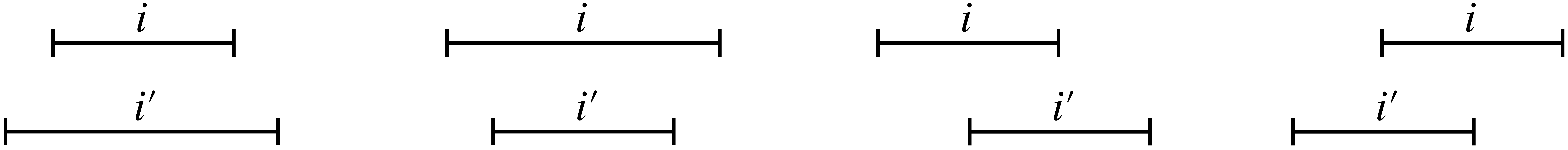
Overlapping intervals



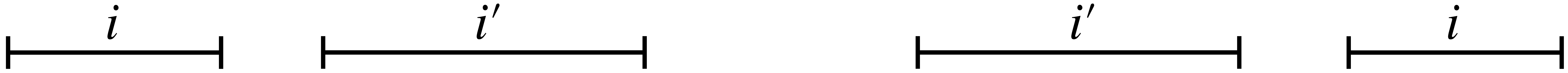
Non-overlapping intervals

Testing Overlapping of Intervals

Defn: Two intervals i and i' do not overlap if and only if $i\,.\,high < i'\,.\,low$ or $i'\,.\,high < i\,.\,low$.



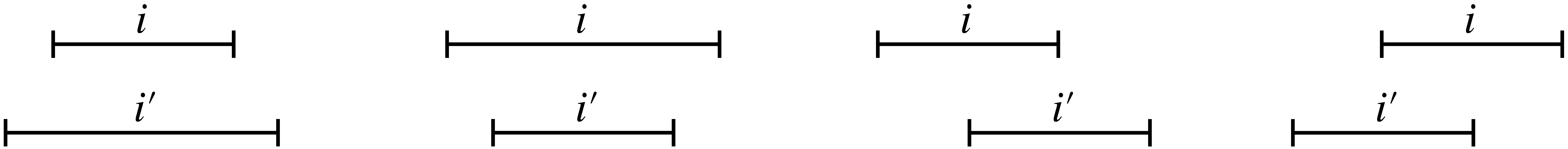
Overlapping intervals



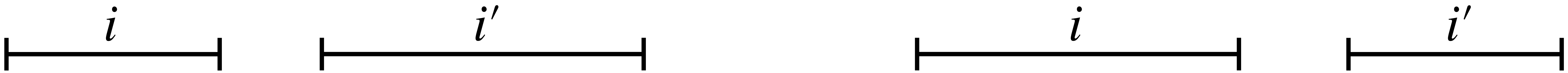
Non-overlapping intervals

Testing Overlapping of Intervals

Defn: Two intervals i and i' overlap if and only if $i\,.\,low \leq i'\,.\,high$ and $i'\,.\,low \leq i\,.\,high$.



Overlapping intervals



Non-overlapping intervals

Interval Trees

Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x

Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x contains an interval $x.int$.

Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x contains an interval $x\,.\,int$. Interval tree supports the following operations:

Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x contains an interval $x\,.\,int$. Interval tree supports the following operations:

- Interval-Insert(T, x)

Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x contains an interval $x\,.\,int$. Interval tree supports the following operations:

- Interval-Insert(T, x)
- Interval-Delete(T, x)

Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x contains an interval $x\,.\,int$. Interval tree supports the following operations:

- Interval-Insert(T, x)
- Interval-Delete(T, x)
- Interval-Search(T, i)

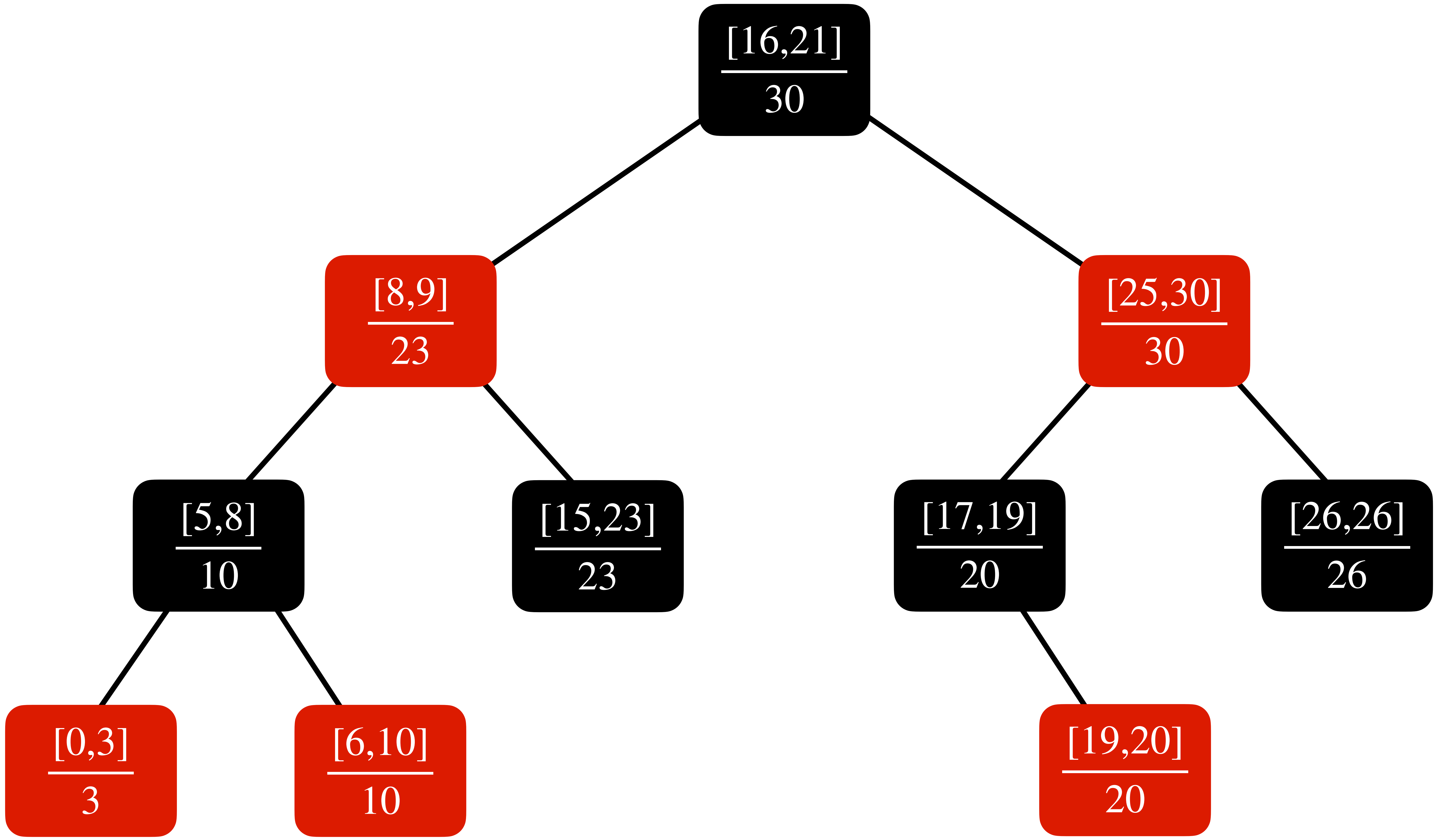
Interval Trees

Interval Trees is a form of RB-tree used to maintain a dynamic set, where every element x contains an interval $x\,.\,int$. Interval tree supports the following operations:

- Interval-Insert(T, x)
- Interval-Delete(T, x)
- Interval-Search(T, i)

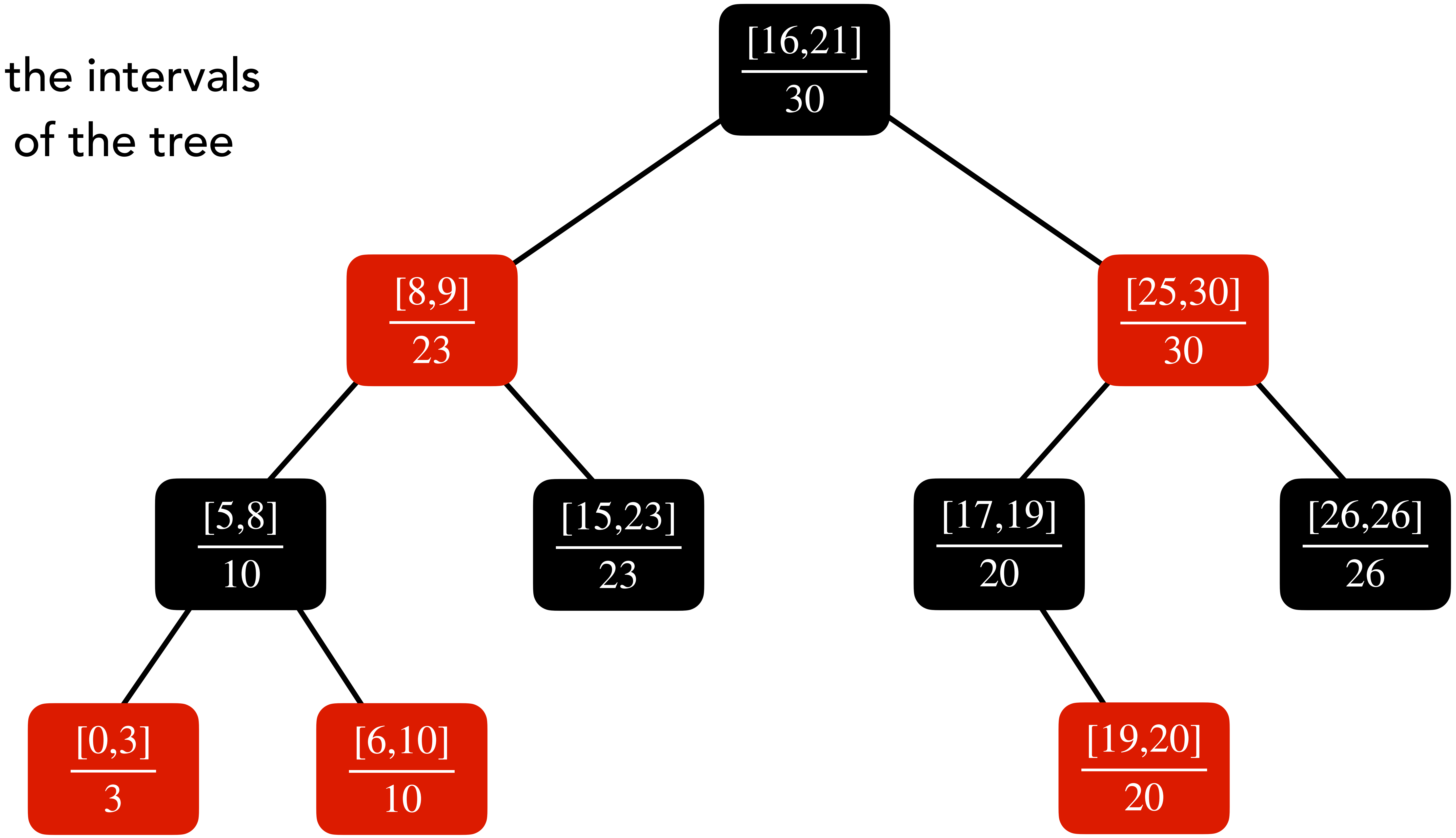
Returns an element x of T such that $x\,.\,int$ overlaps with i.

