CS2040S
Data Structures and Algorithms
(e-learning edition)

Augmented Trees!
Part 2

Today

Three examples of augmenting balanced BSTs

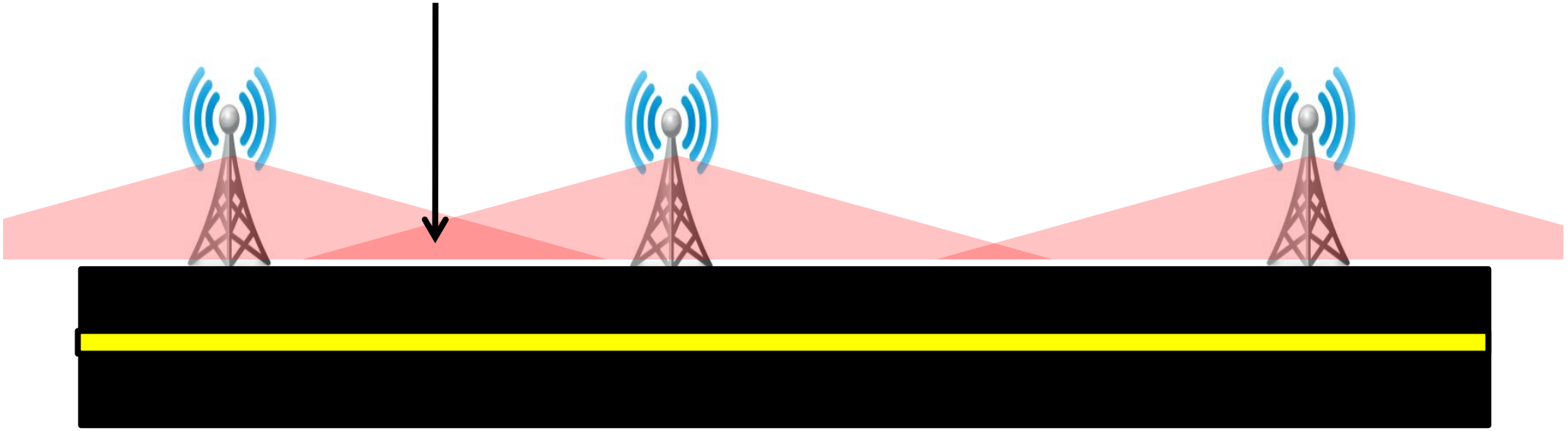
1. Order Statistics

2. Intervals

3. Orthogonal Range Searching

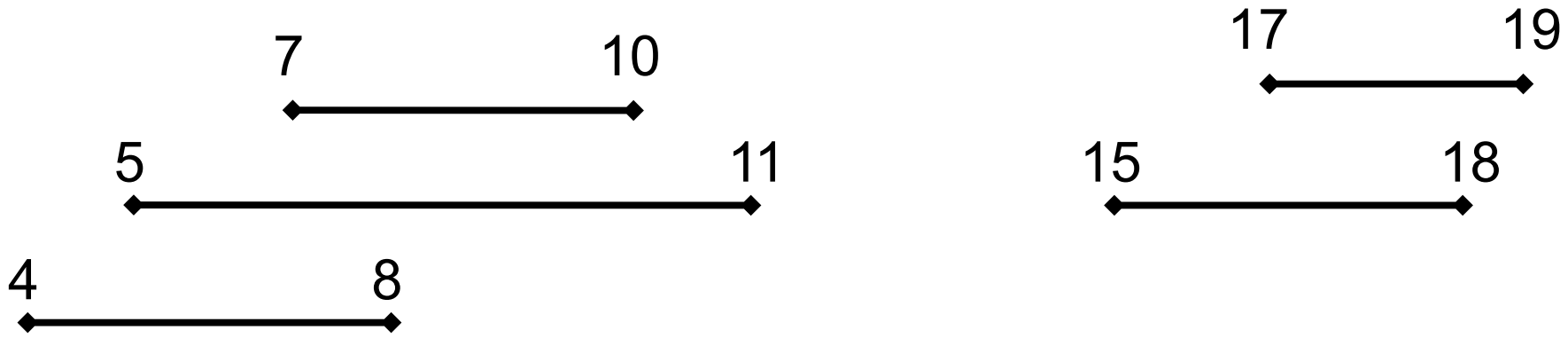
Cell Tower Coverage

Find a tower that covers my location.



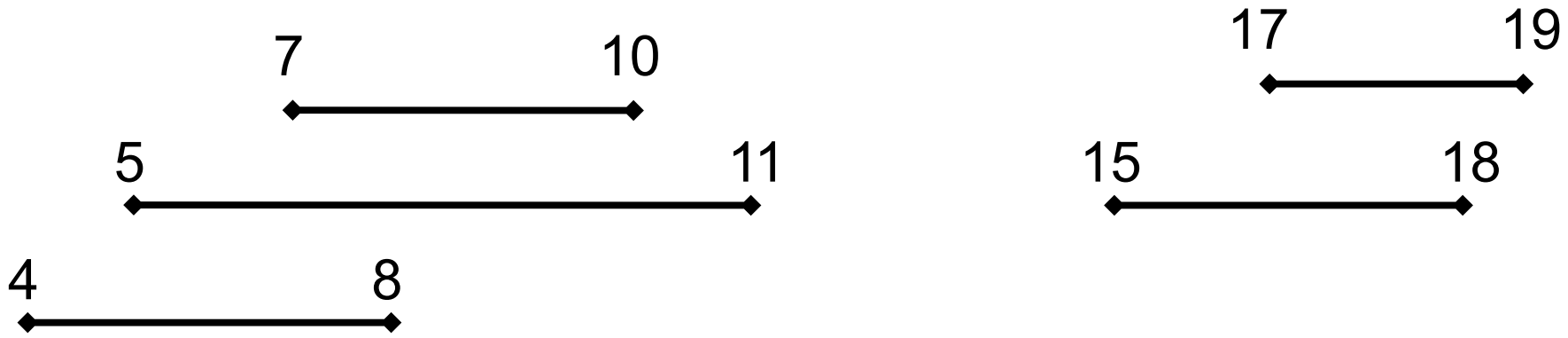
Cell Tower Coverage

Find a tower that covers my location.



Cell Tower Coverage

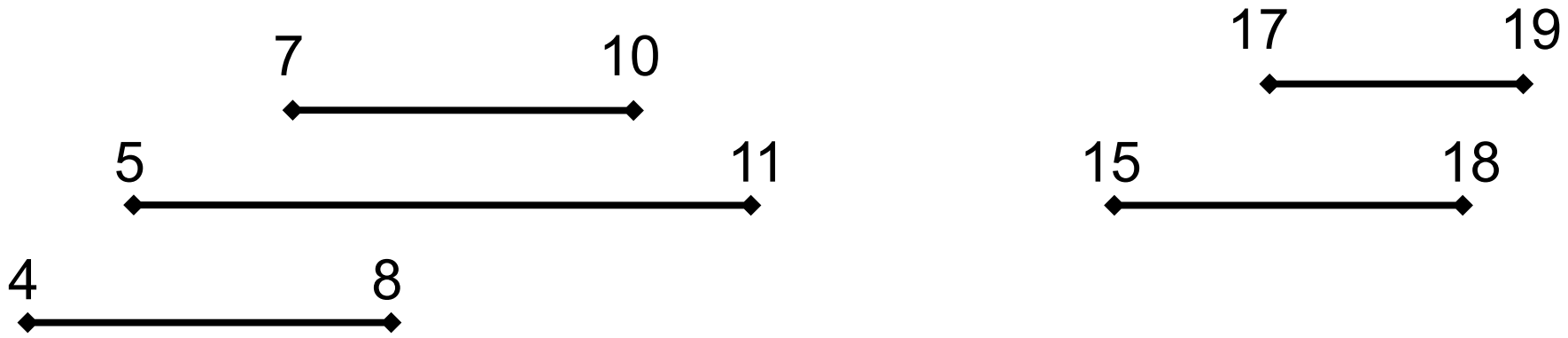
Find a tower that covers my location.



insert(begin, end)
delete(begin, end)

Cell Tower Coverage

Find a tower that covers my location.

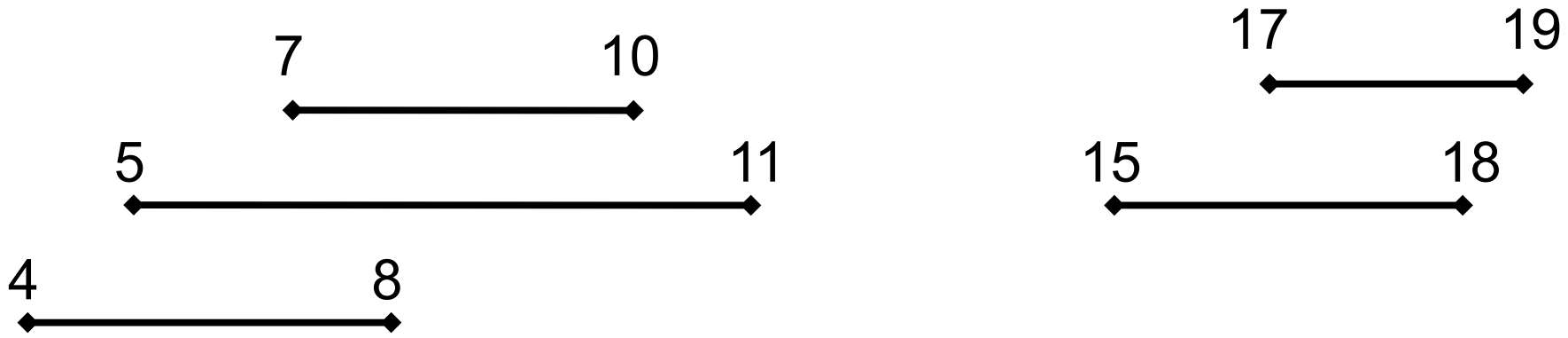


insert(begin, end)
delete(begin, end)

query(x): find an interval that overlaps x.

Cell Tower Coverage

Find a tower that covers my location.



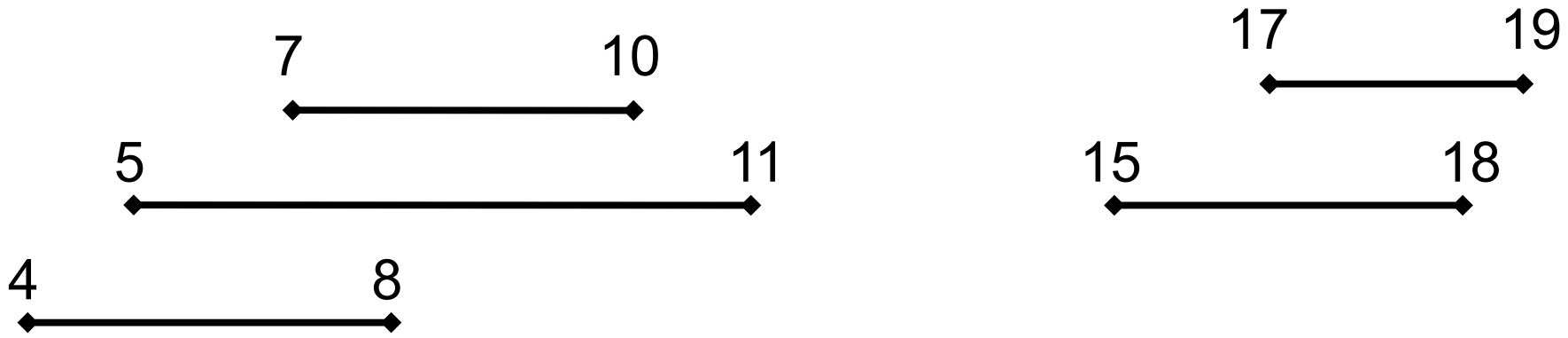
Idea 1: Keep intervals in a list.

Query: scan entire list.

Does sorting help?

Cell Tower Coverage

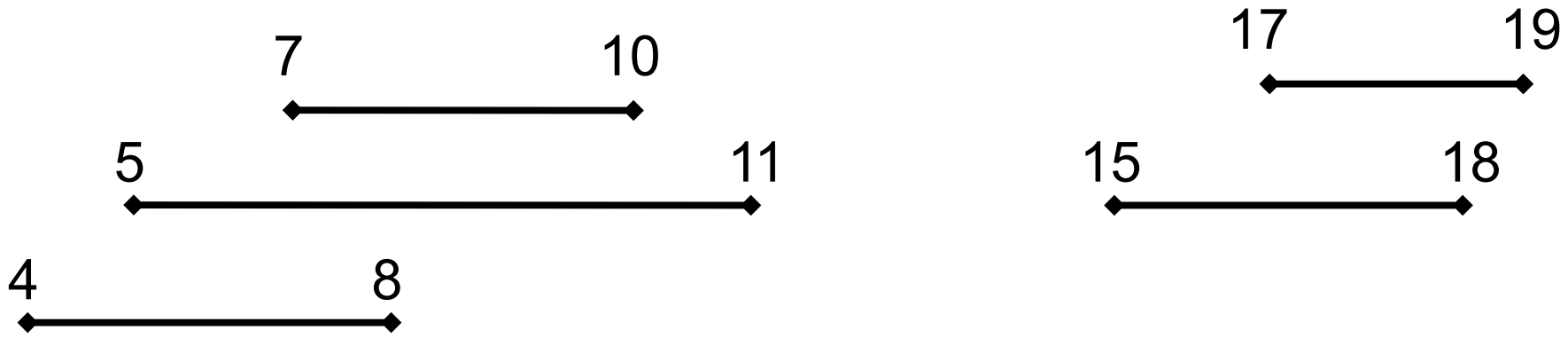
Find a tower that covers my location.



Idea 2: O(1) queries??

Cell Tower Coverage

Find a tower that covers my location.



Idea 2: O(1) queries

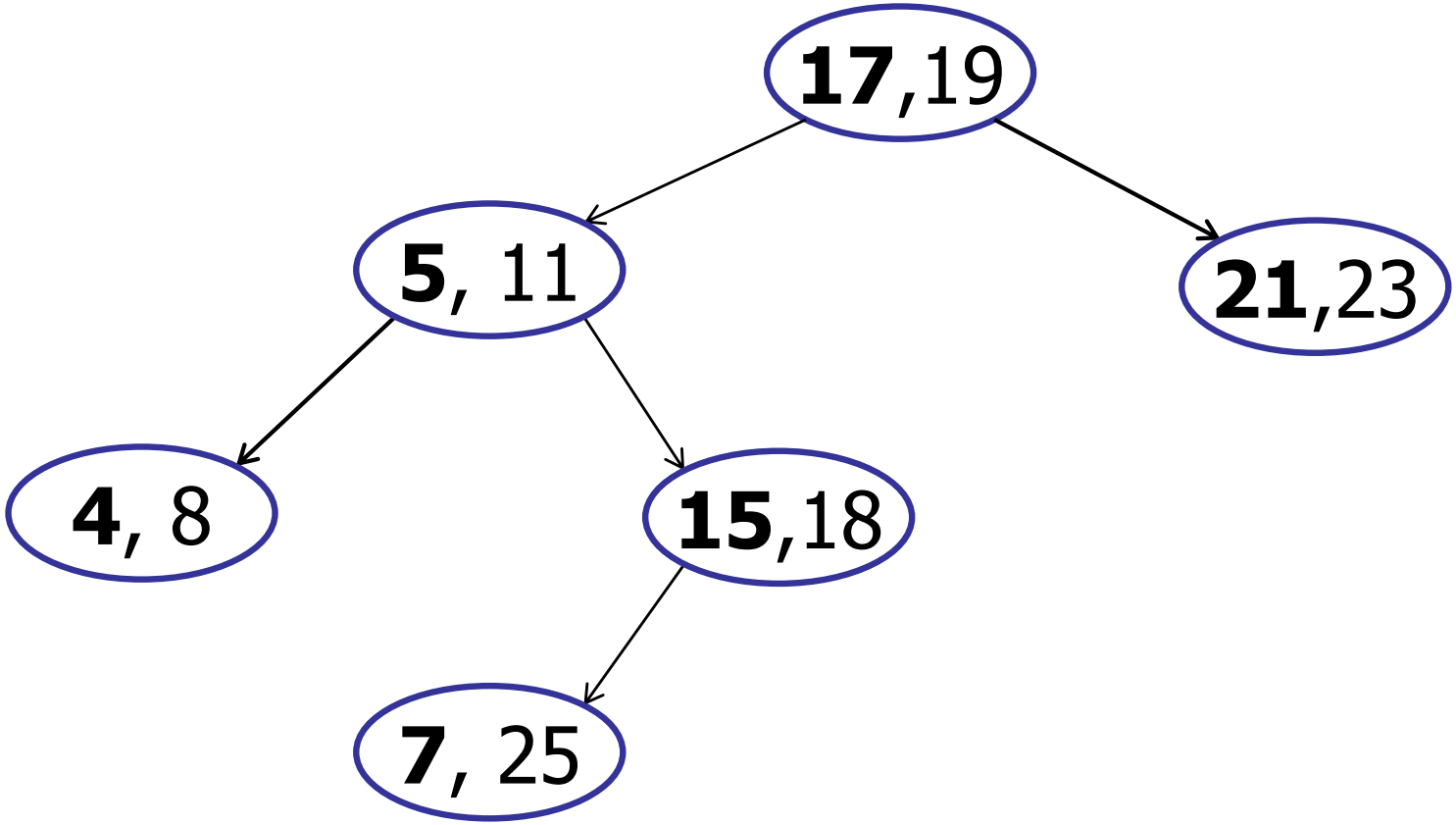
			A	A	A	A	A	B	B	C				D	D	D	D	E	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Problems??