Example of an Interval Tree

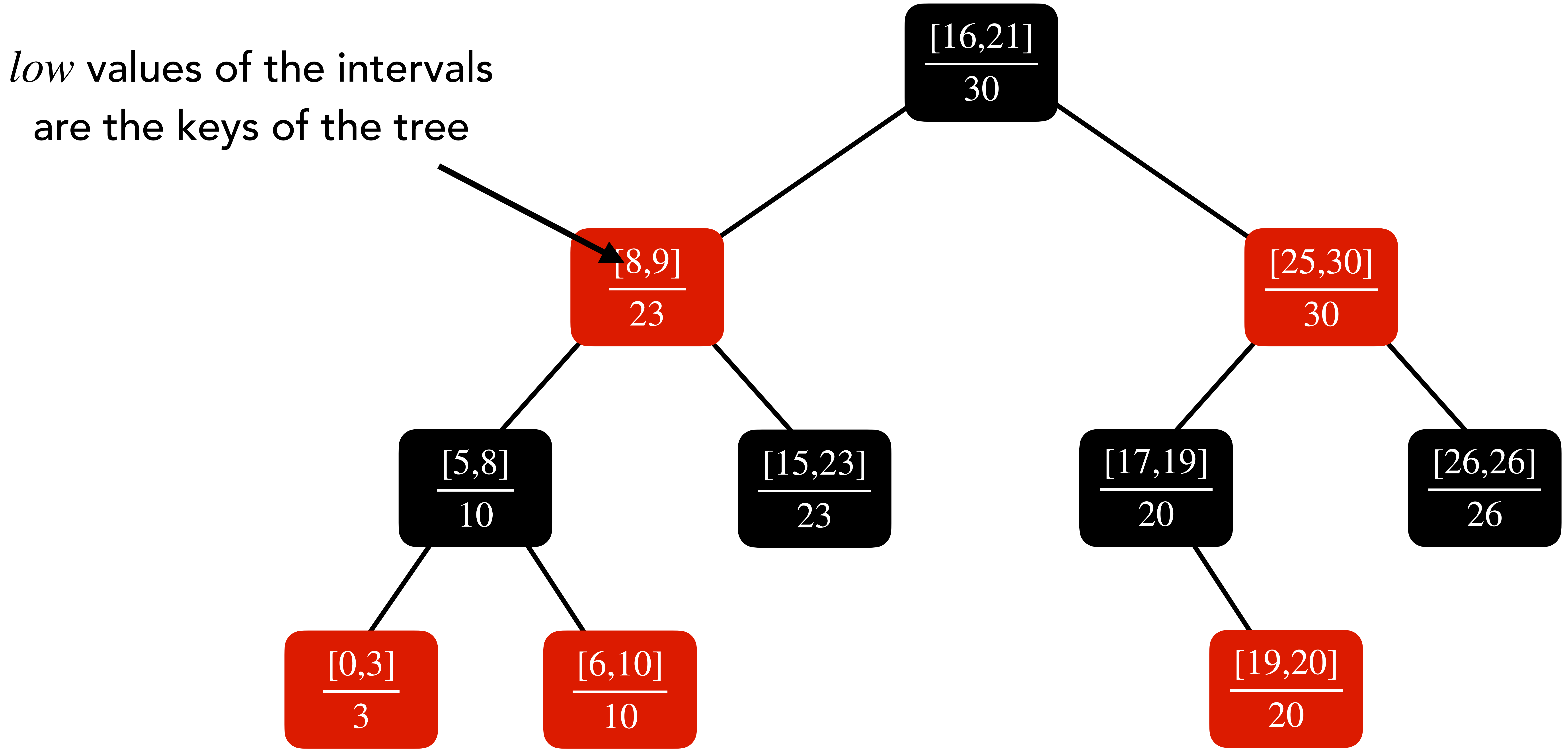


Example of an Interval Tree

low values of the intervals are the keys of the tree



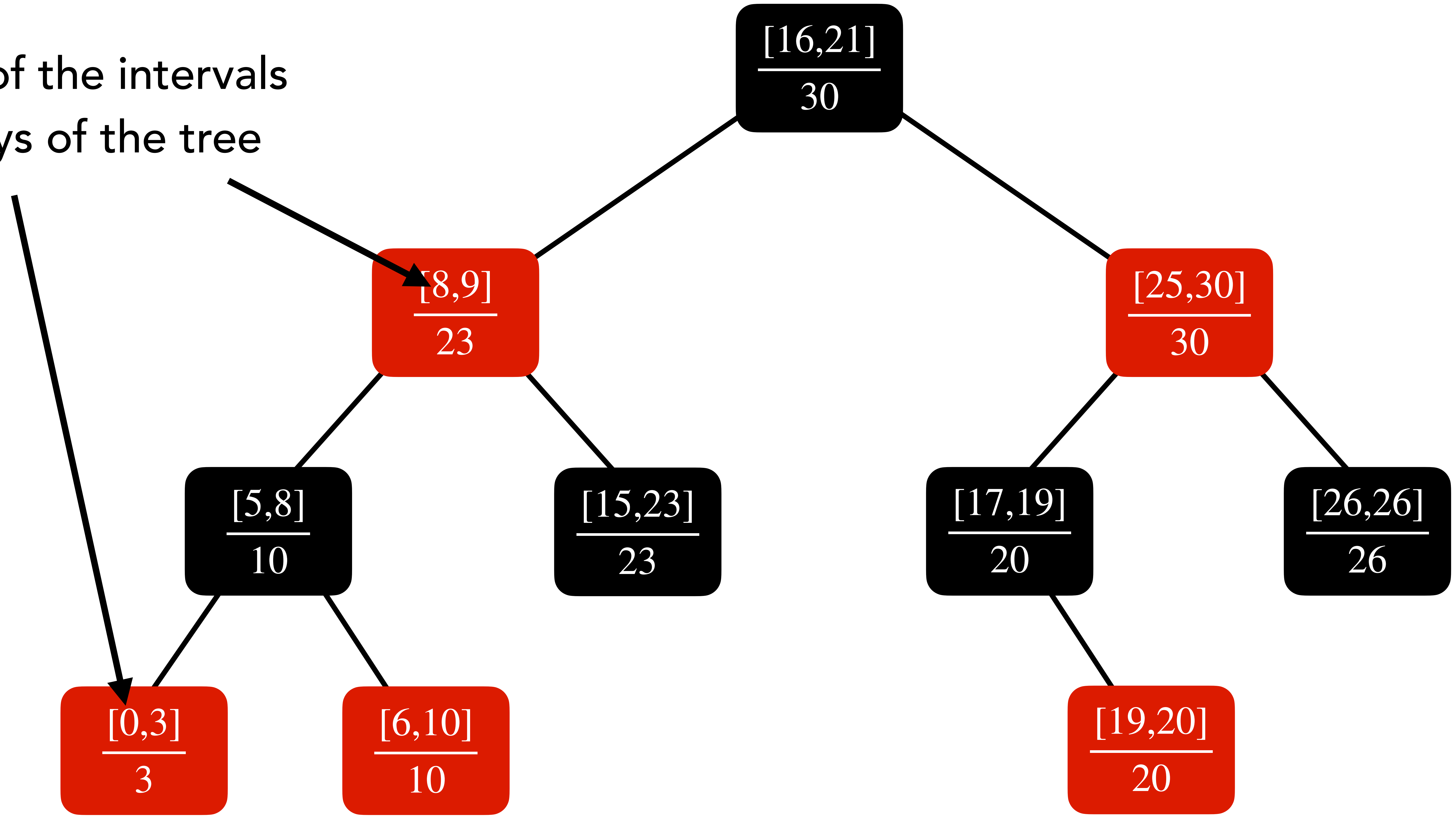
Example of an Interval Tree



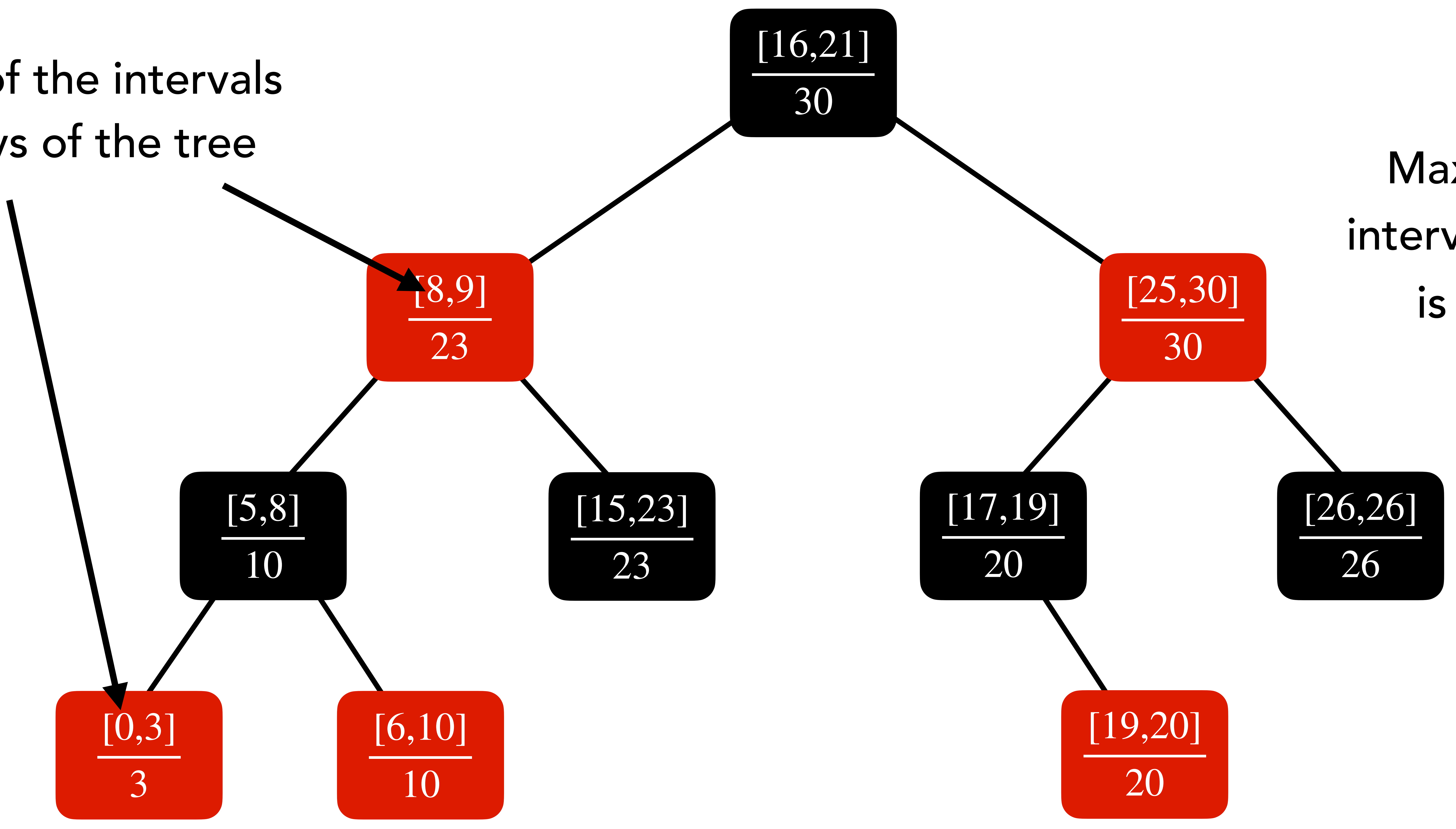
low values of the intervals are the keys of the tree

Example of an Interval Tree

low values of the intervals are the keys of the tree



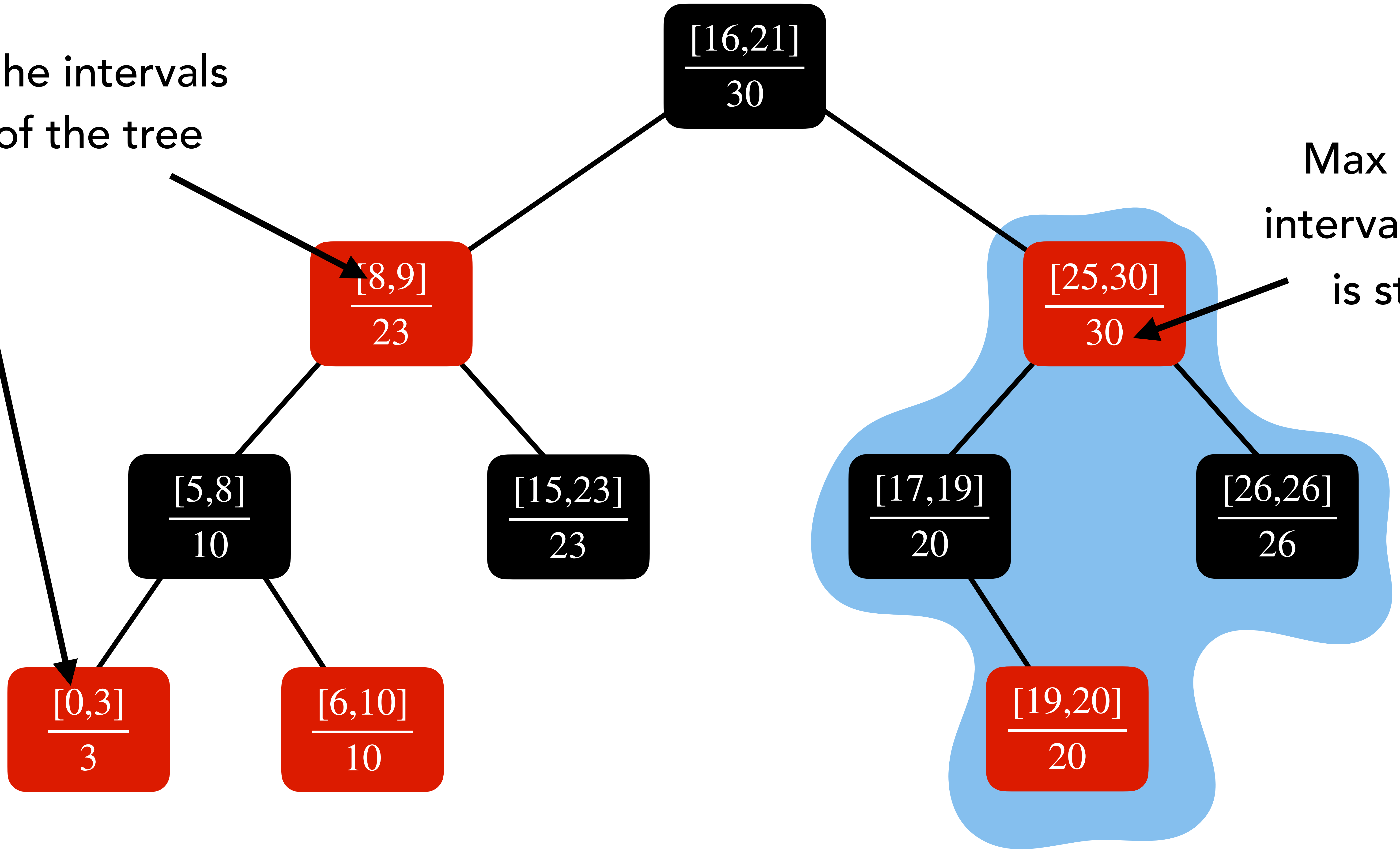
Example of an Interval Tree



low values of the intervals are the keys of the tree

Max high value of the intervals in the subtree(x) is stored as $x.max$

Example of an Interval Tree



low values of the intervals are the keys of the tree

Max high value of the intervals in the subtree(x) is stored as $x.max$

Interval Tree Definition

Interval Tree Definition

Defn: An interval tree is an R**B-tree**, where every node x contains an interval $x.int$, such that

Interval Tree Definition

Defn: An interval tree is an R**B-tree**, where every node x contains an interval $x.int$, such that

- The key of x is $x.int.low$.

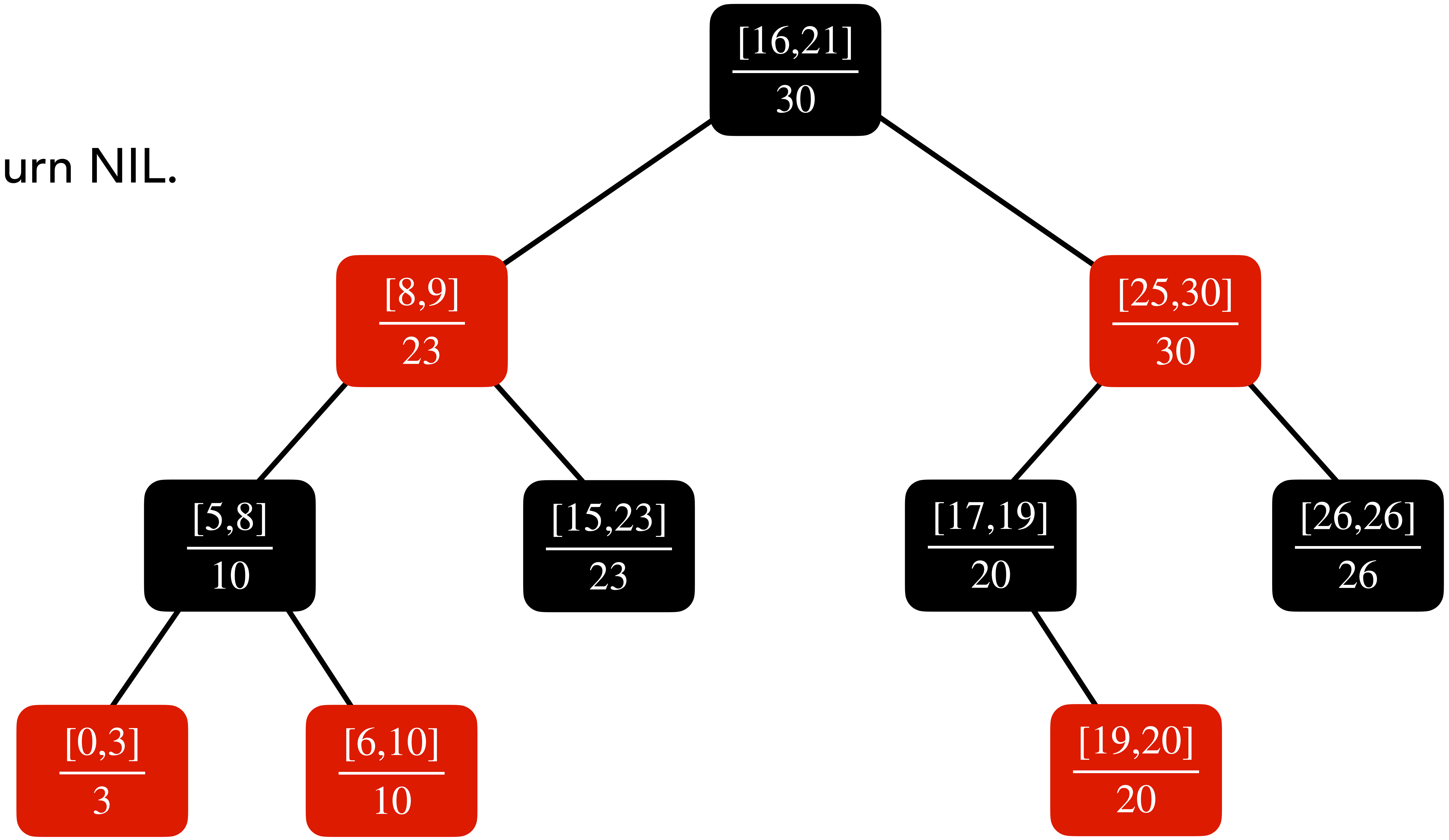
Interval Tree Definition

Defn: An interval tree is an R**B-tree**, where every node x contains an interval $x.int$, such that

- The key of x is $x.int.low$.
- Maximum of all the high values of the intervals in the **subtree**(x) is stored as $x.max$.