Idea 3: Interval Trees

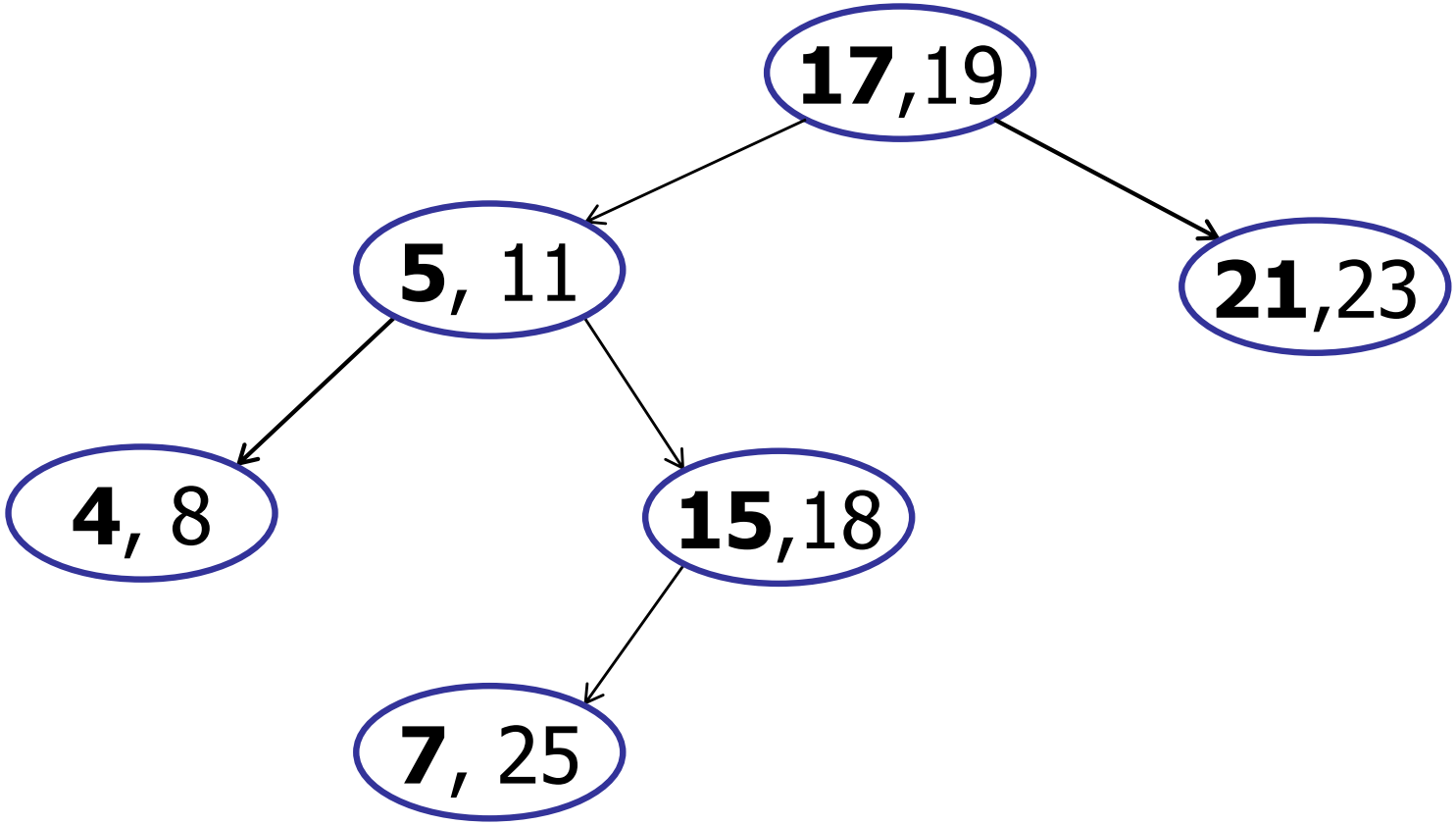
Interval Trees

Each node is an interval



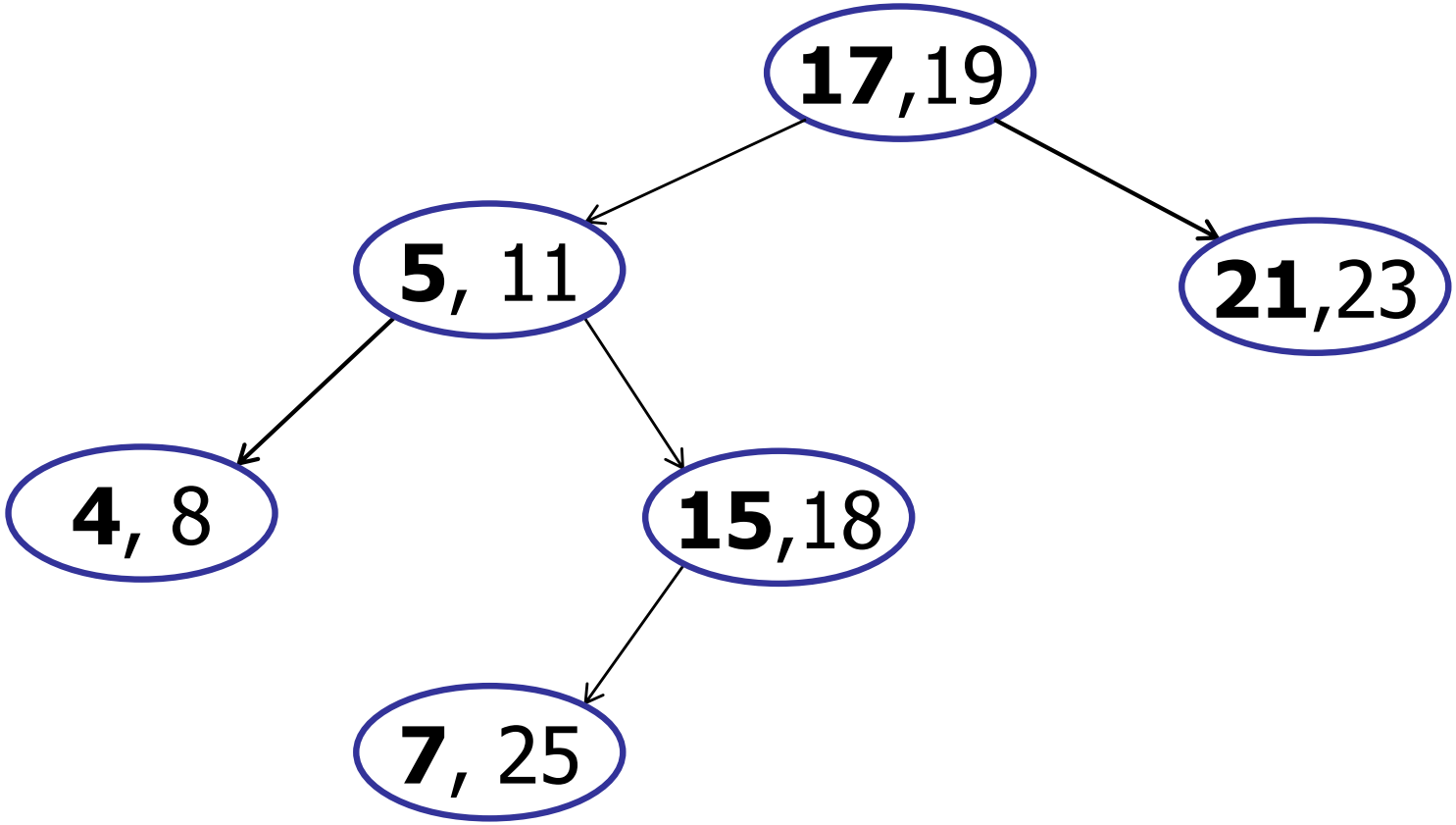
Interval Trees

Sorted by left endpoint



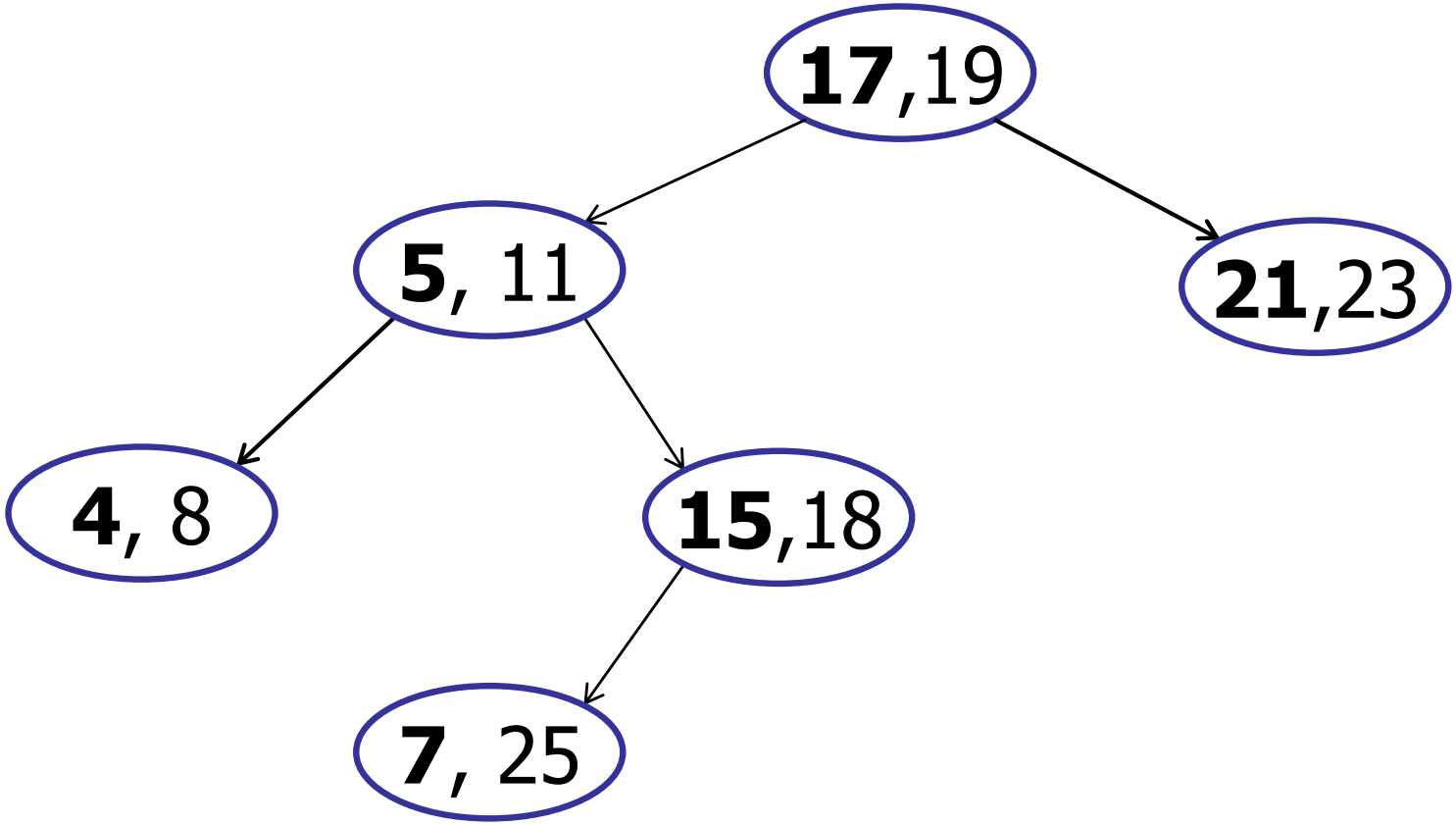
Interval Trees

search-interval(25) = ?



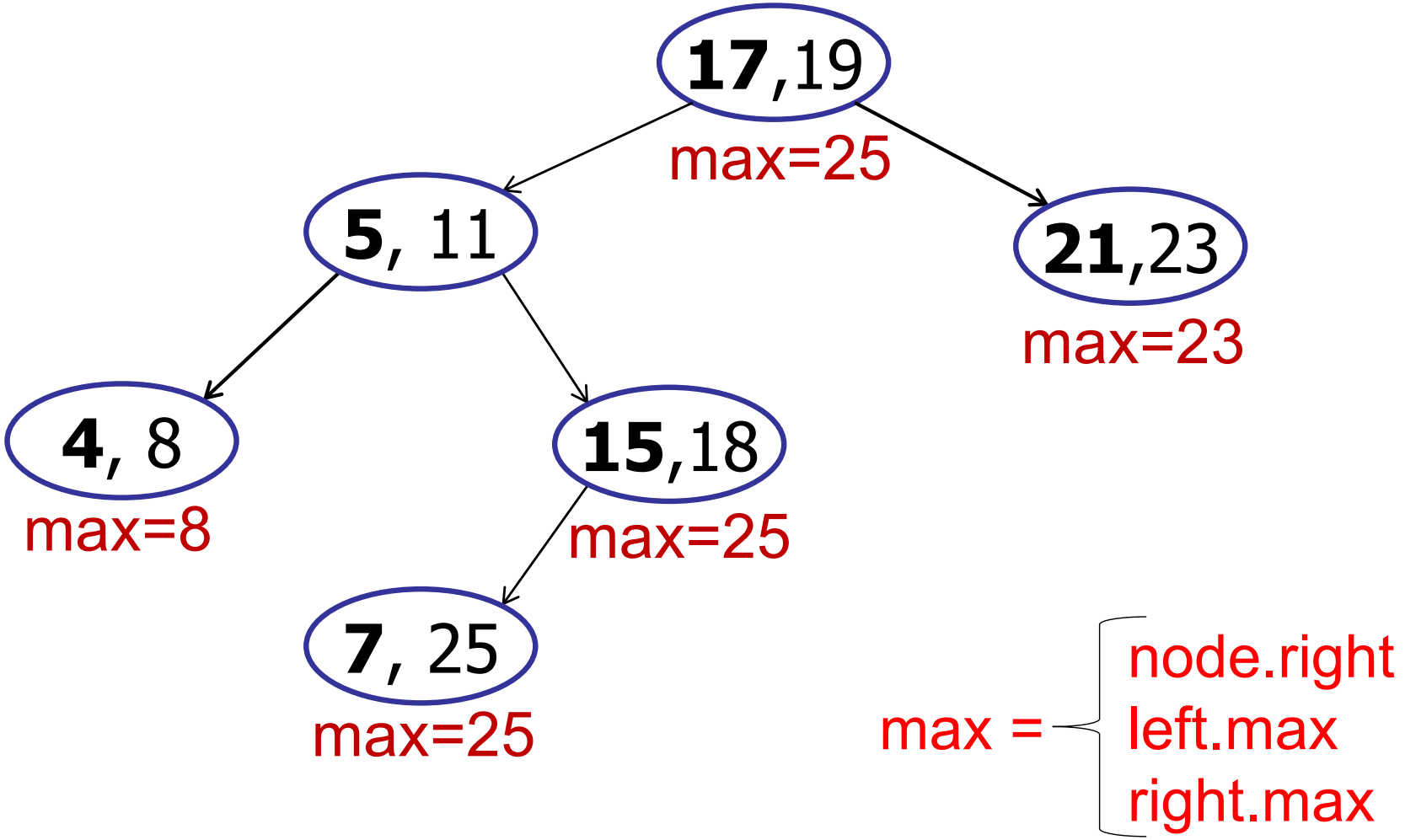
Interval Trees

Augment: ??