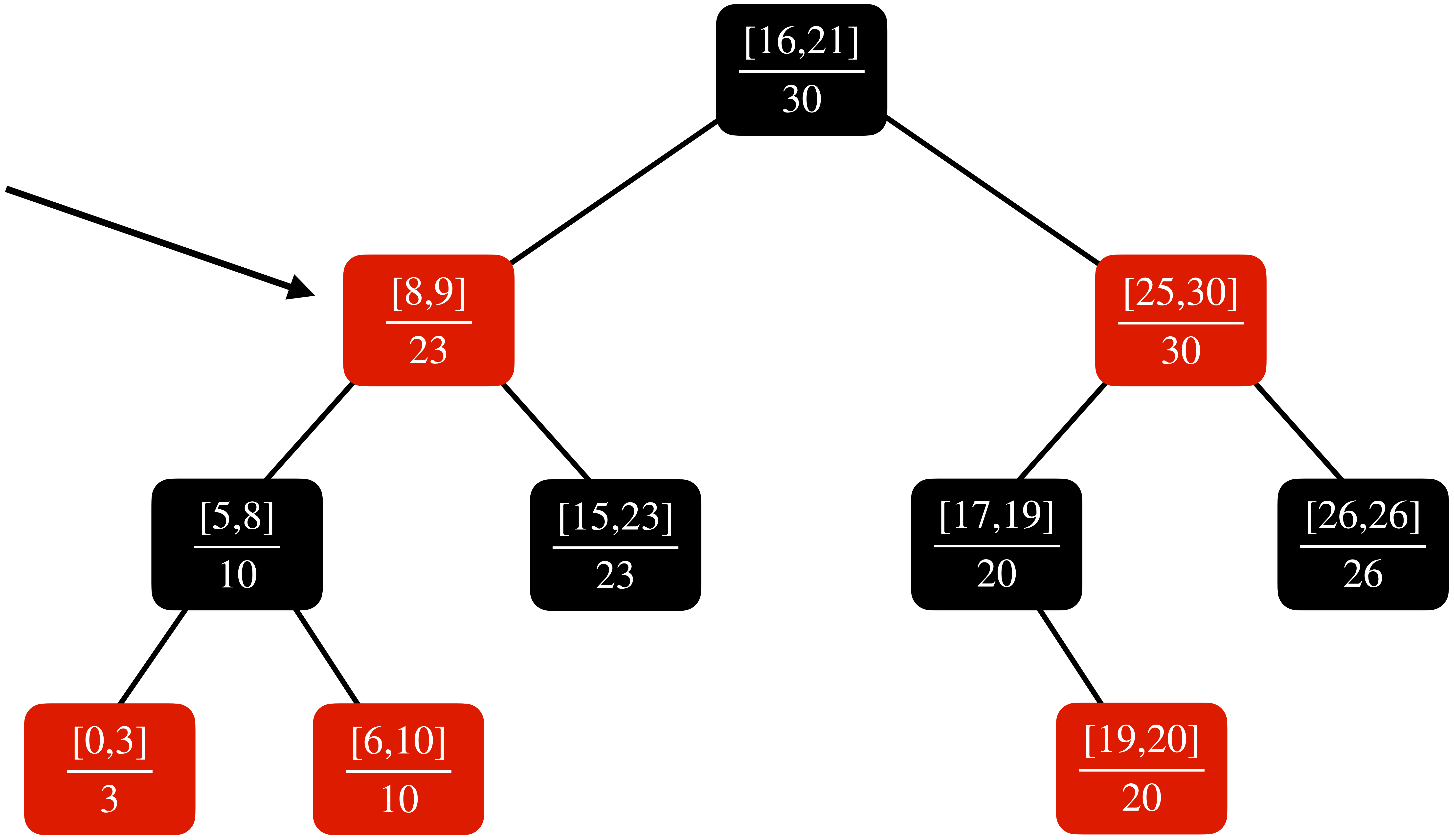
Idea Behind Interval-Search

Interval-Search$(T, [11{,}14])$ should return NIL.



Idea Behind Interval-Search

Interval-Search$(T, [6,10])$ can return



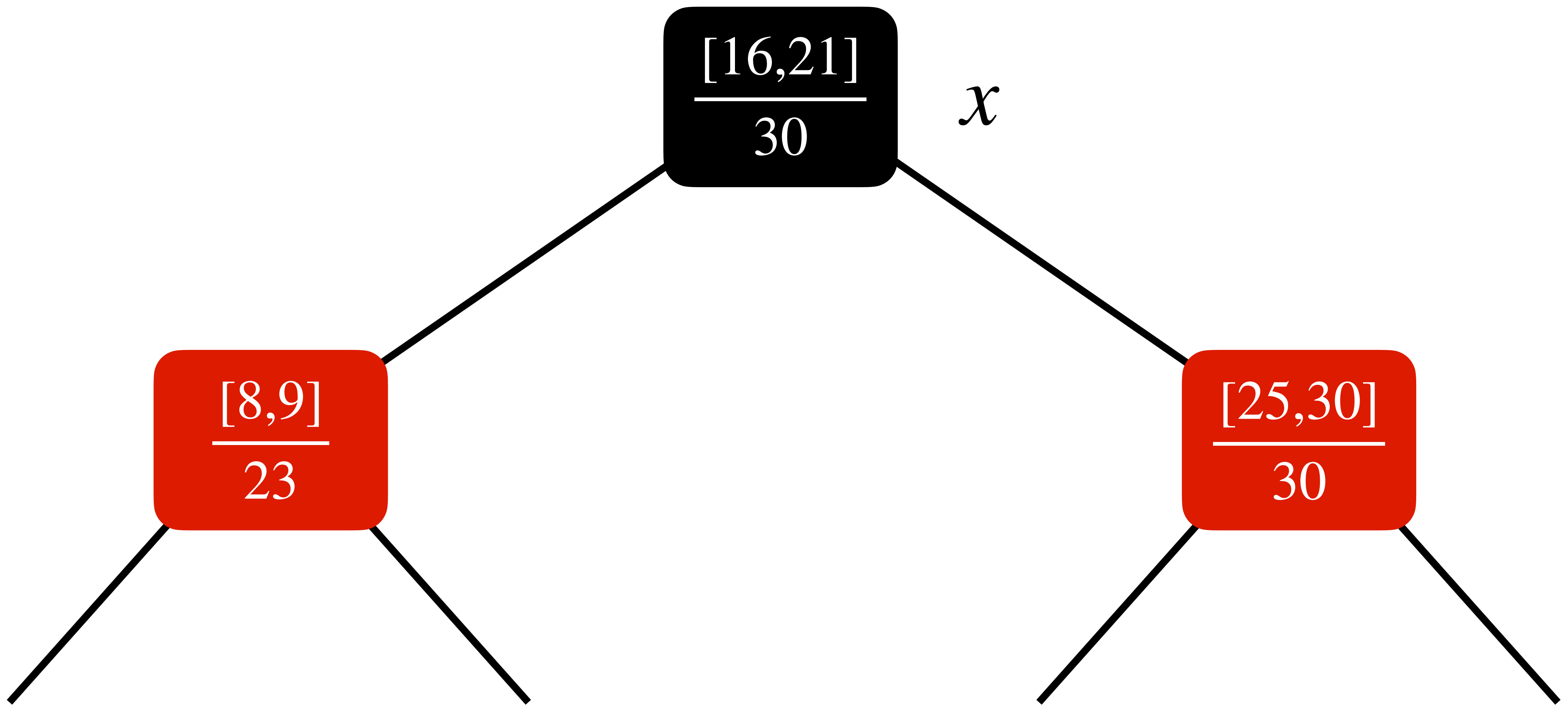
Idea Behind Interval-Search

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [25,40]$.

Idea Behind Interval-Search

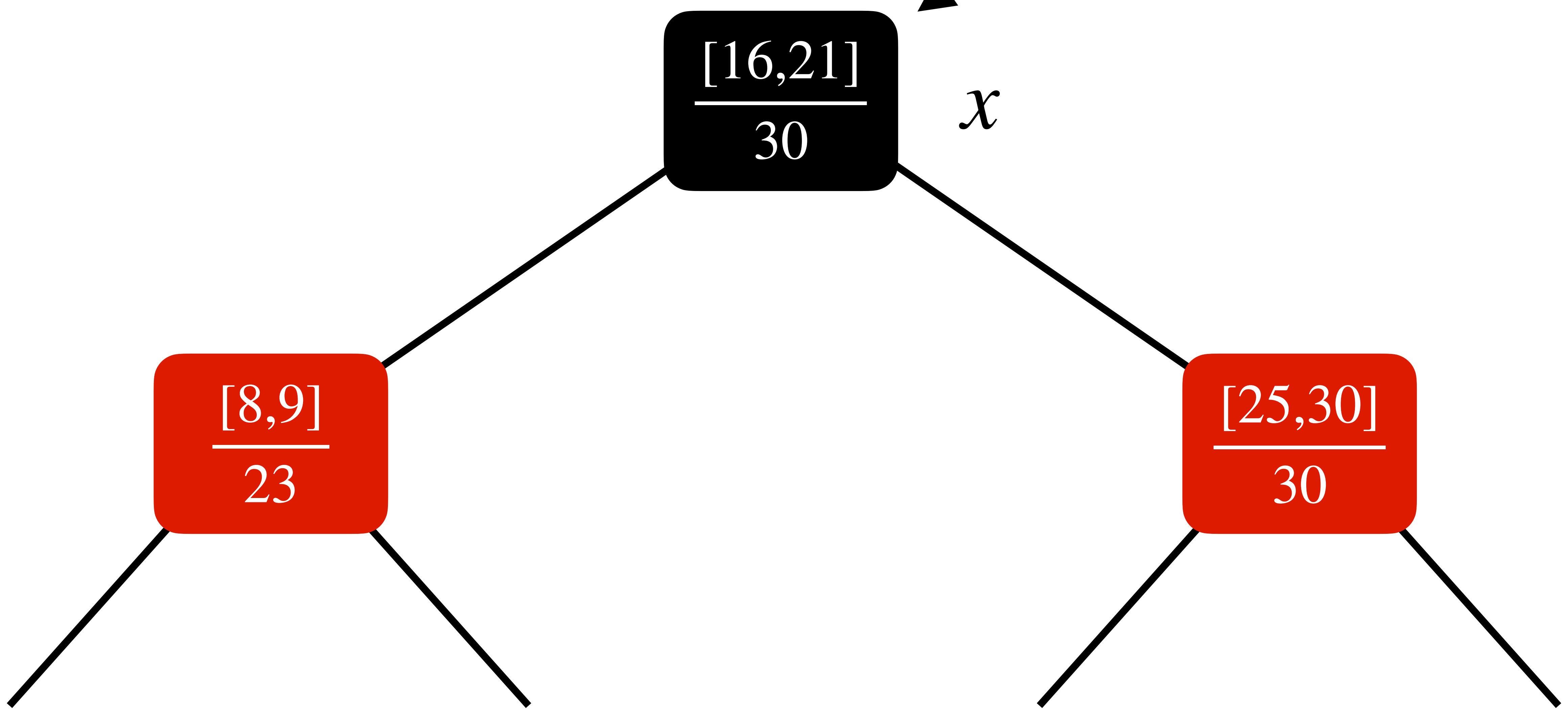
Find the node with interval overlapping with $i = [25,40]$.



Idea Behind Interval-Search

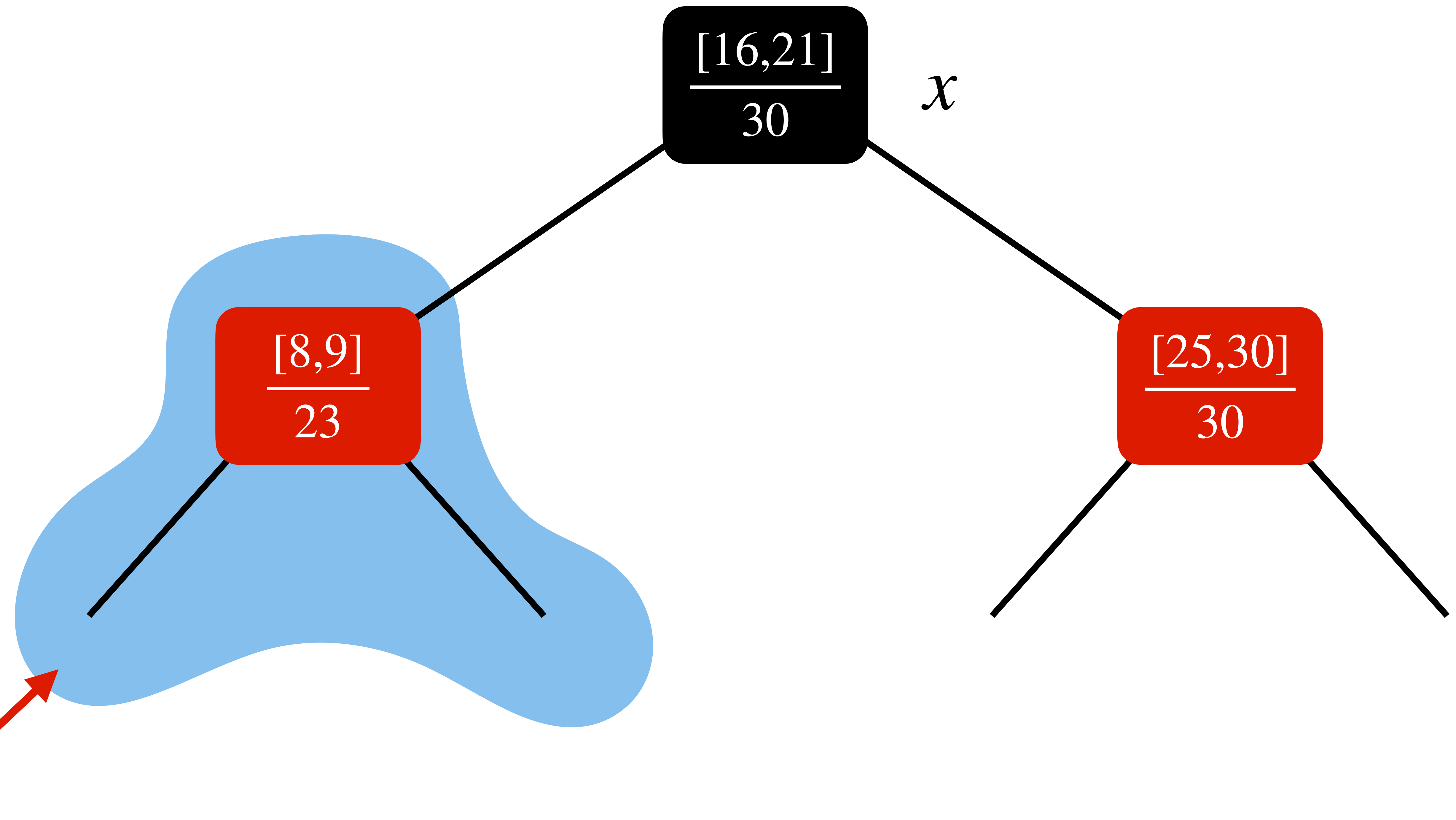
Find the node with interval overlapping with $i = [25,40]$.

[16,21] clearly doesn't overlap with [25,40].



Idea Behind Interval-Search

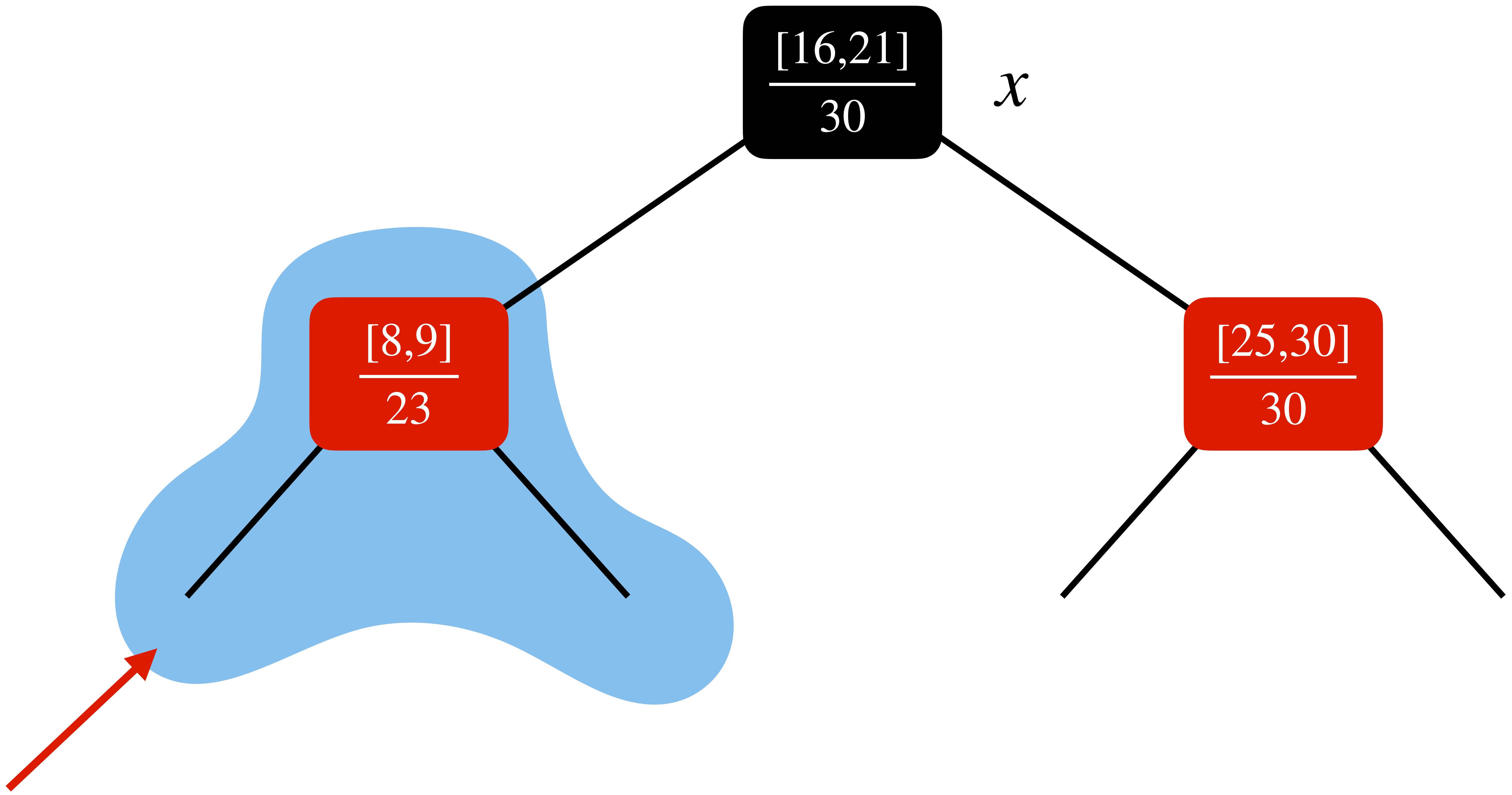
Find the node with interval overlapping with $i = [25,40]$.



Can [25,40] overlap with an interval in the left subtree?

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [25,40]$.

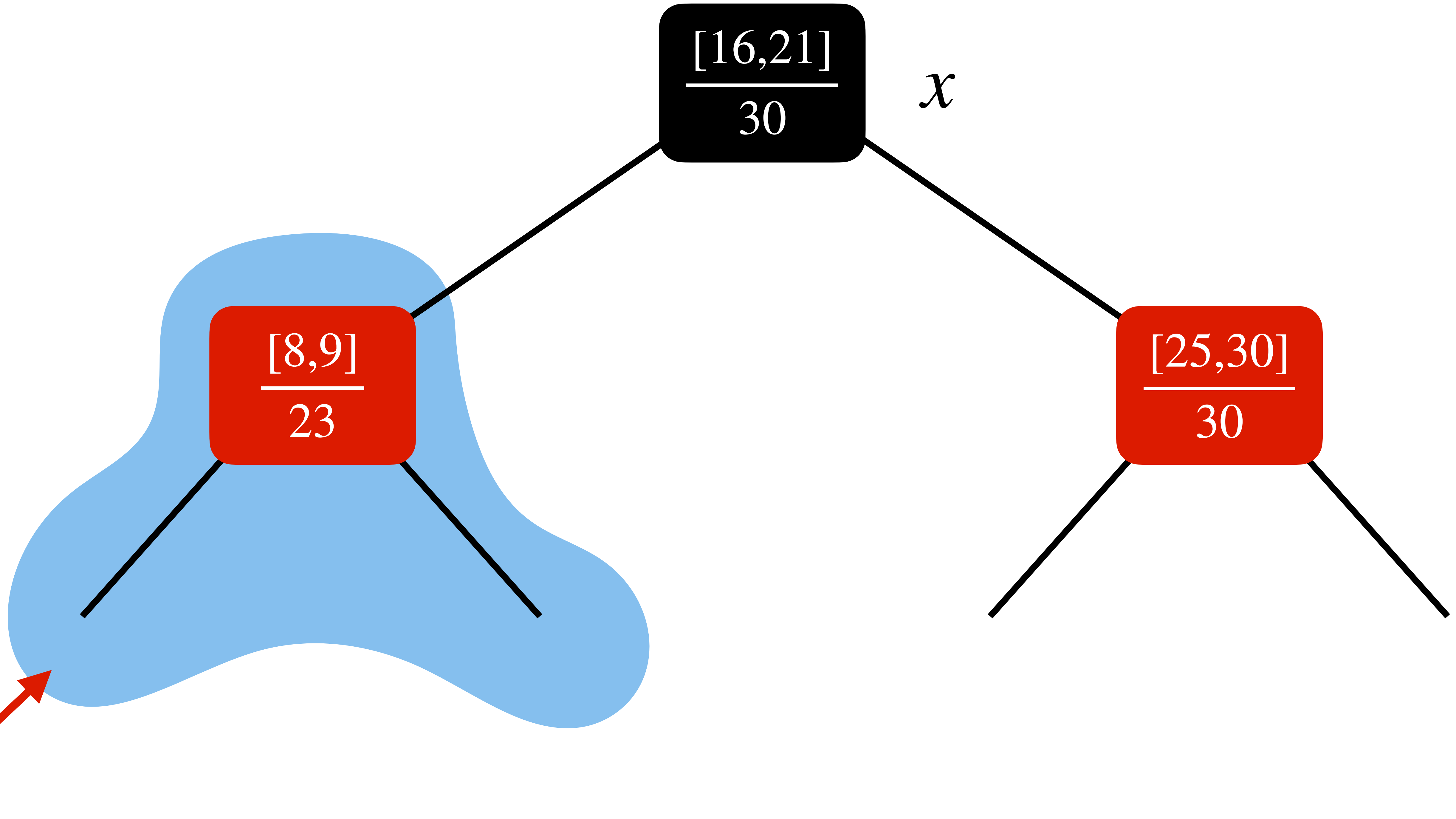


Can $[25,40]$ overlap with an interval in the left subtree?

No, because maximum high is 23.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [25,40]$.

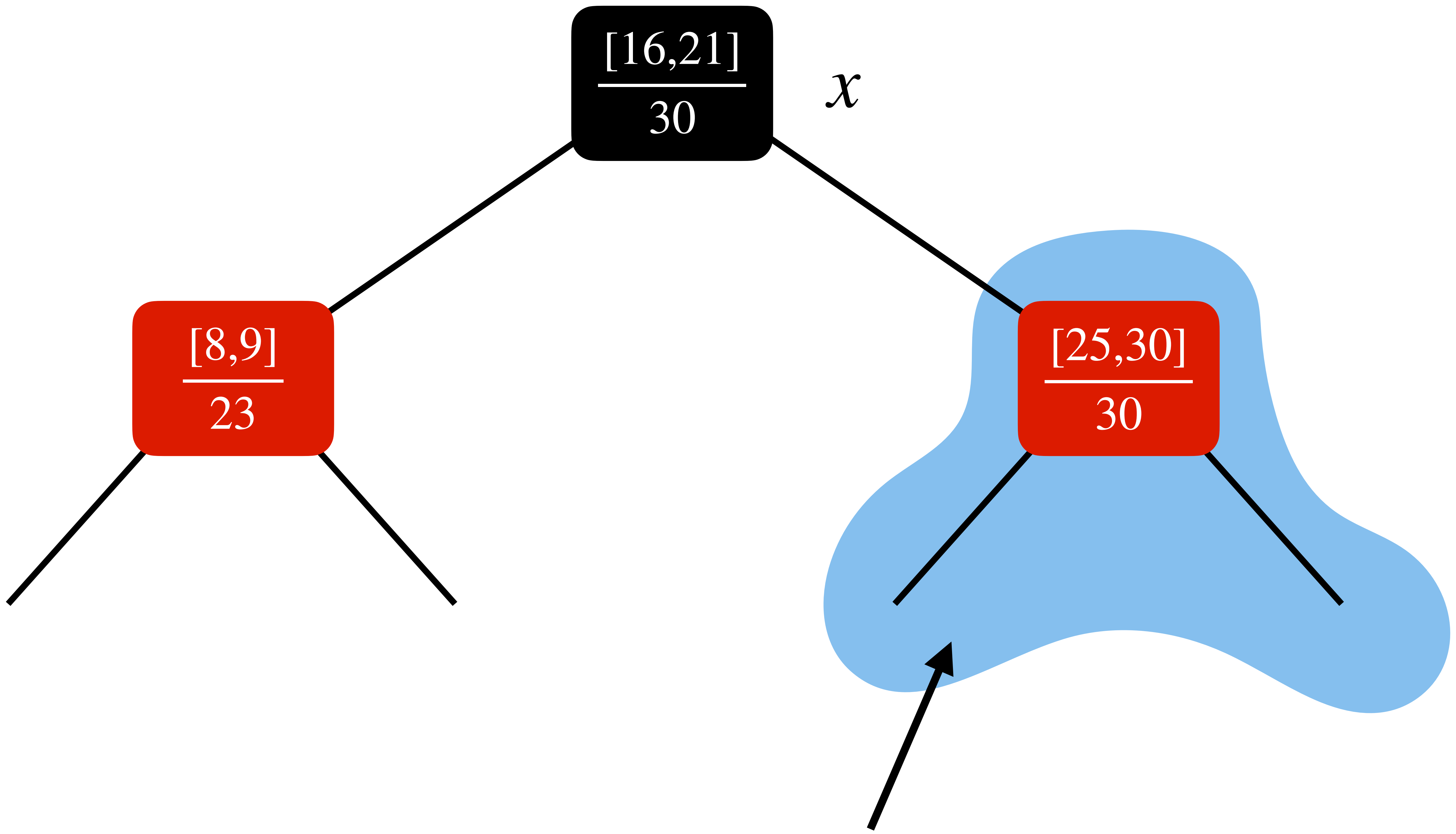


Can $[25,40]$ overlap with an interval in the left subtree?

No, because maximum high is 23. Hence, every interval in this subtree will be to the left of $[25,40]$.

Idea Behind Interval-Search

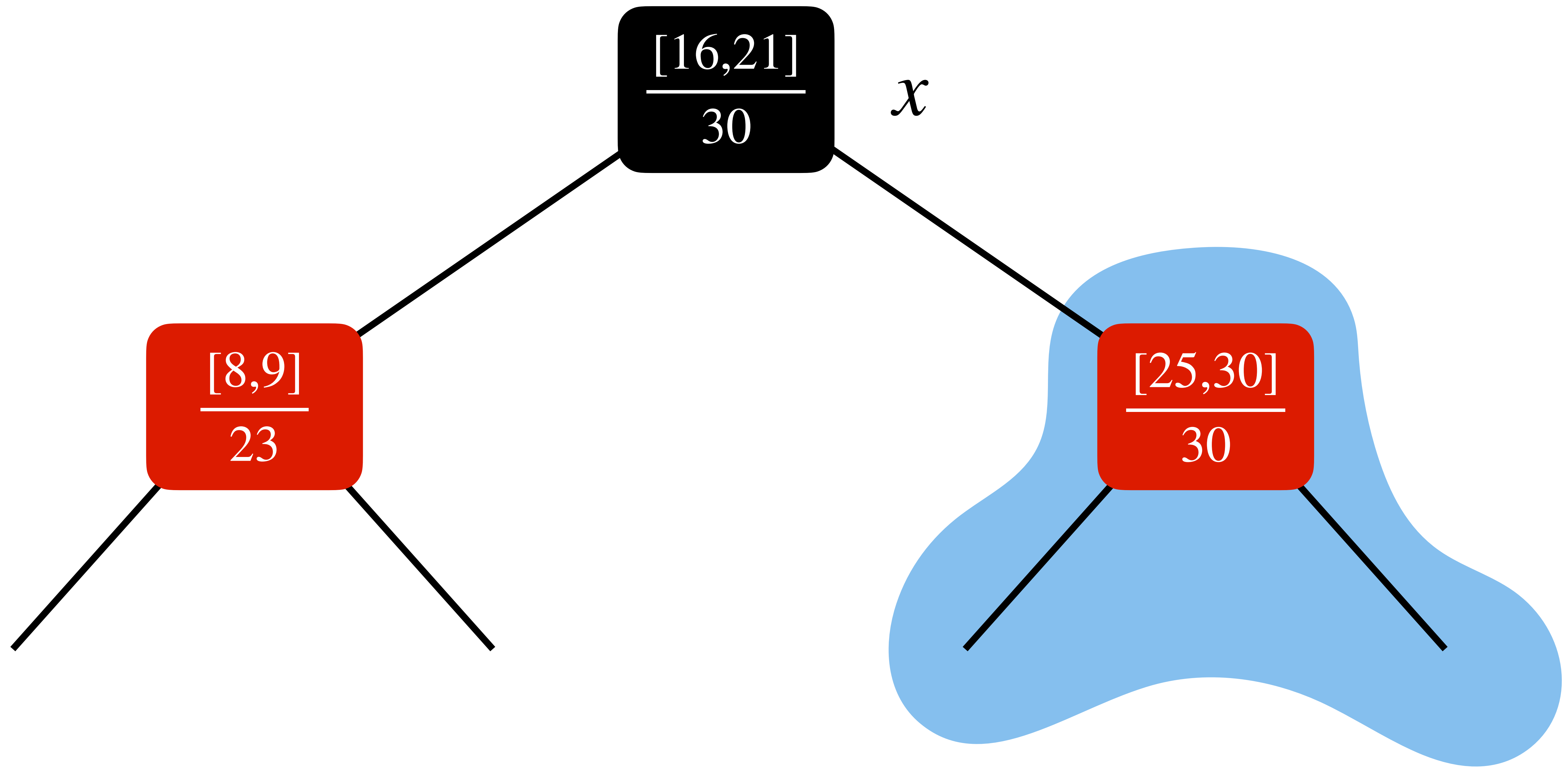
Find the node with interval overlapping with $i = [25,40]$.



Any interval overlapping with $[25,40]$, if present must be in the right subtree.

Idea Behind Interval-Search

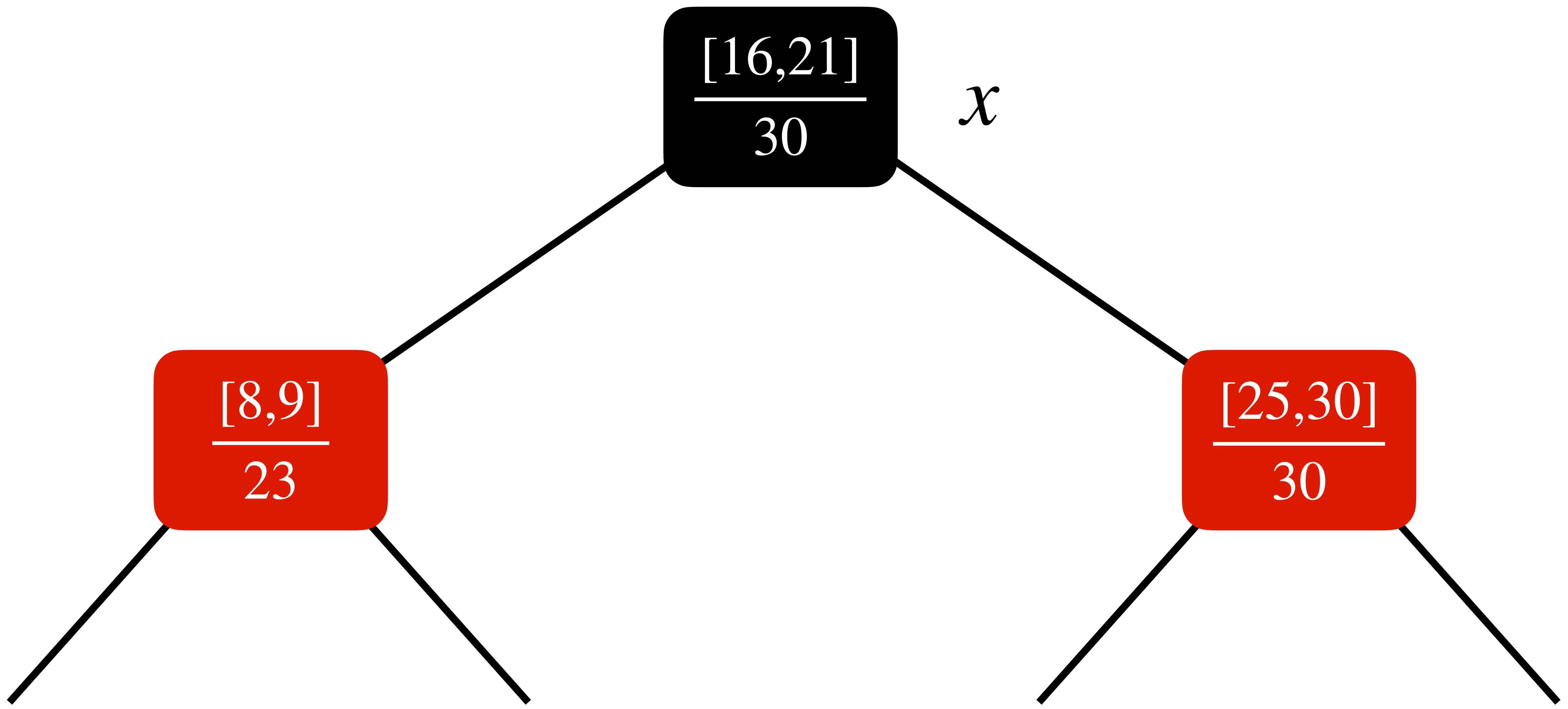
Find the node with interval overlapping with $i = [25,40]$.