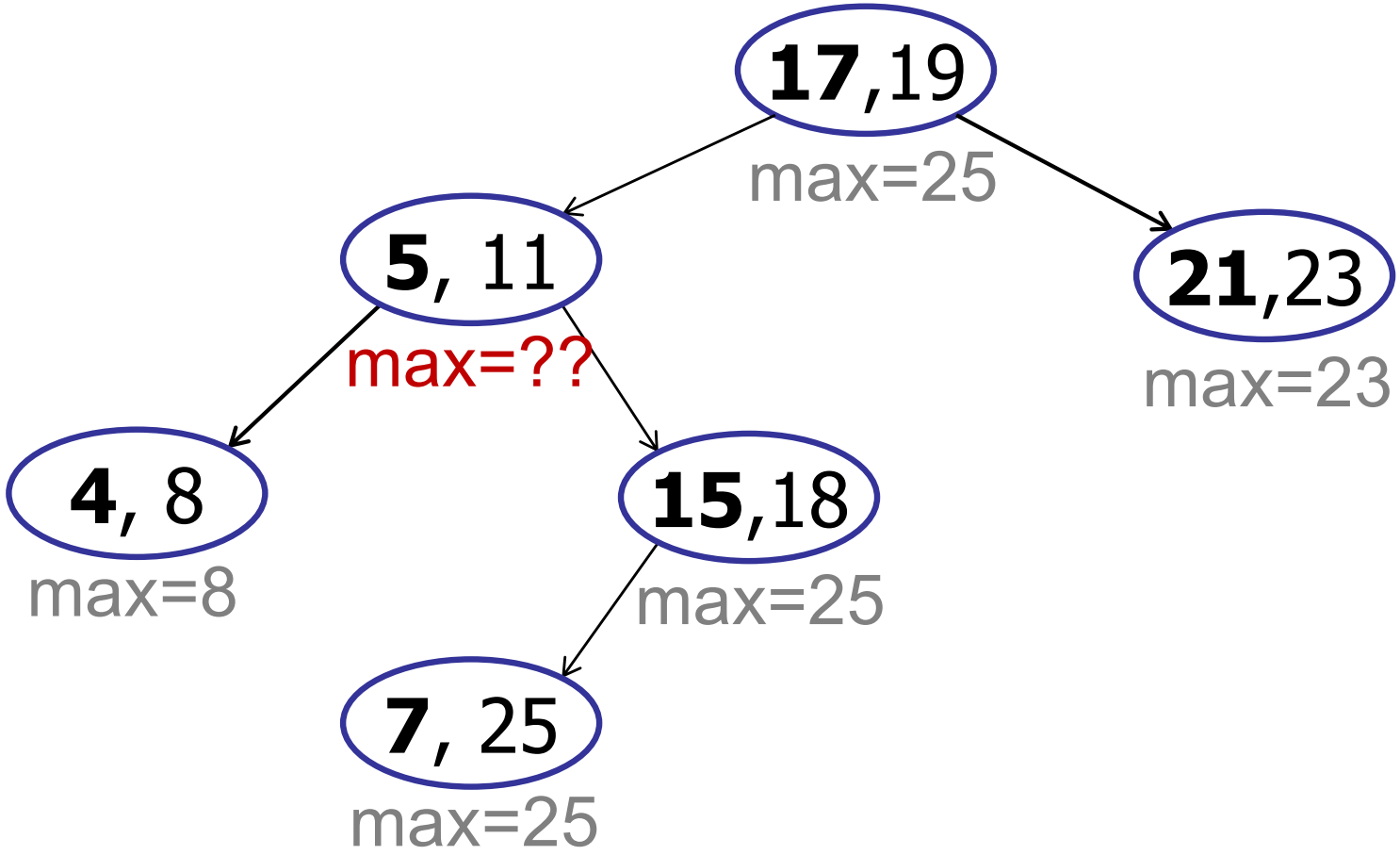


Interval Trees

Augment: maximum endpoint in subtree



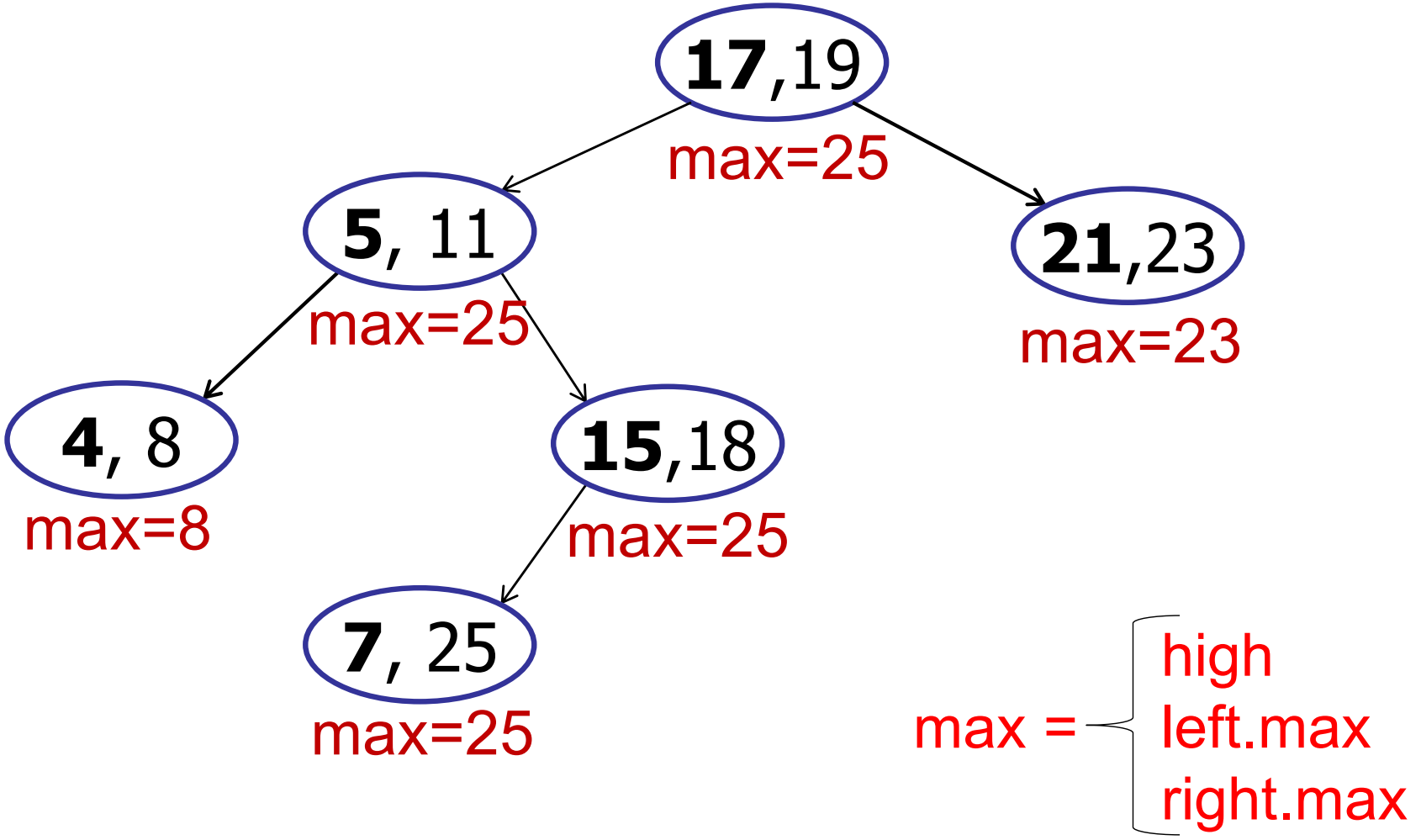
max=??



1. 5
2. 8
3. 11
4. 18
5. 25 ✓
6. 19

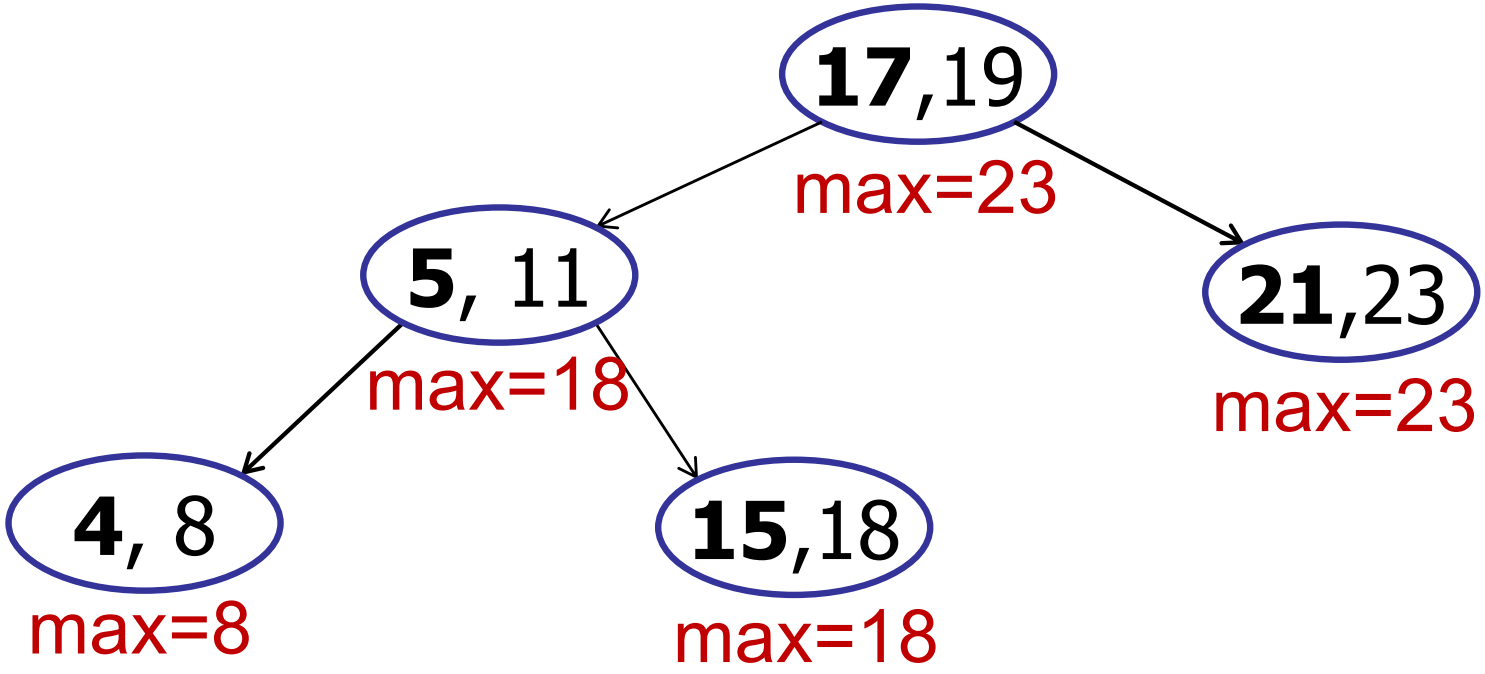
Interval Trees

Augment: maximum endpoint in subtree



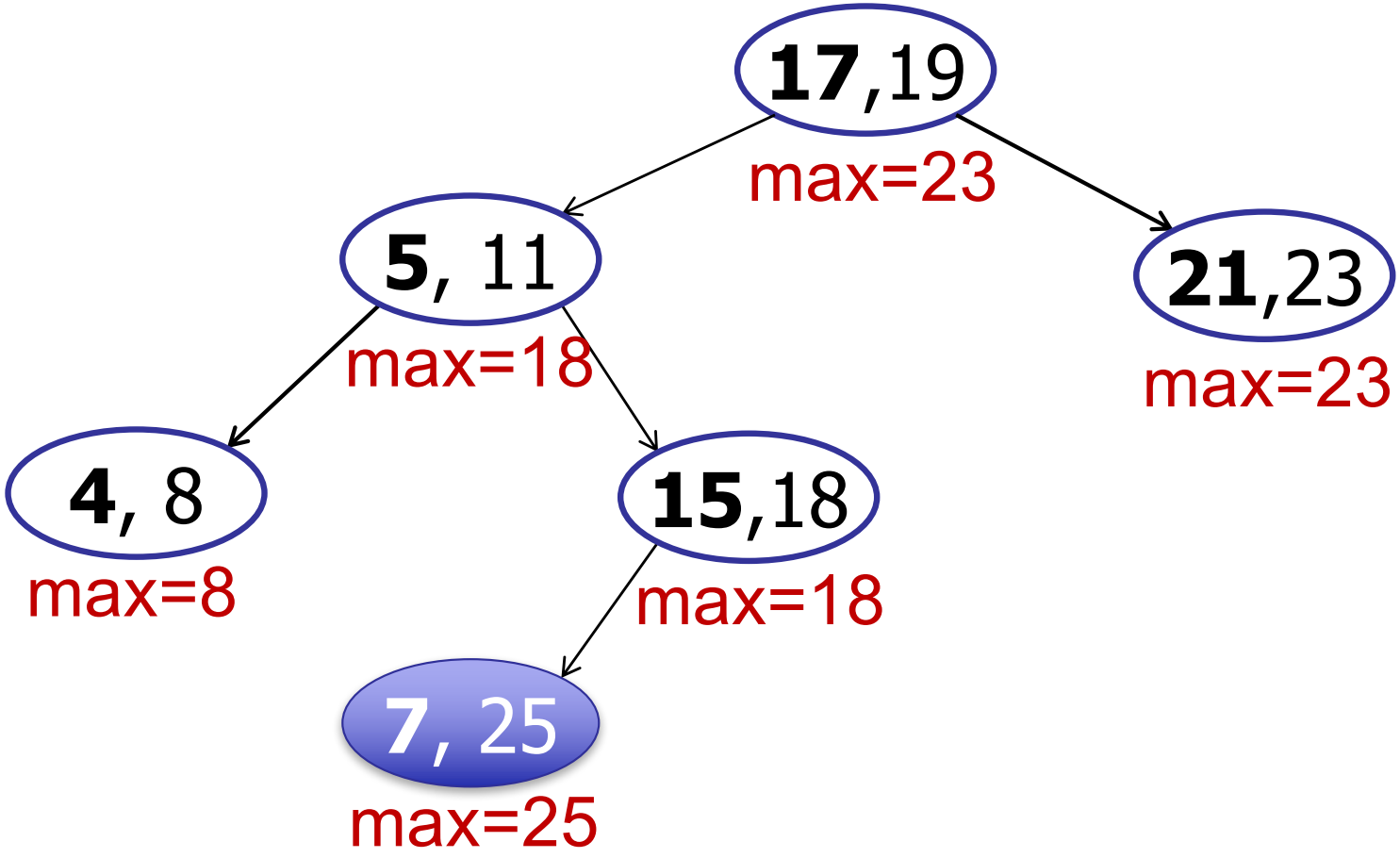
Interval Trees

Insertion: *example* – **insert(7, 25)**



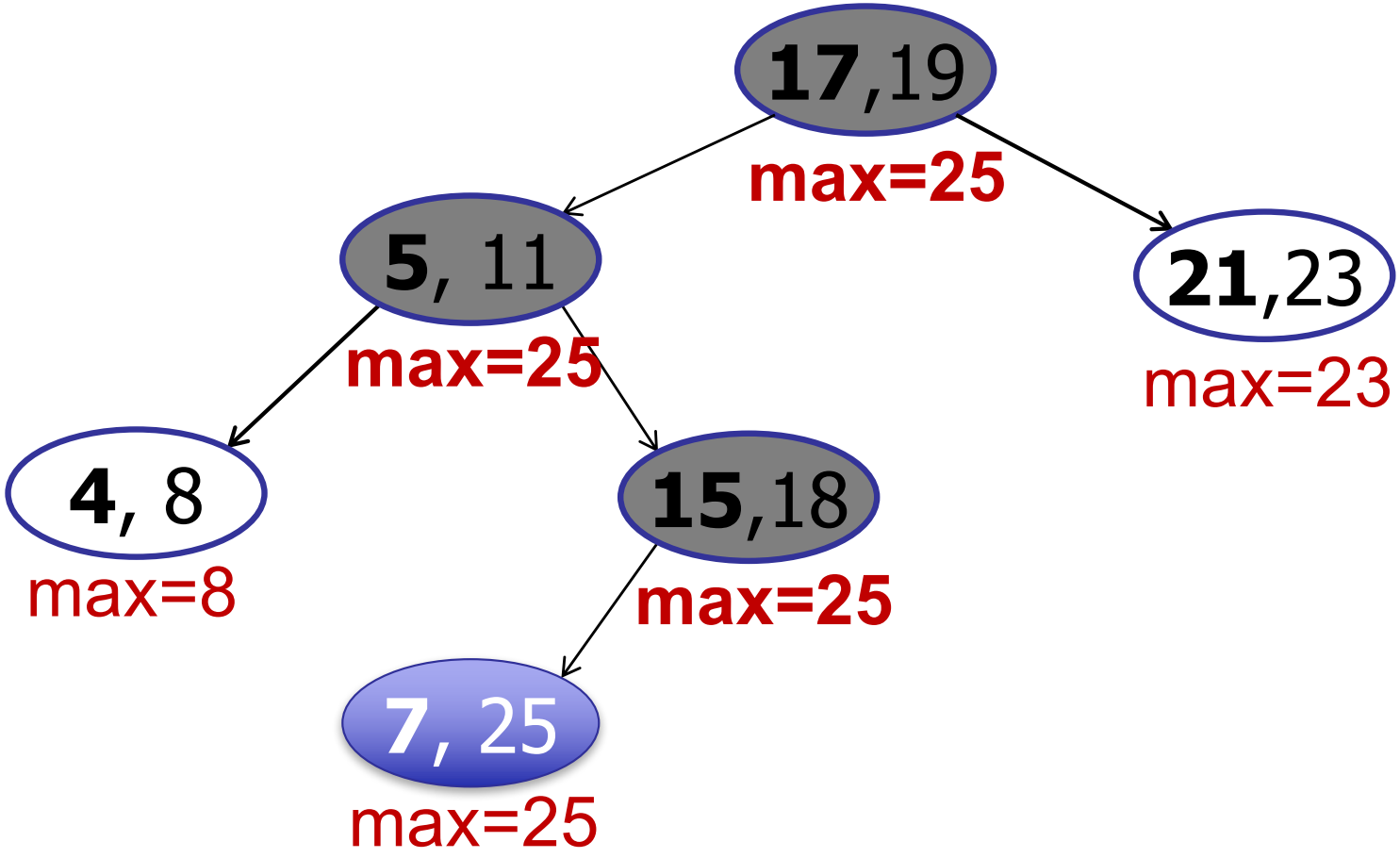
Interval Trees

Insertion: *example* – **insert(7, 25)**



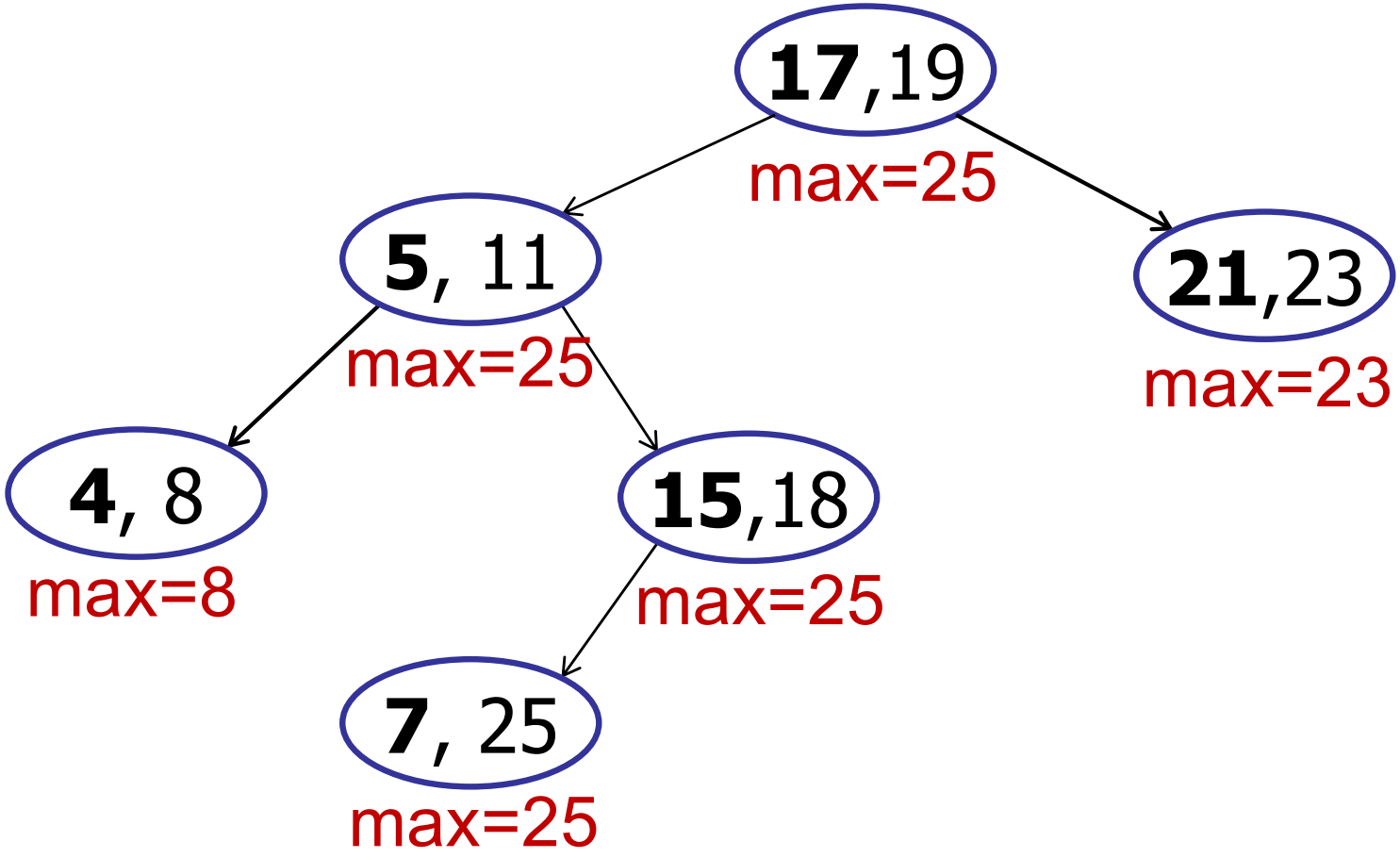