When $i.low > x.left.max$, go right

Idea Behind Interval-Search

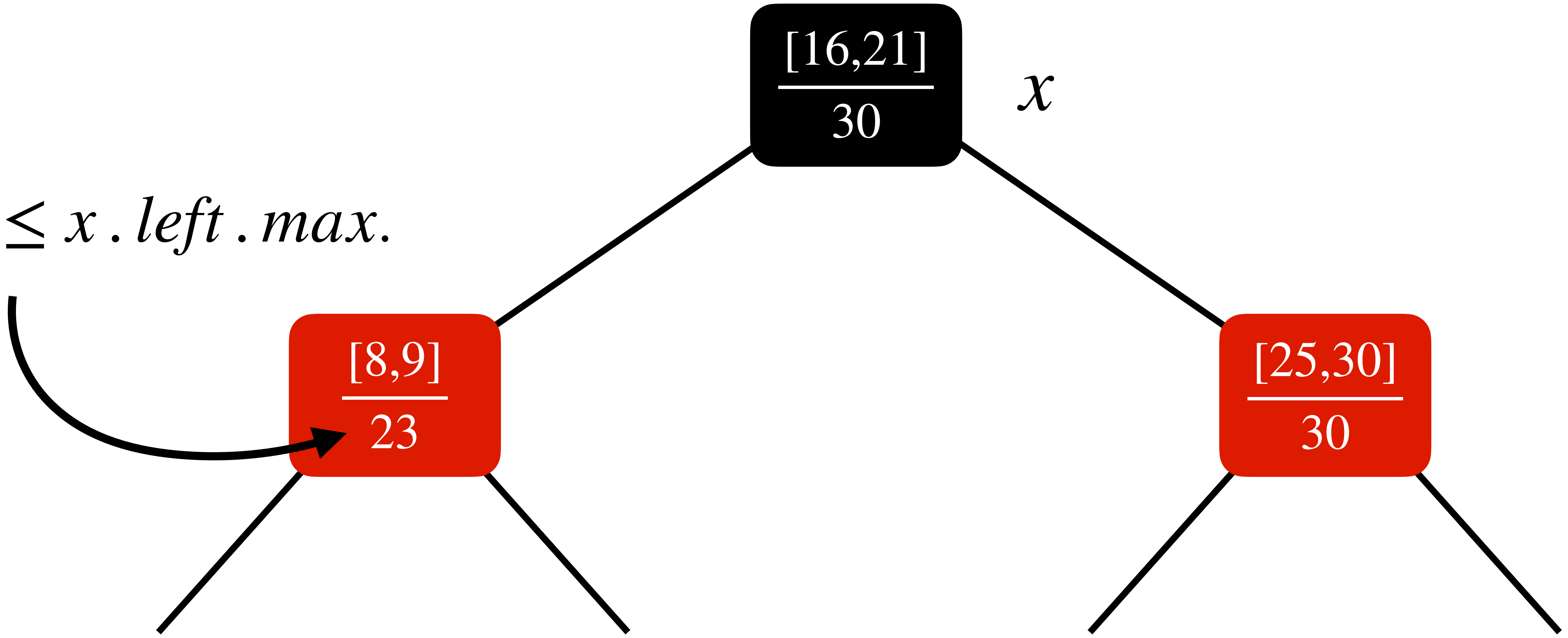
Find the node with interval overlapping with $i = [10,14]$.



Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

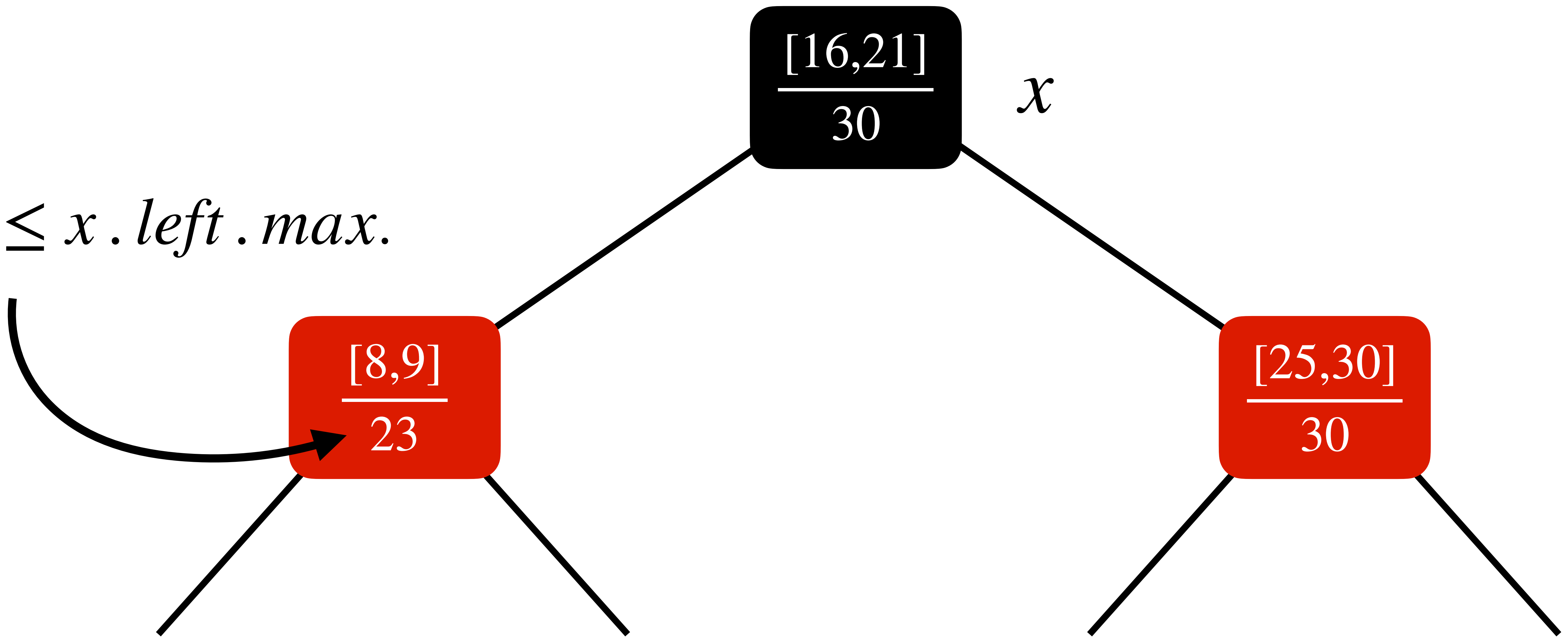
Notice that $i.low \le x.left.max$.



Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

Notice that $i.low \leq x.left.max$.

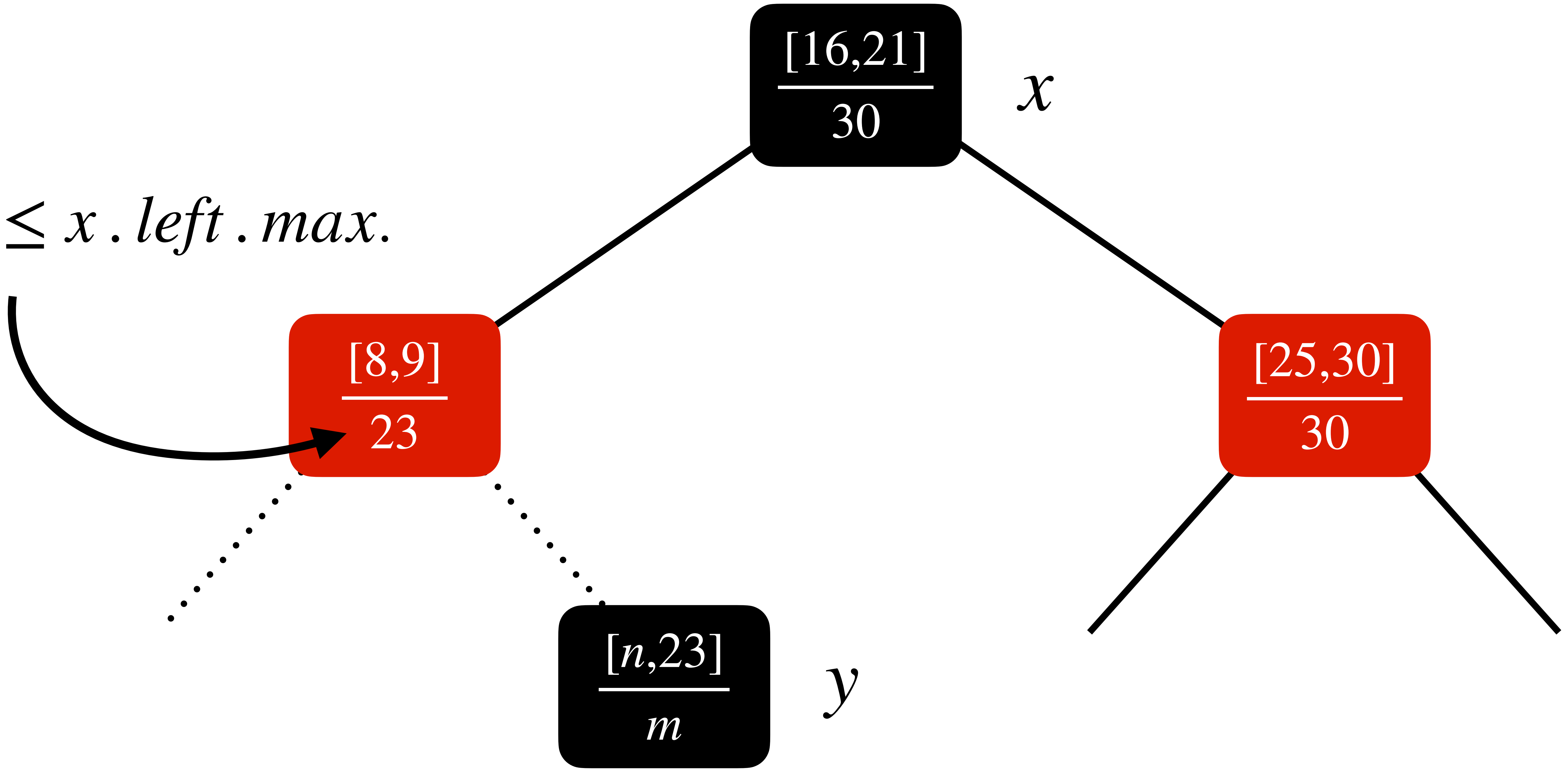


We will prove now that when $i.low \leq x.left.max$, it is safe to go left.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

Notice that $i.low \leq x.left.max$.

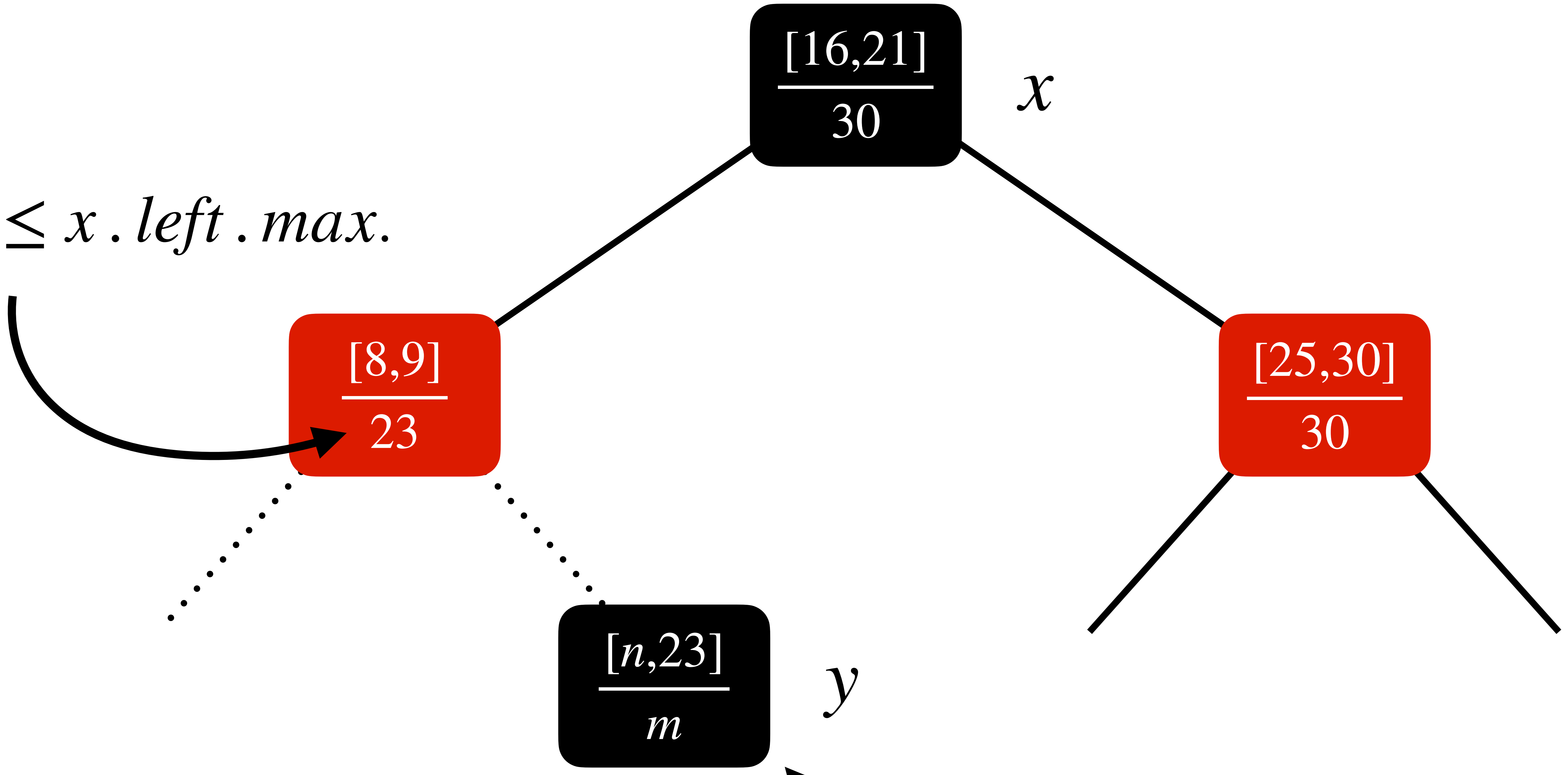


There must exist a node like this in the left subtree

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

Notice that $i.low \leq x.left.max$.