Interval Trees

Insertion: *example* – **insert(7, 25)**



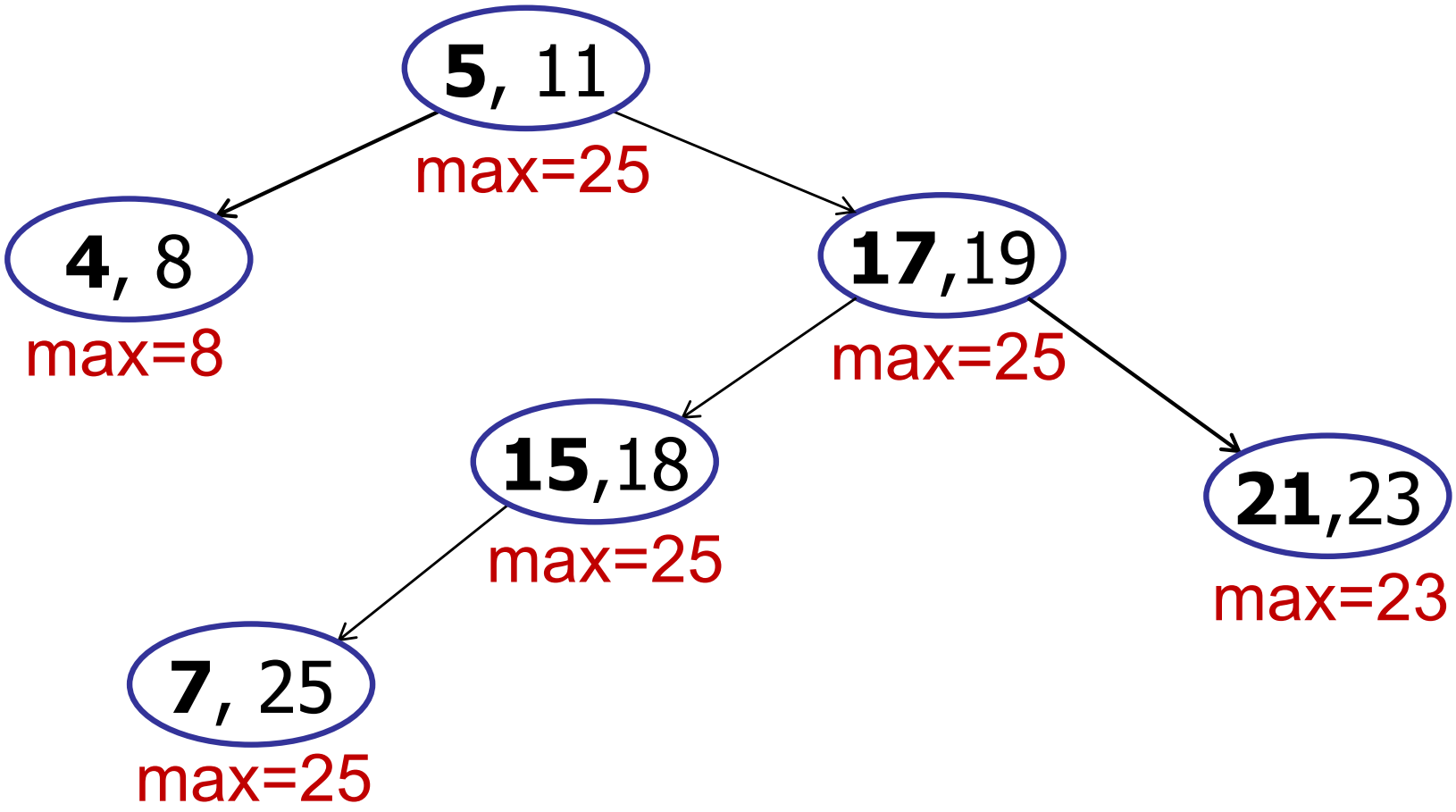
Interval Trees

Insertion: *out-of-balance*



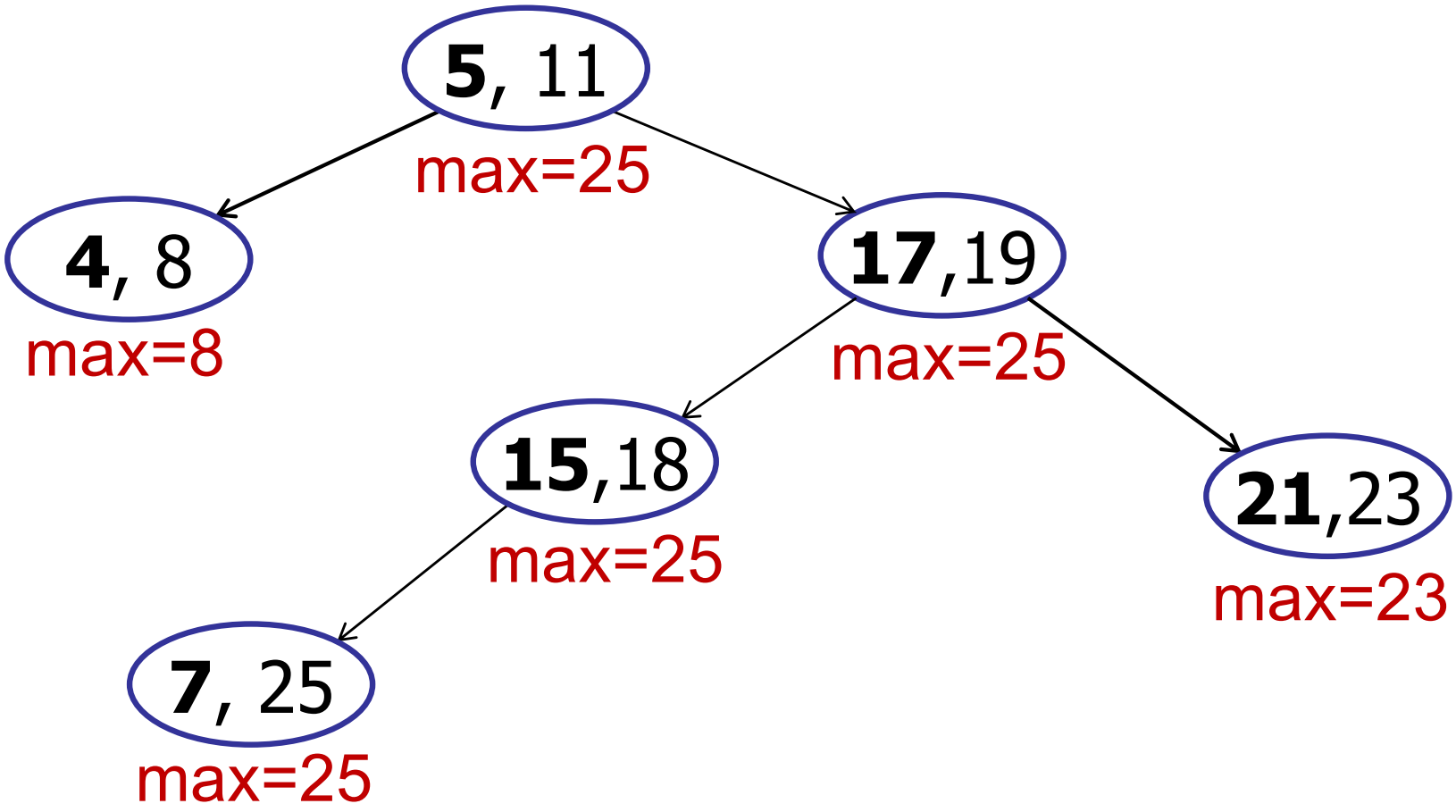
Interval Trees

Insertion: right-rotate (17, 19)



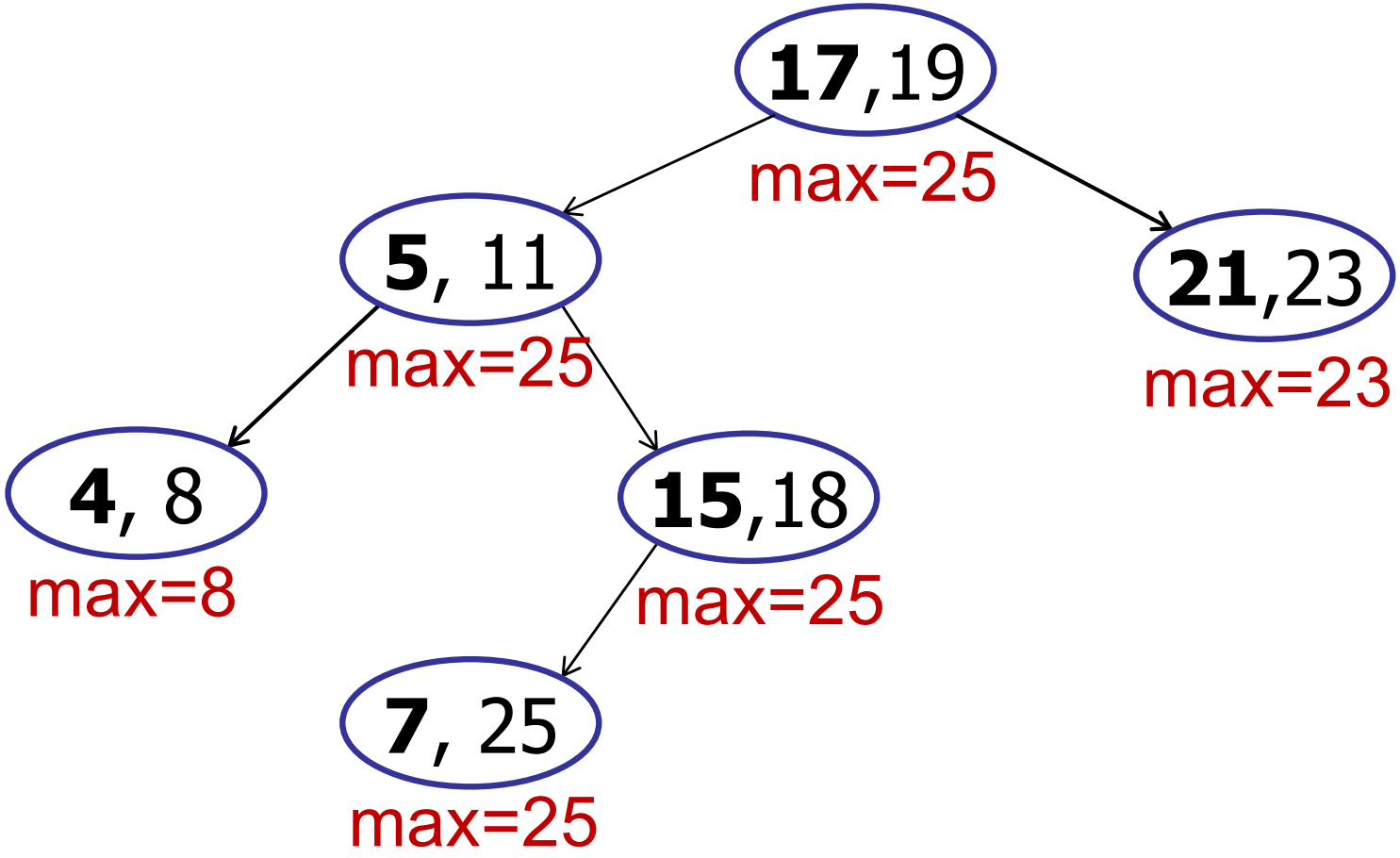
Interval Trees

Insertion: *right-rotate (17, 19)*, OOPS!