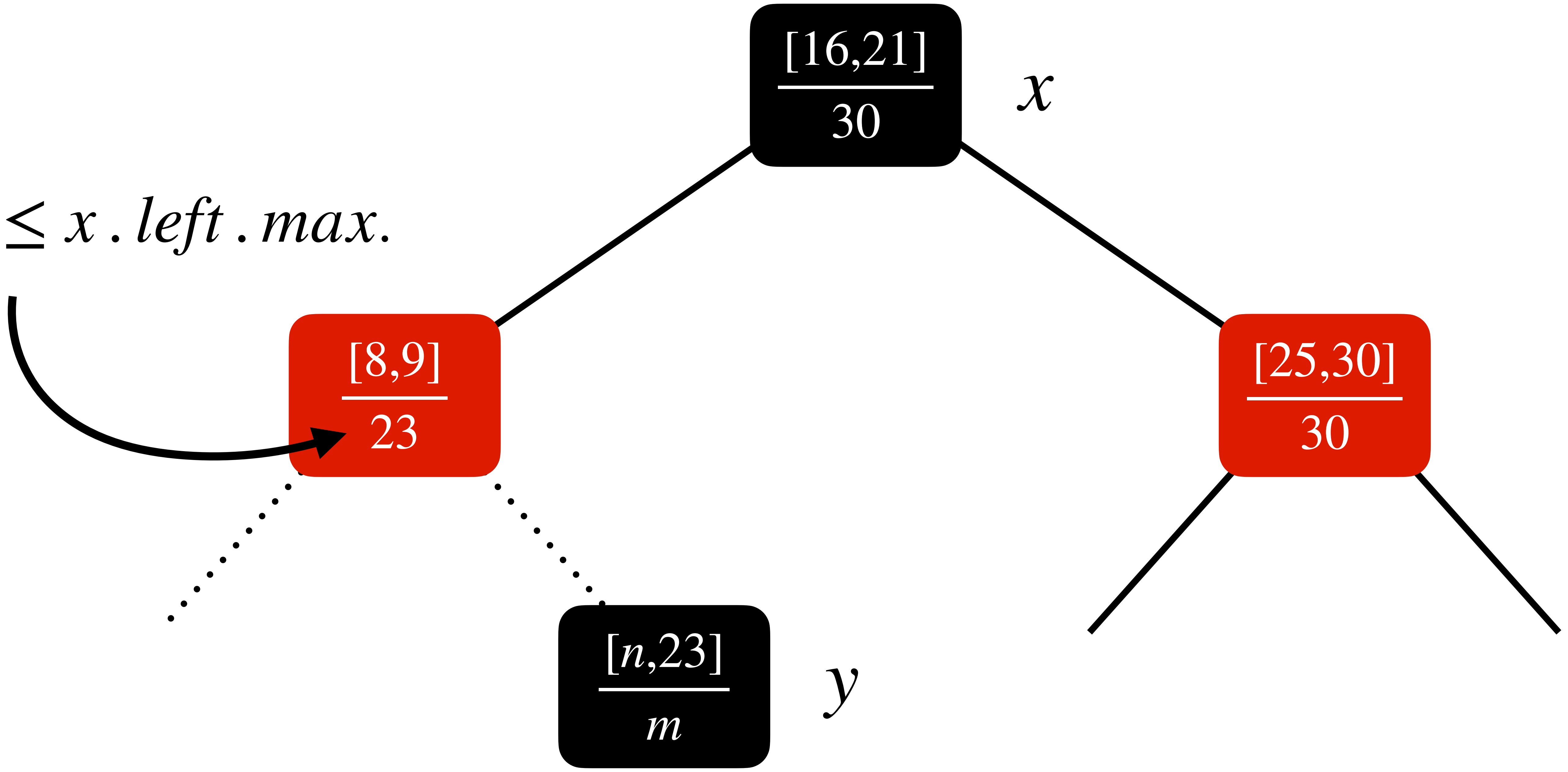


There must exist a node like this in the left subtree

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

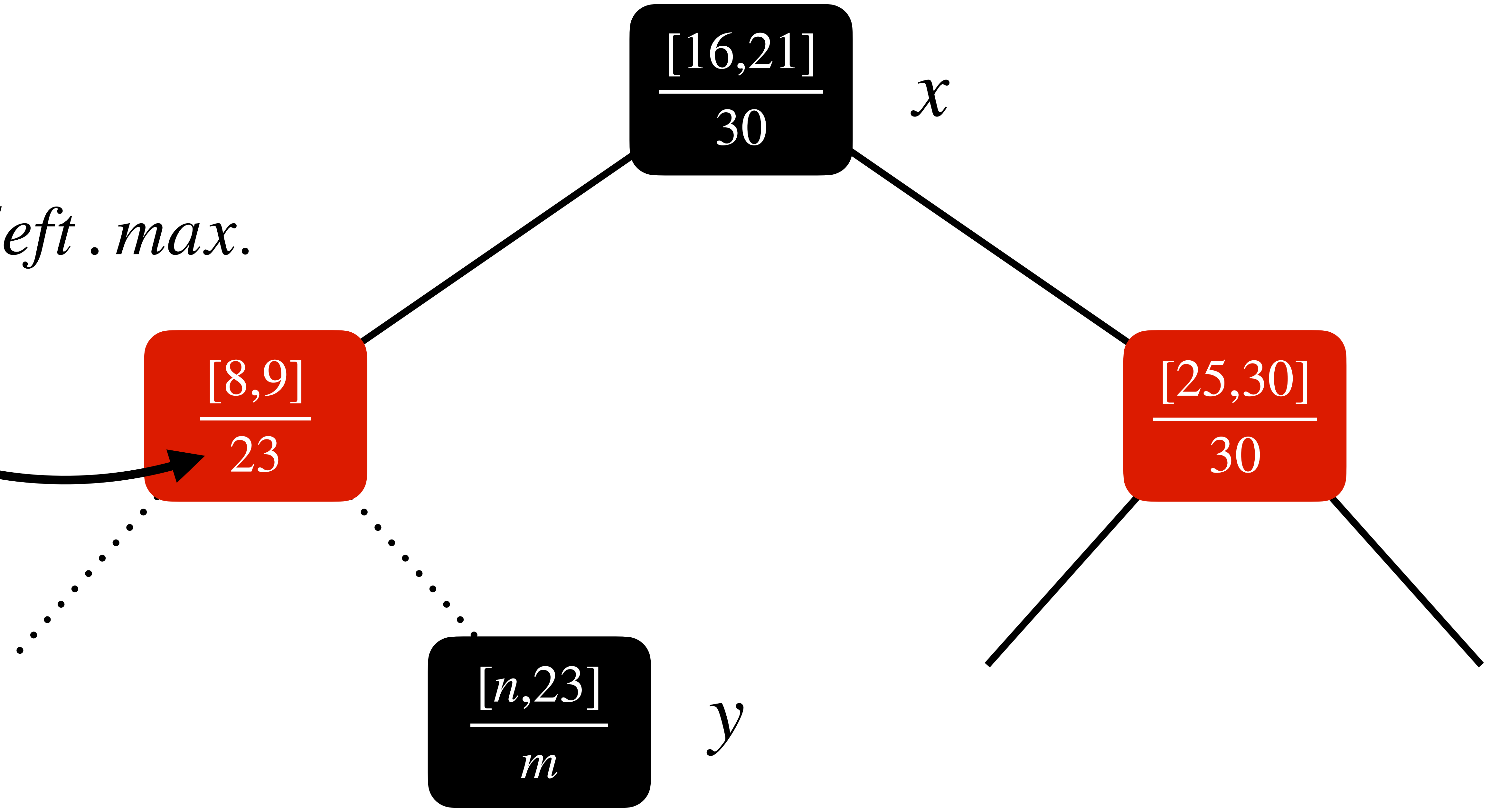
Notice that $i.low \leq x.left.max$.



Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

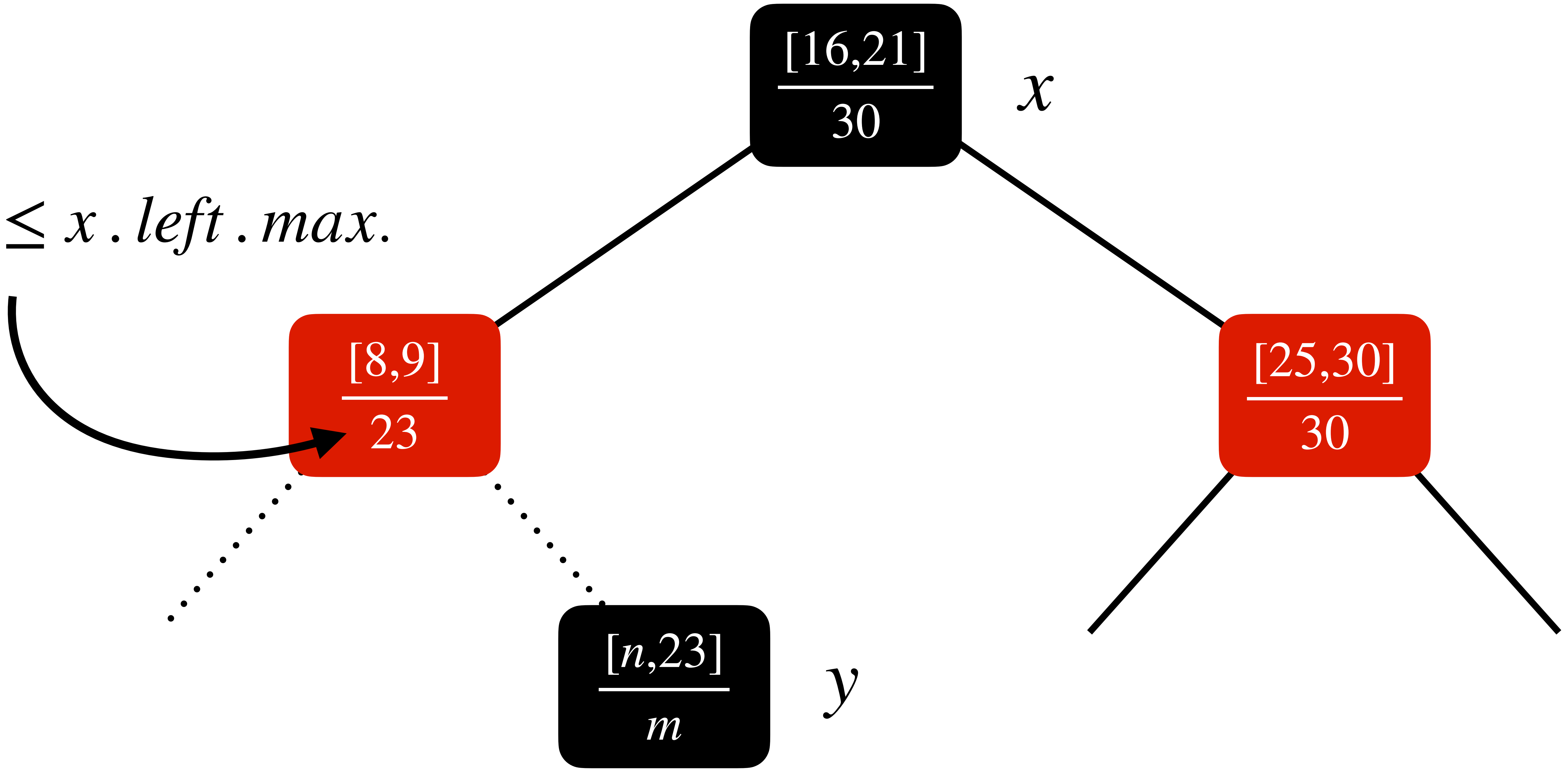
Notice that $i.low \leq x.left.max$.



If $n \leq 14$, y in the left subtree will overlap with $[10,14]$.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.



Notice that $i.low \leq x.left.max$.

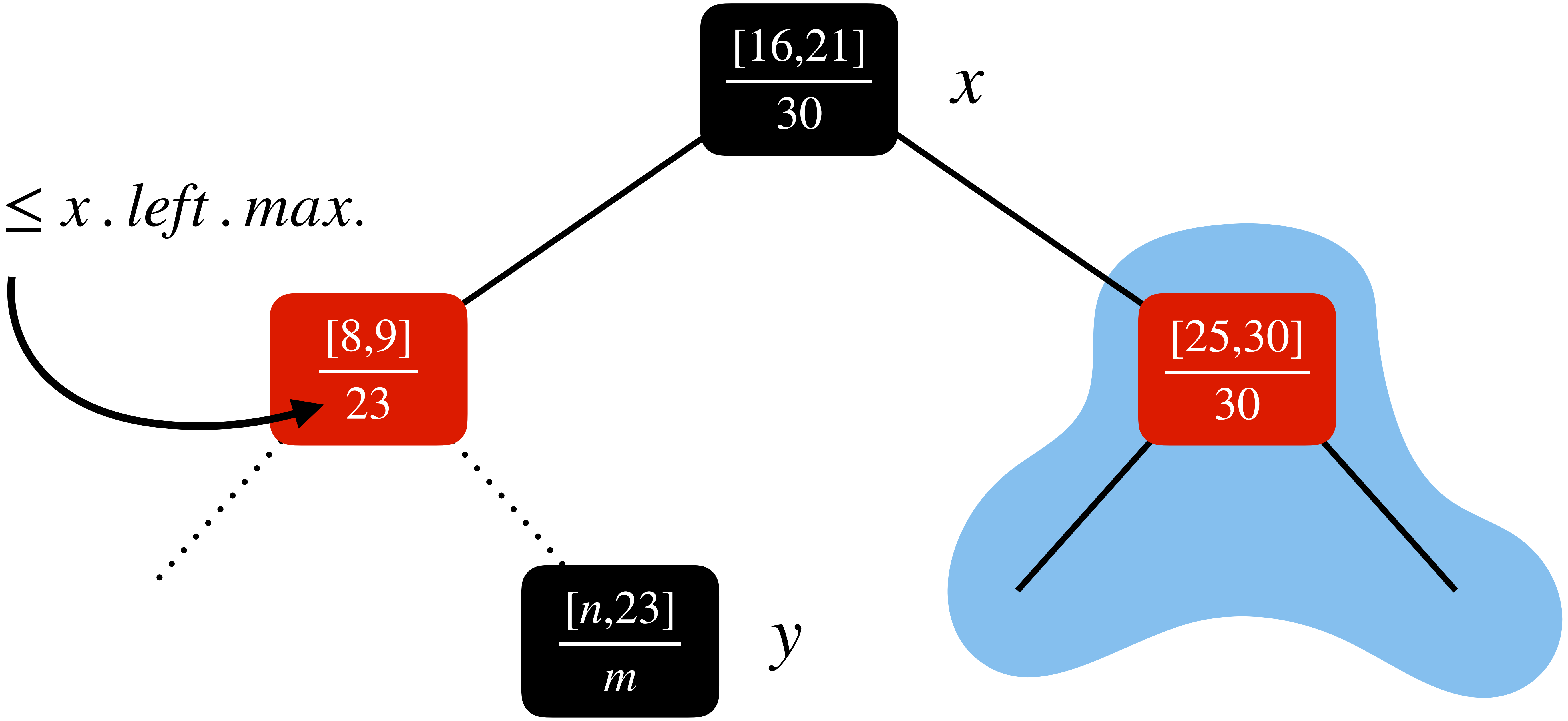
If $n \leq 14$, y in the left subtree will overlap with $[10,14]$.

If $n > 14$, y will not overlap with $[10,14]$.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

Notice that $i\,.\,low \leq x\,.\,left\,.\,max$.

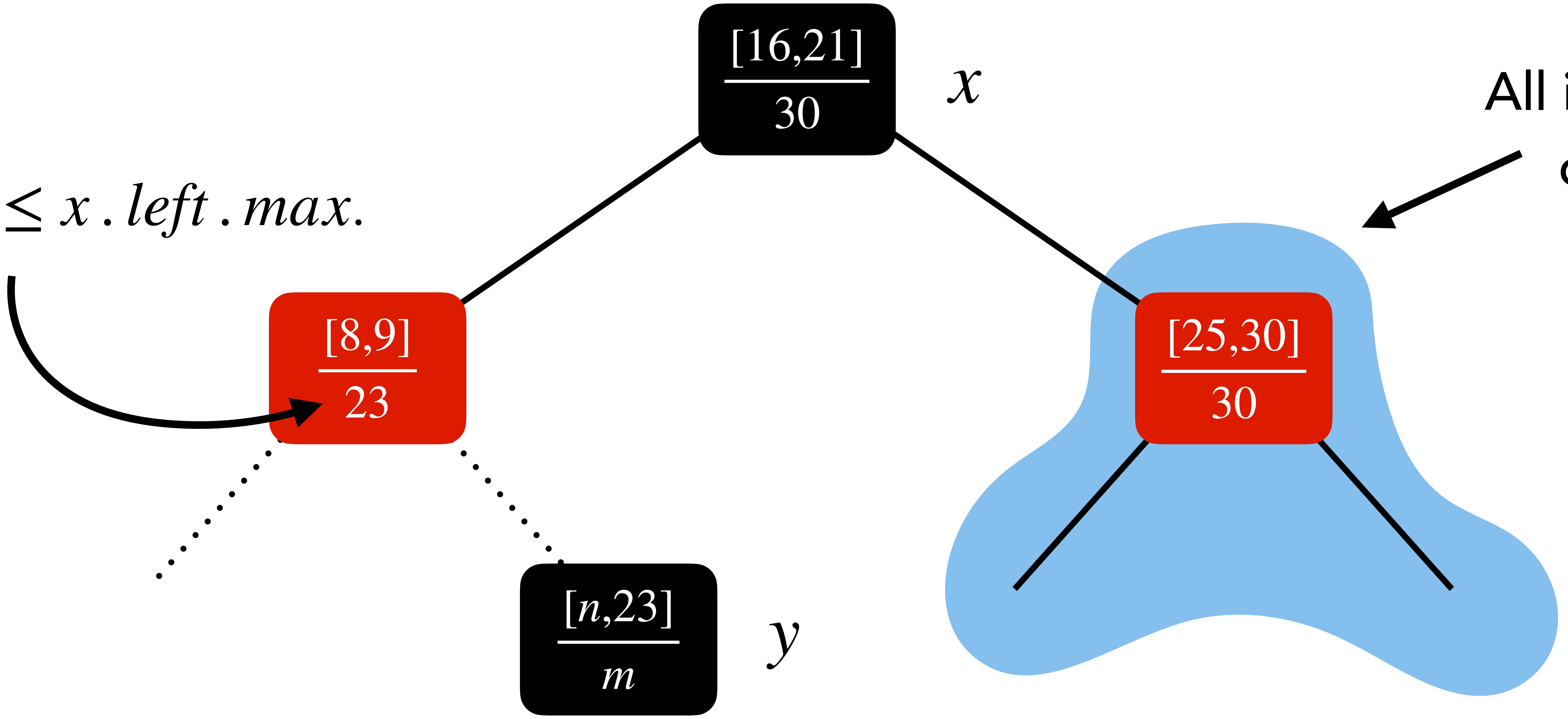


If $n \leq 14$, y in the left subtree will overlap with $[10,14]$.

If $n > 14$, y will not overlap with $[10,14]$.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.



Notice that $i.low \leq x.left.max$.

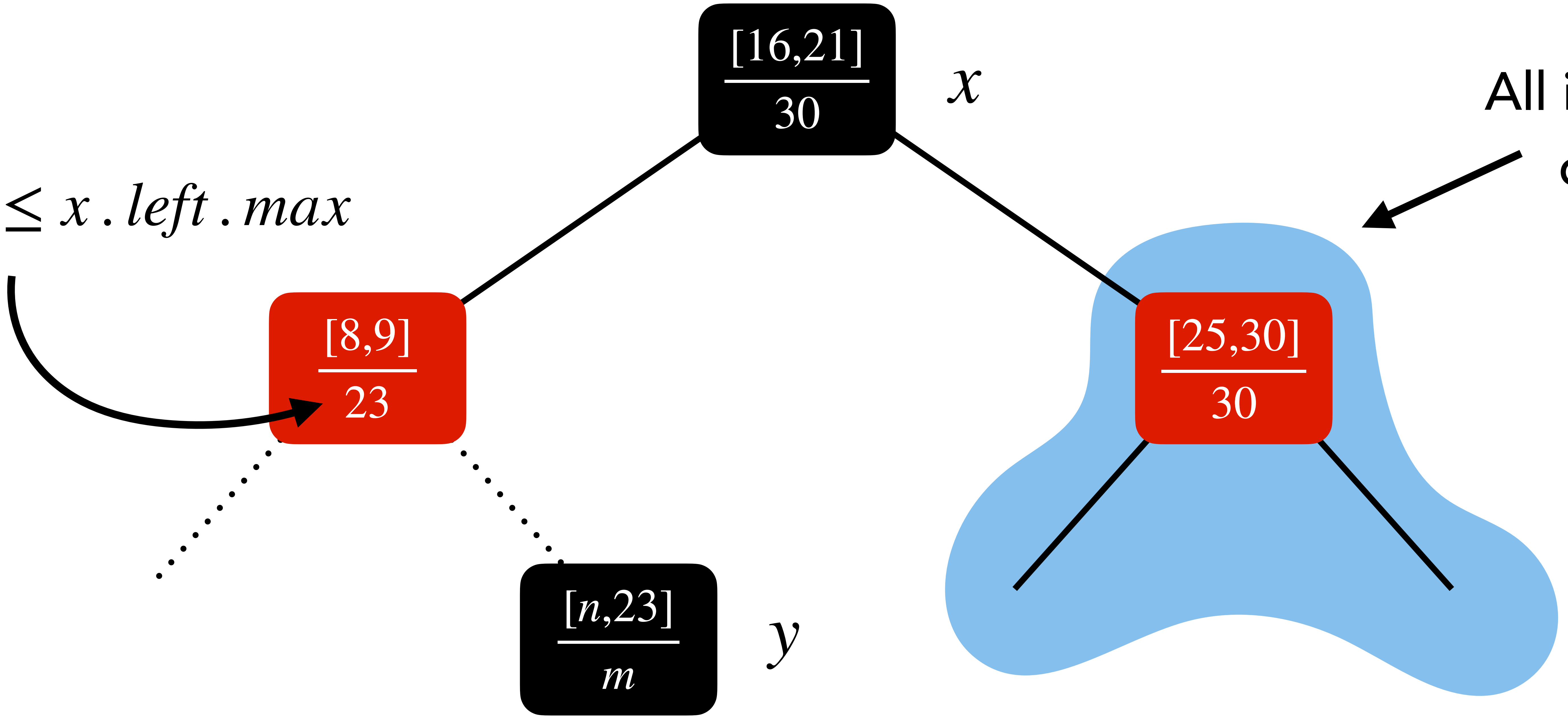
All intervals' low will be > 14 due to BST properties.

If $n \leq 14$, y in the left subtree will overlap with $[10,14]$.

If $n > 14$, y will not overlap with $[10,14]$.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.



Notice that $i\,.\,low \leq x\,.\,left\,.\,max$

All intervals' low will be > 14 due to BST properties.

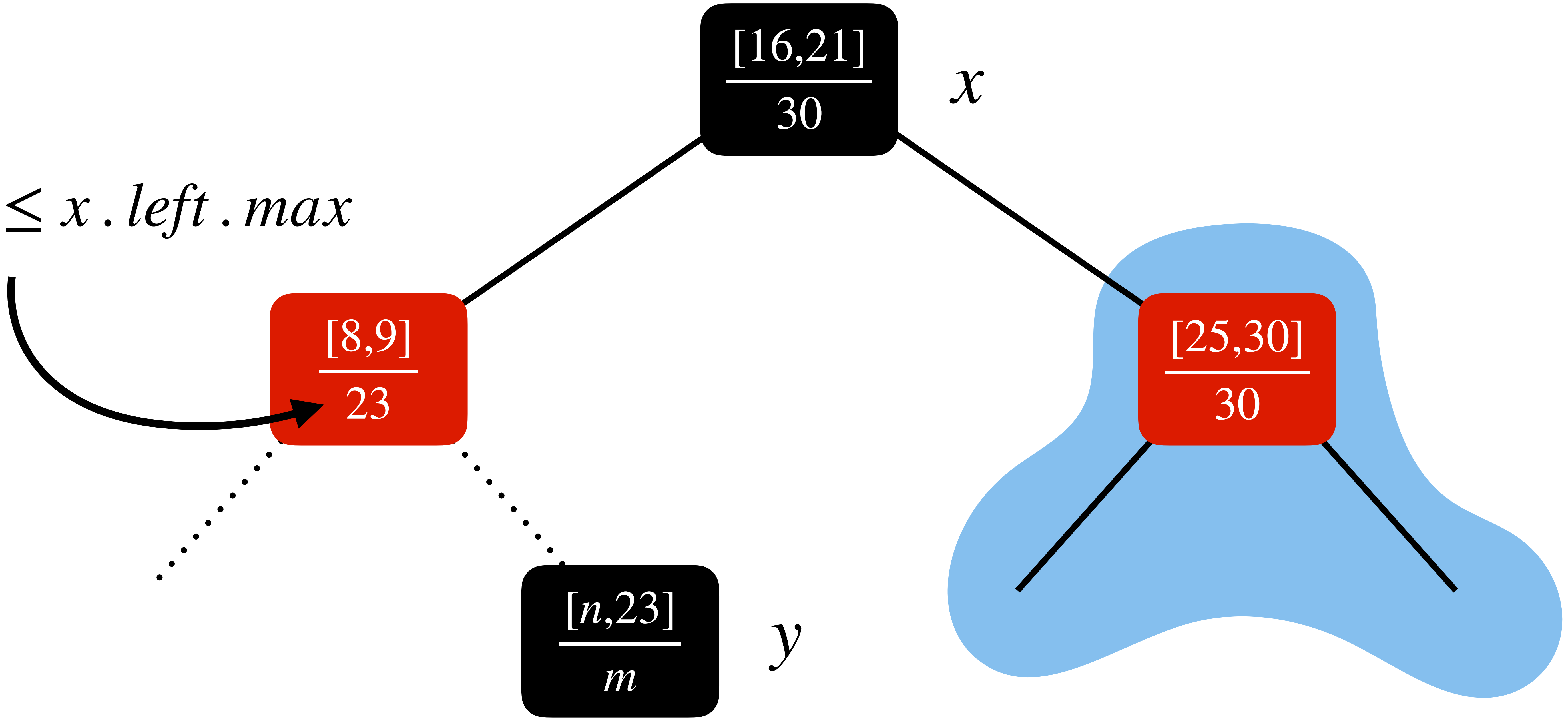
If $n \leq 14$, y in the left subtree will overlap with $[10,14]$.

If $n > 14$, no node in the right subtree will overlap with $[10,14]$.

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10{,}14]$.

Notice that $i\,.\,low \leq x\,.\,left\,.\,max$

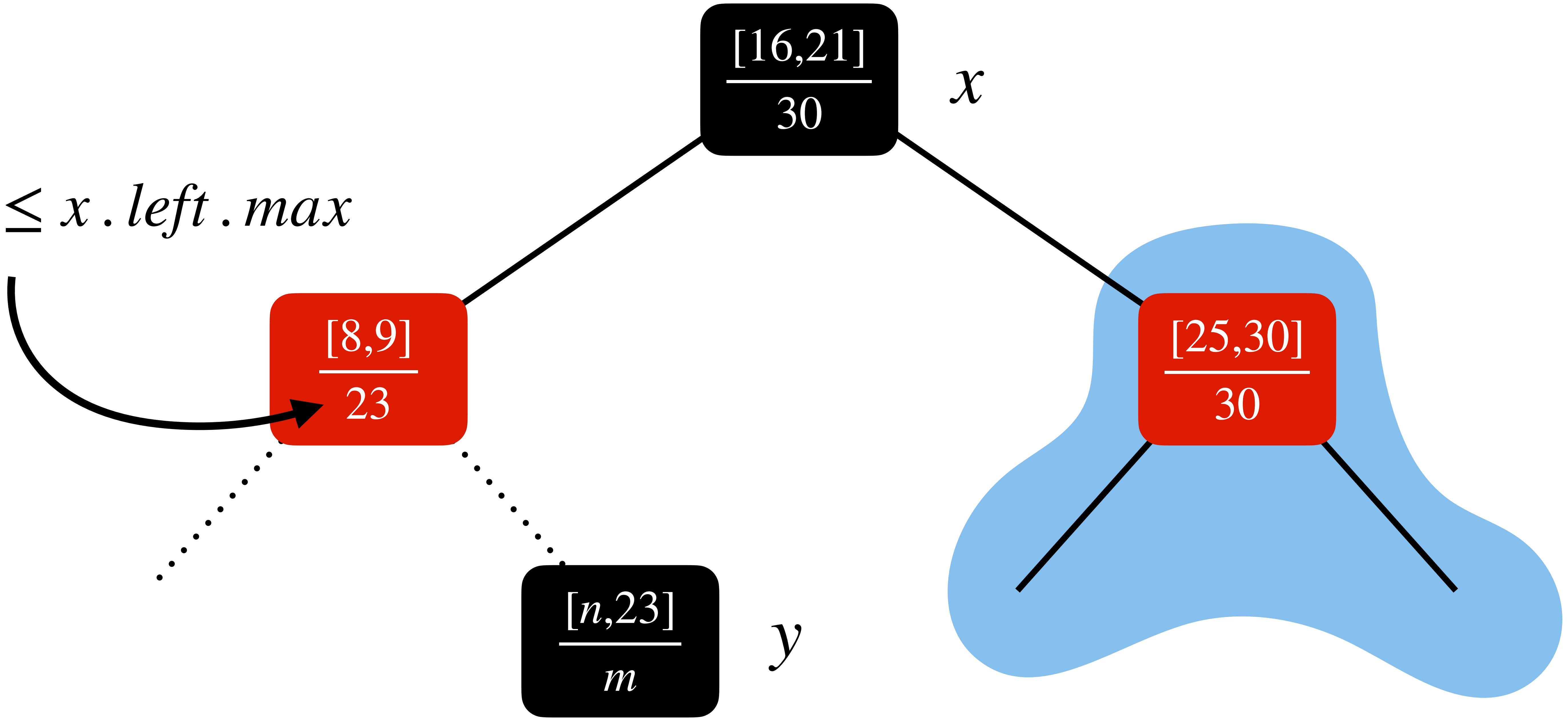


When $i\,.\,low \leq x\,.\,left\,.\,max$, go left

Idea Behind Interval-Search

Find the node with interval overlapping with $i = [10,14]$.

Notice that $i.low \le x.left.max$



When $i.low \le x.left.max$, go left

When $i.low > x.left.max$, go right

Interval-Search Pseudocode

Interval-Search Pseudocode

```
Interval-Search($T, i$):
```

Interval-Search Pseudocode

```
Interval-Search($T, i$):
    $x = T.root$
```

Interval-Search Pseudocode

```
**Interval-Search**$(T, i)$:
    $x = T.root$
    **while** $x \neq T.nil$ **and** $i$ does not overlap with $x.int$
```

Interval-Search Pseudocode

```
Interval-Search($T, i$):
    $x = T.root$
    while $x \neq T.nil$ and $i$ does not overlap with $x.int$
        if $x$ ______________________________
```

Interval-Search Pseudocode

```
**Interval-Search**($T, i$):
$x = T.root$
**while** $x \neq T.nil$ **and** $i$ does not overlap with $x.int$
    **if** $x$ ________________________________
        $x = x.left$
```

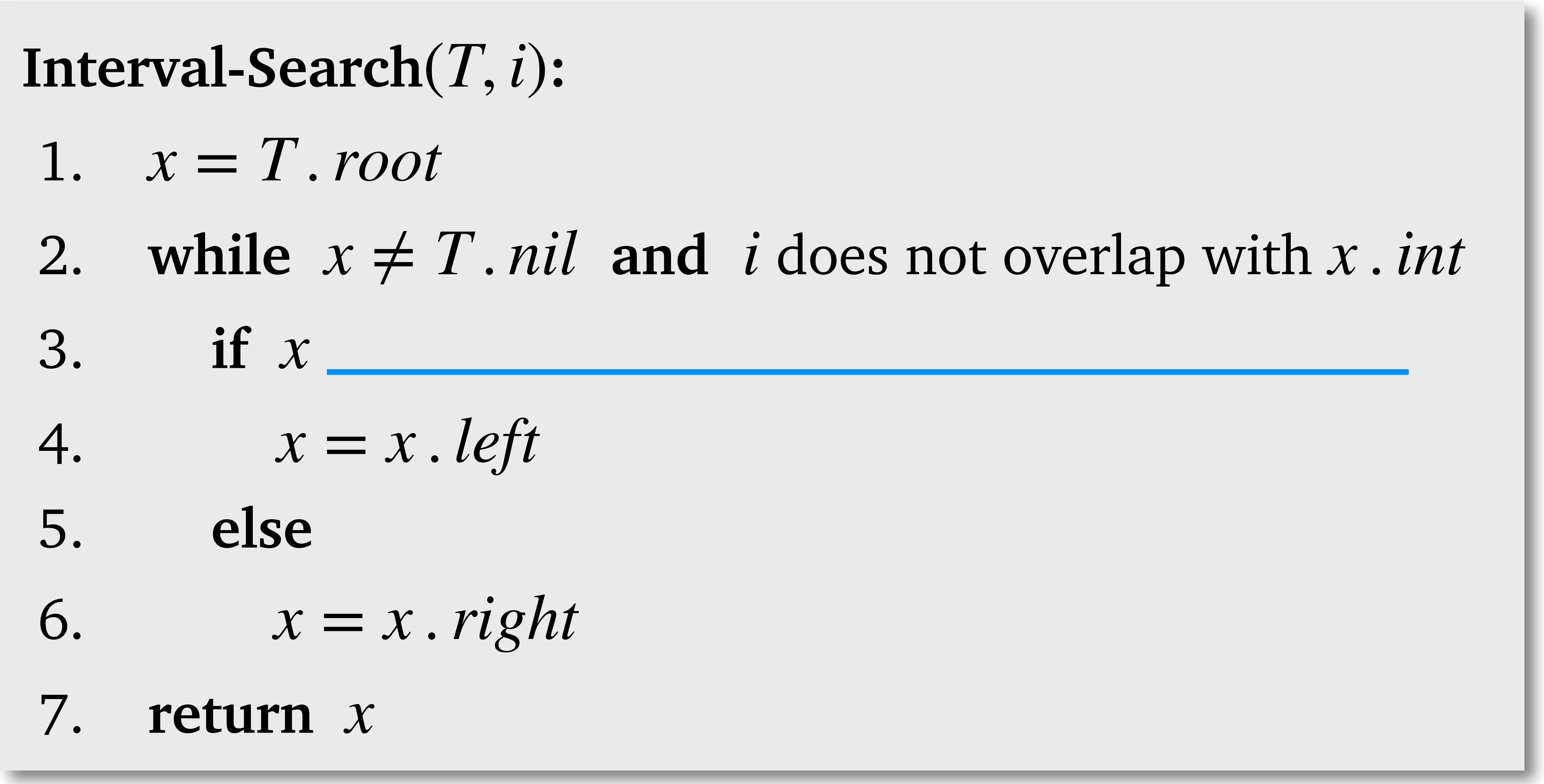
Interval-Search Pseudocode

```
**Interval-Search**($T, i$):
$x = T.root$
**while** $x \neq T.nil$ **and** $i$ does not overlap with $x.int$
    **if** $x$ ______________________________
        $x = x.left$
    **else**
```

Interval-Search Pseudocode

```
**Interval-Search**$(T, i)$:
while $x \neq T.nil$ and $i$ does not overlap with $x.int$
    if $x$ ________________________________
        $x = x.left$
    else
        $x = x.right$
```

Interval-Search Pseudocode

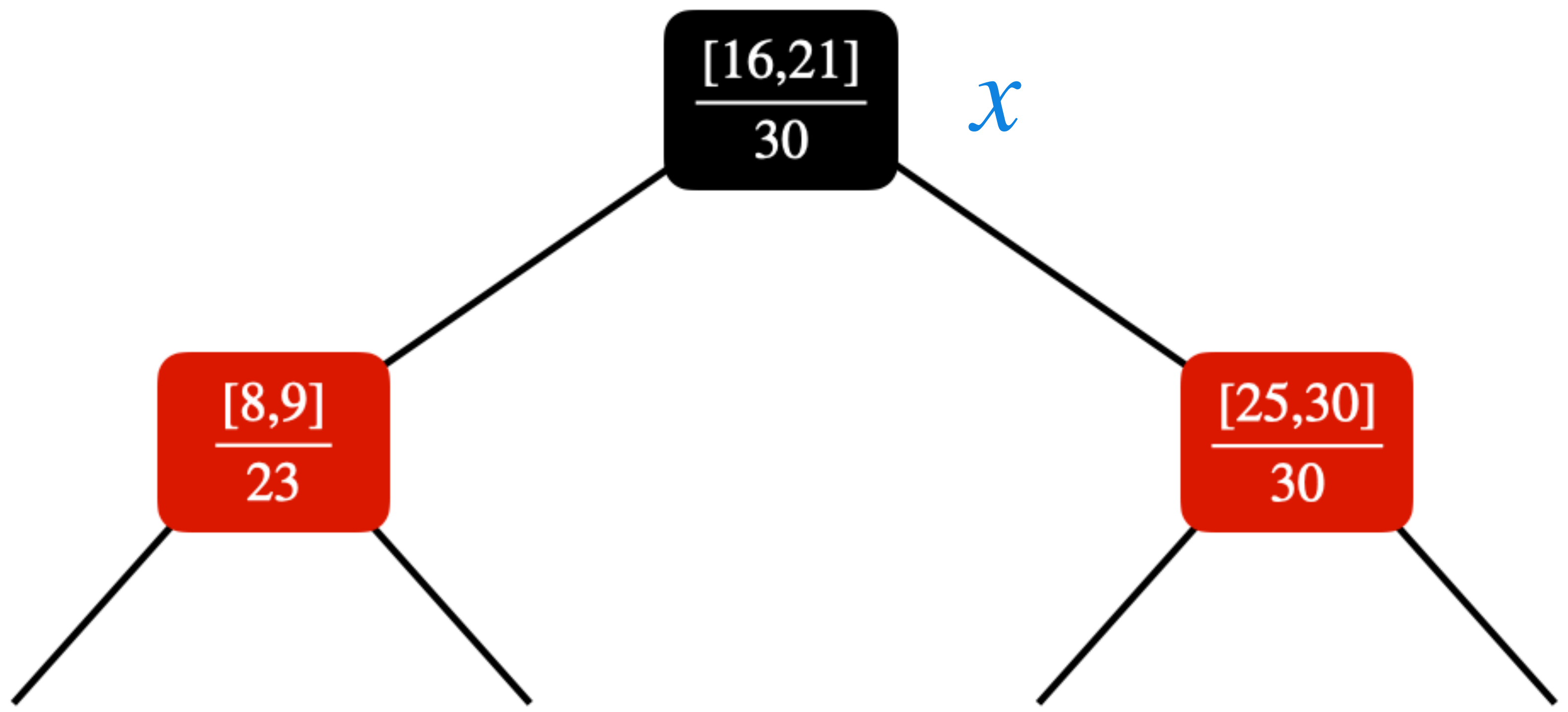


```
Interval-Search($T, i$):
x = T.root
while $x \neq T.nil$ and $i$ does not overlap with $x.int$
    if $x$ ________________________
        $x = x.left$
    else
        $x = x.right$
return $x$
```

Interval-Search Pseudocode

```
Interval-Search($T, i$):
$x = T.root$
while $x \neq T.nil$ and $i$ does not overlap with $x.int$
    if $x$ ____________________
        $x = x.left$
    else
        $x = x.right$
return $x$
```

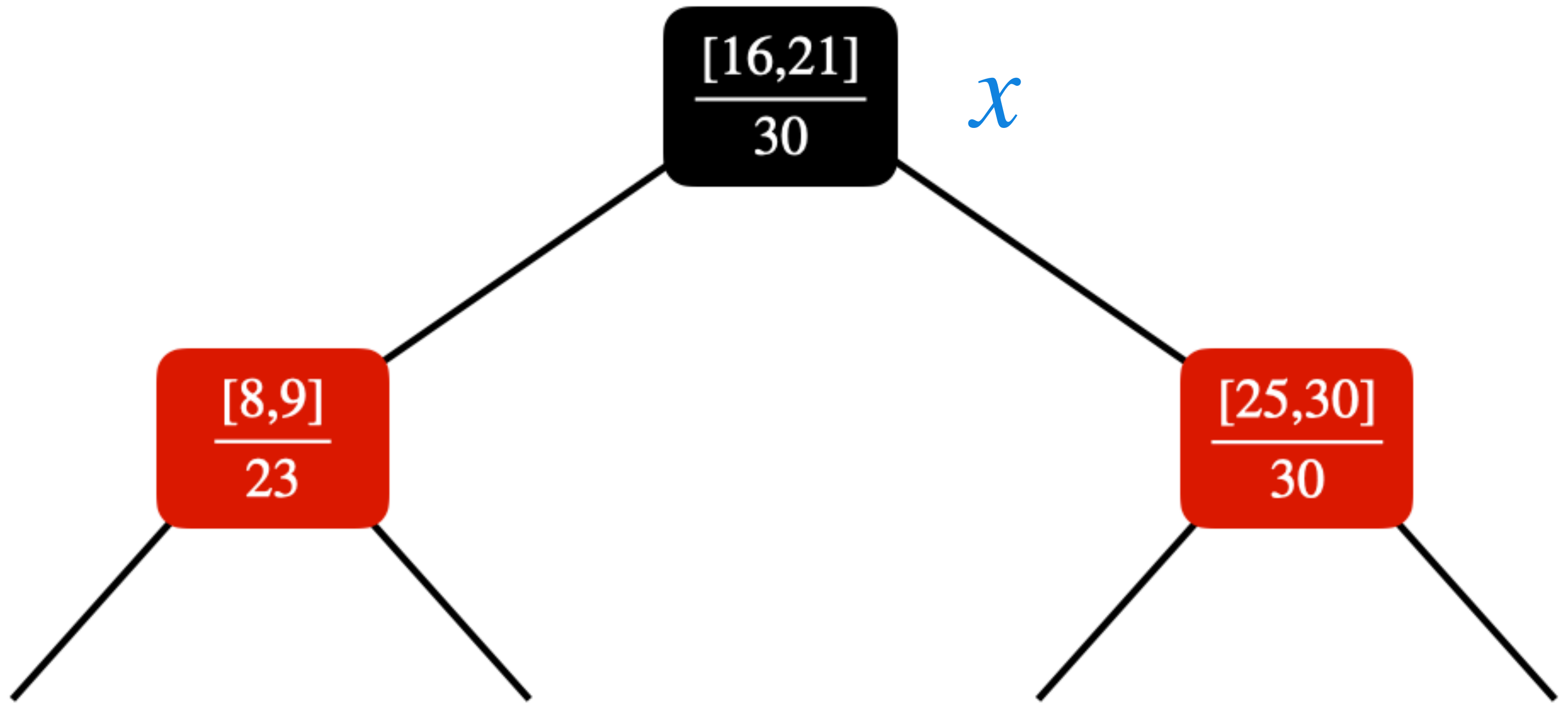
i = [10,14]



Interval-Search Pseudocode

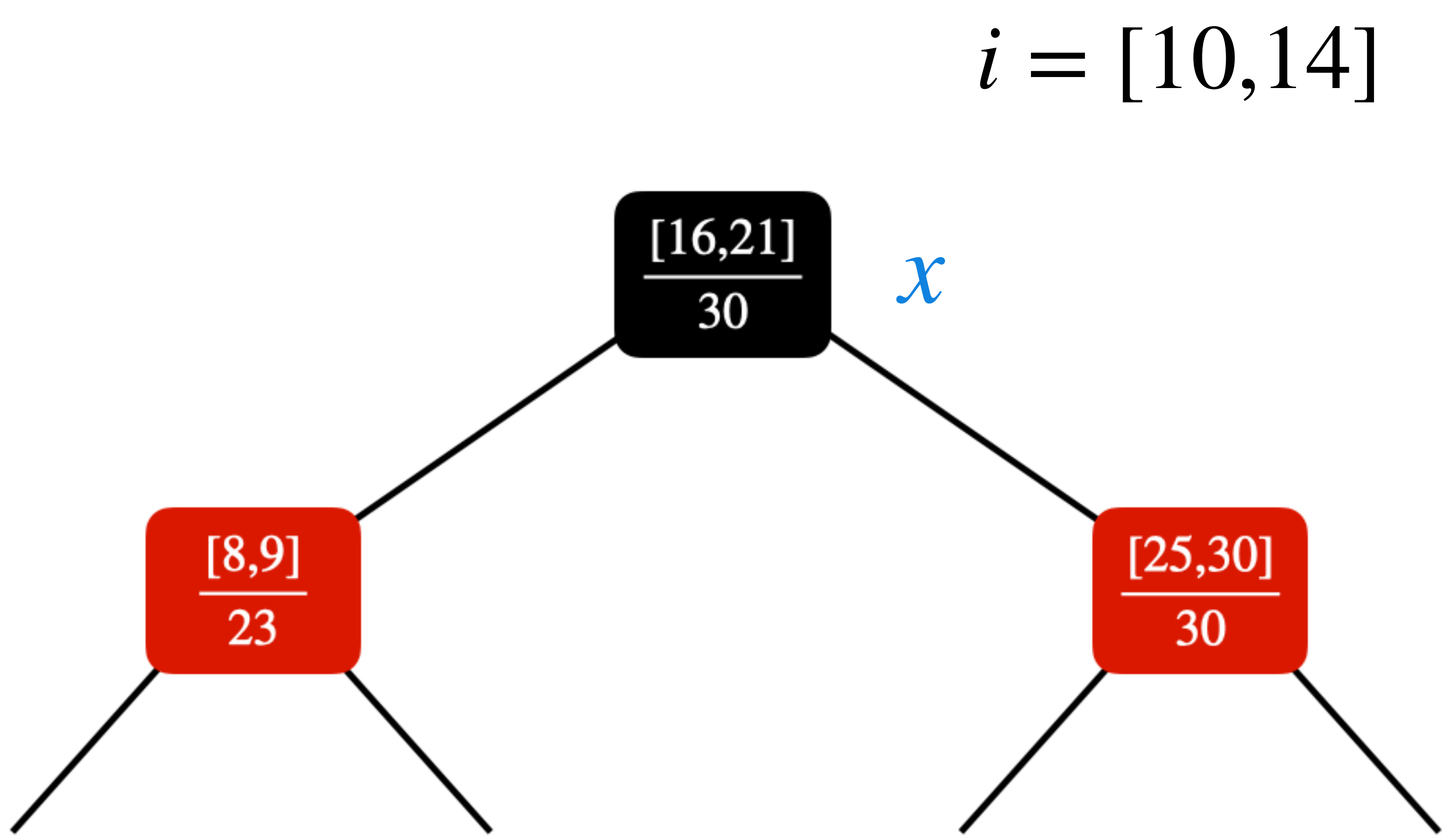
```
Interval-Search($T, i$):
    $x = T.root$
    while $x \neq T.nil$ and $i$ does not overlap with $x.int$
        if $x.left \neq T.nil$ and $x.left.max \geq i.low$
            $x = x.left$
        else
            $x = x.right$
    return $x$
```

$i = [10,14]$



Interval-Search Pseudocode

```
Interval-Search($T, i$):
    $x = T.root$
    while $x \neq T.nil$ and $i$ does not overlap with $x.int$
        if $x.left \neq T.nil$ and $x.left.max \geq i.low$
            $x = x.left$
        else
            $x = x.right$
    return $x$
```



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

Maintaining Subtree Max Highs

Maintaining Subtree Max Highs

Idea: Inserting or deleting an element will only affect the maximum values of its ancestor.

Maintaining Subtree Max Highs

Idea: Inserting or deleting an element will only affect the maximum values of its ancestor.

Similar to how we maintained sizes of the subtrees in the previous data structure.