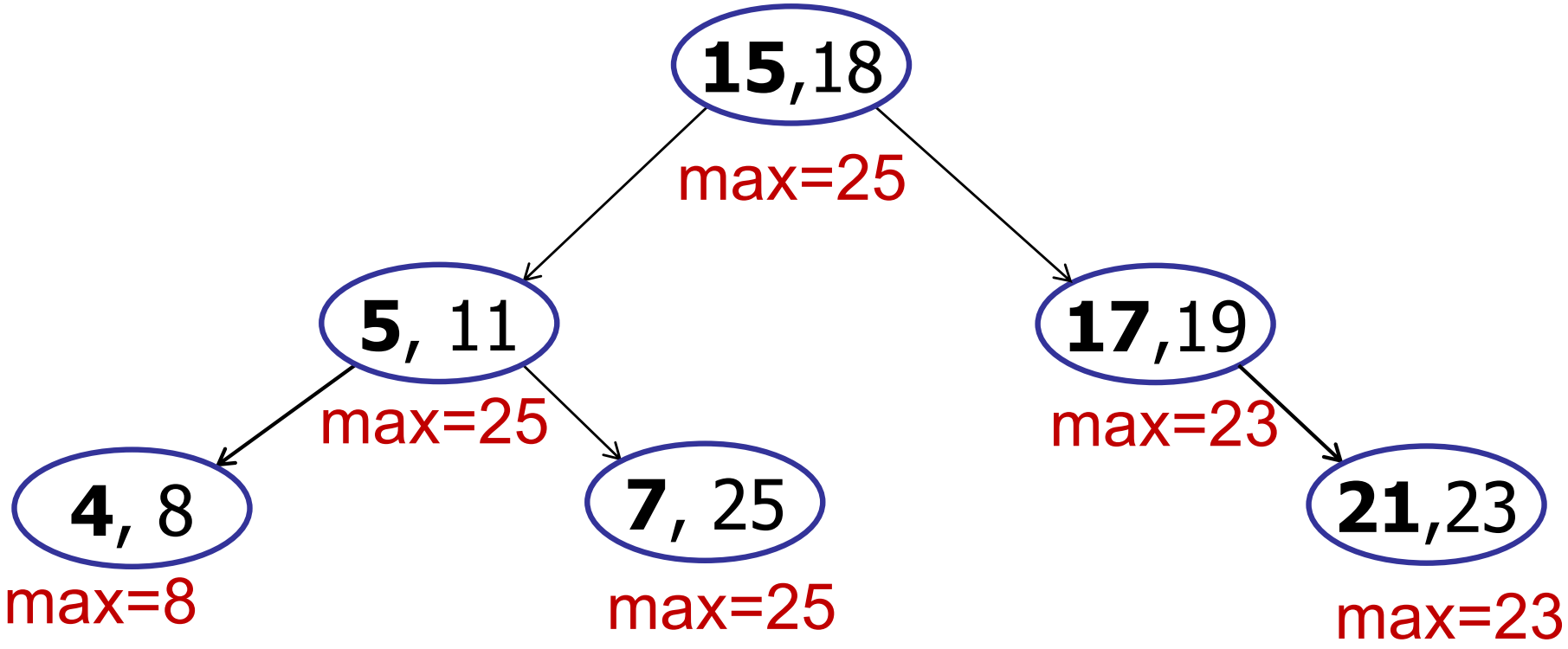
Interval Trees

Insertion: *out-of-balance*



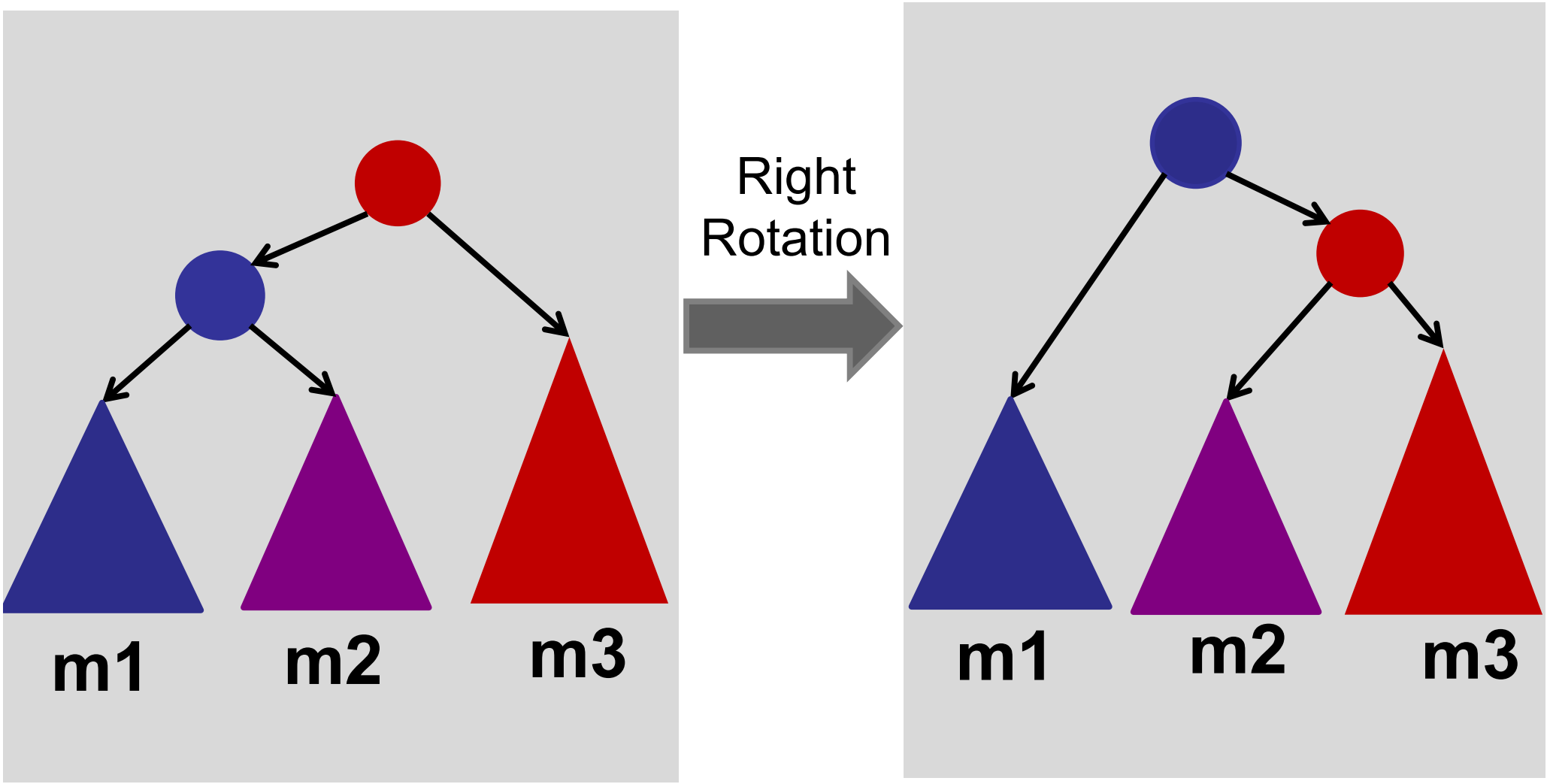
Interval Trees

Insertion: left-rotate, right-rotate



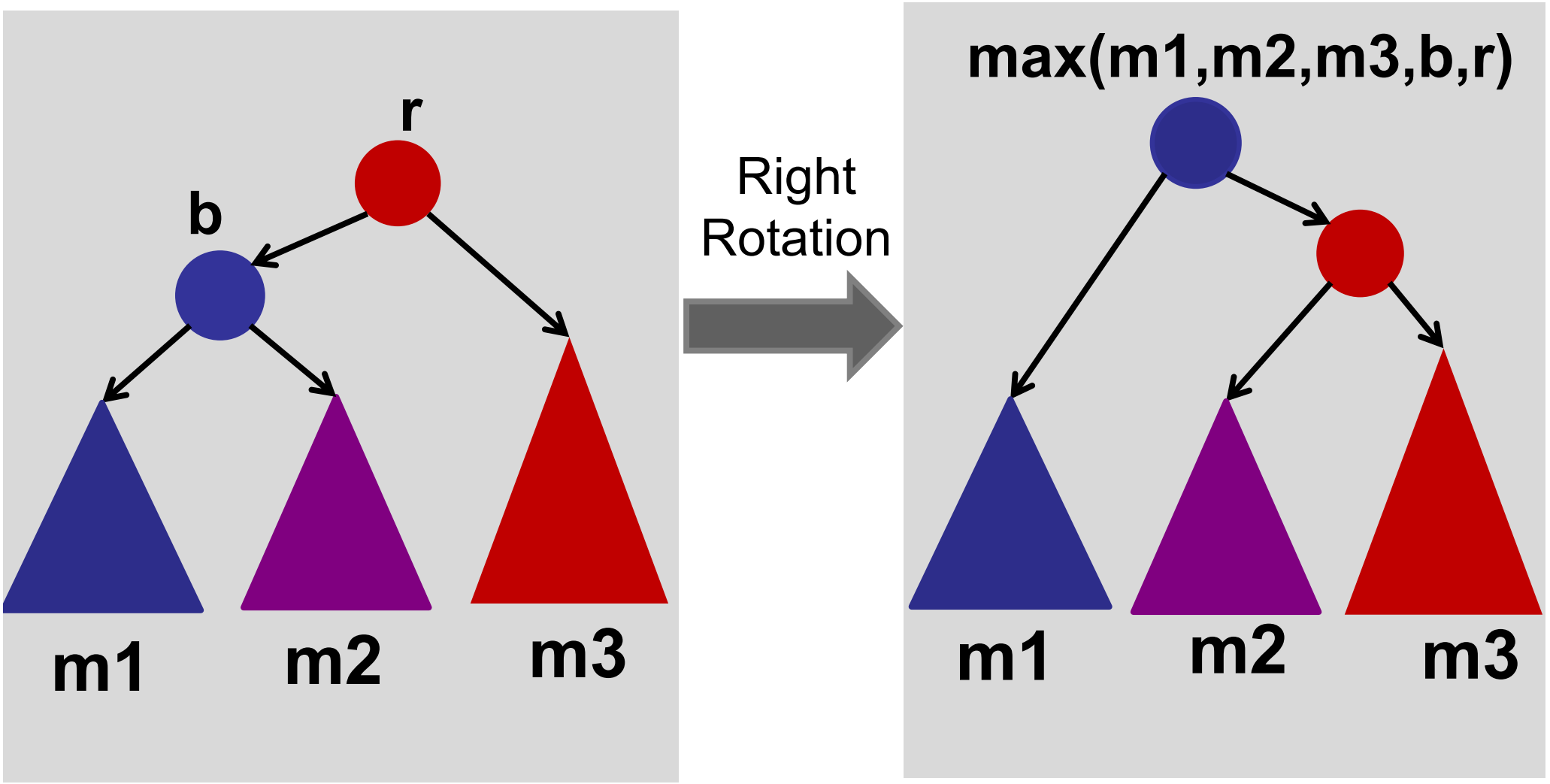
Interval Trees

Maintain MAX during rotations:



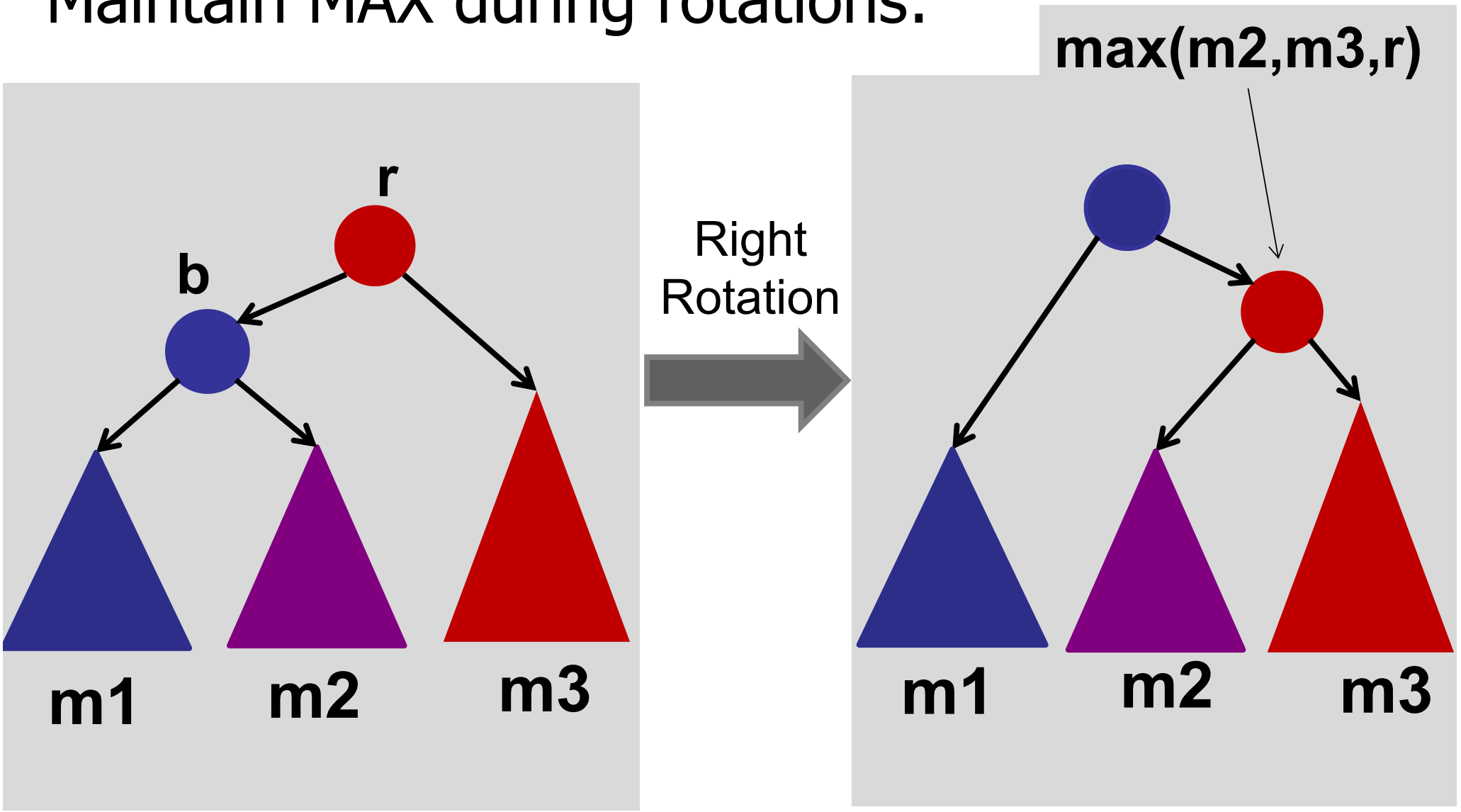
Interval Trees

Maintain MAX during rotations:



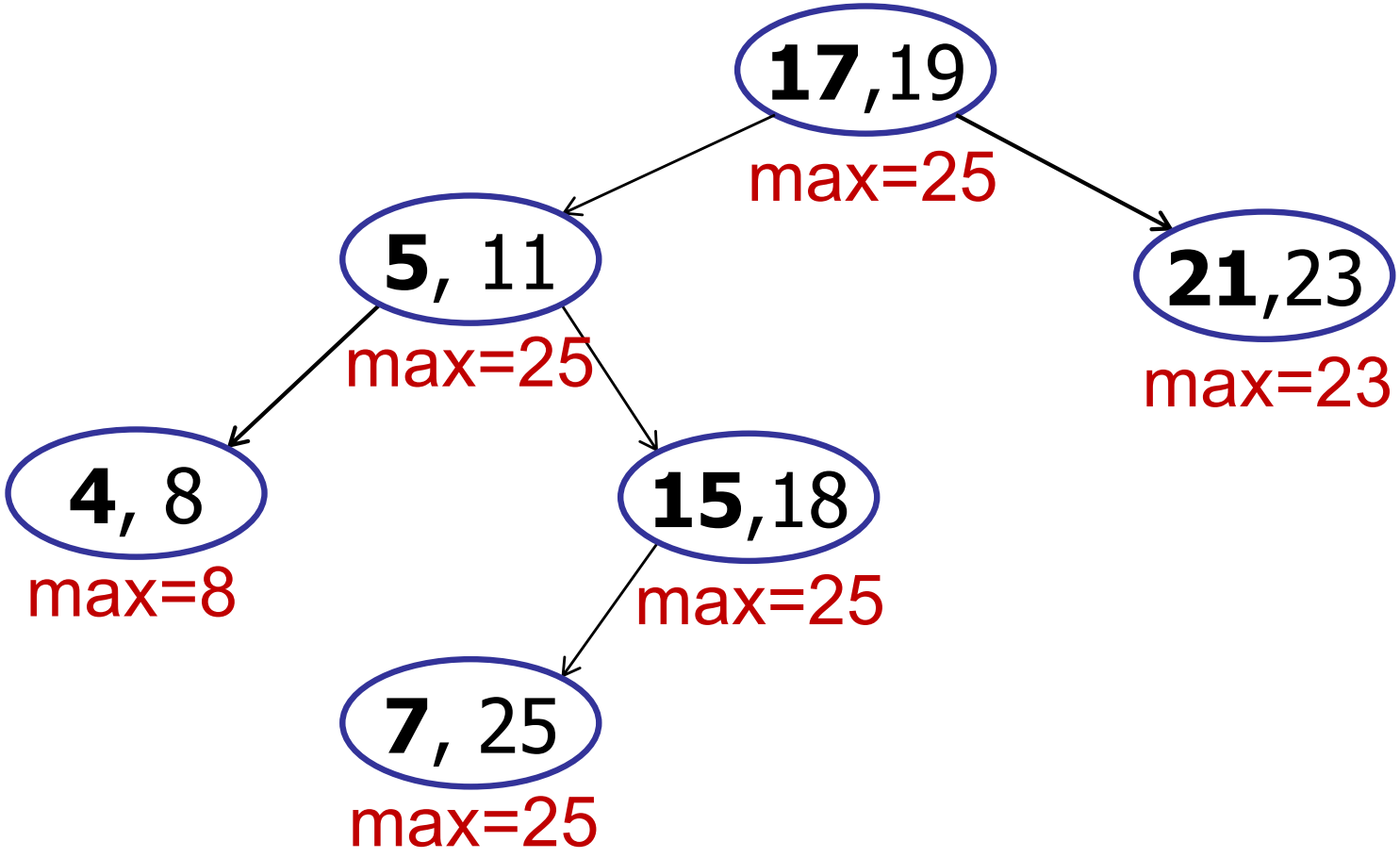
Interval Trees

Maintain MAX during rotations:



Interval Trees

Searching: interval-search(22)



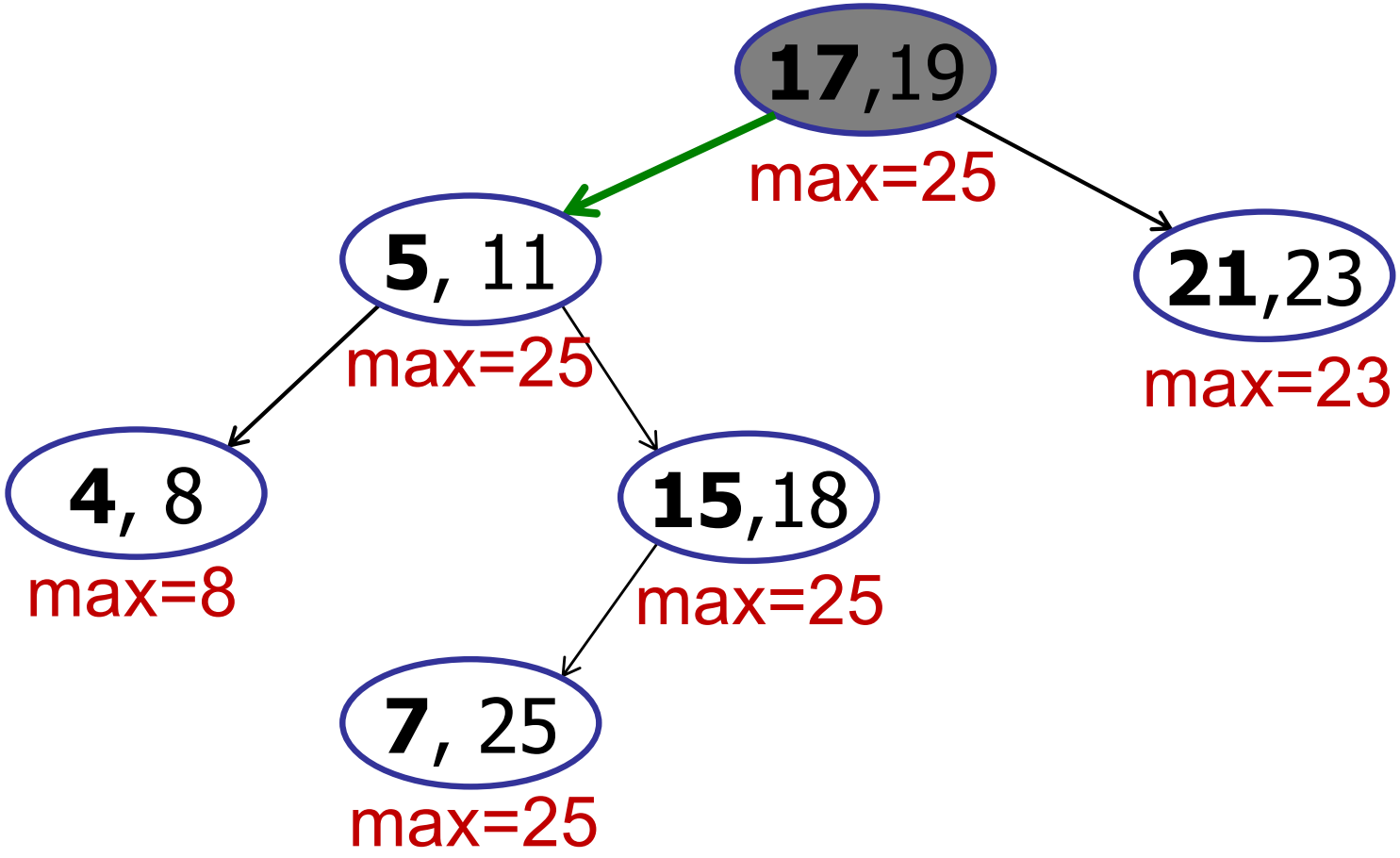
Interval Trees

interval-search(x) : find interval containing x

```
interval-search(x)
        c = root;
        while (c != null and x is not in c.interval) do
                if (c.left == null) then
                        c = c.right;
                else if  (x > c.left.max)  then
                        c = c.right;
                else c = c.left;
        return c.interval;
```

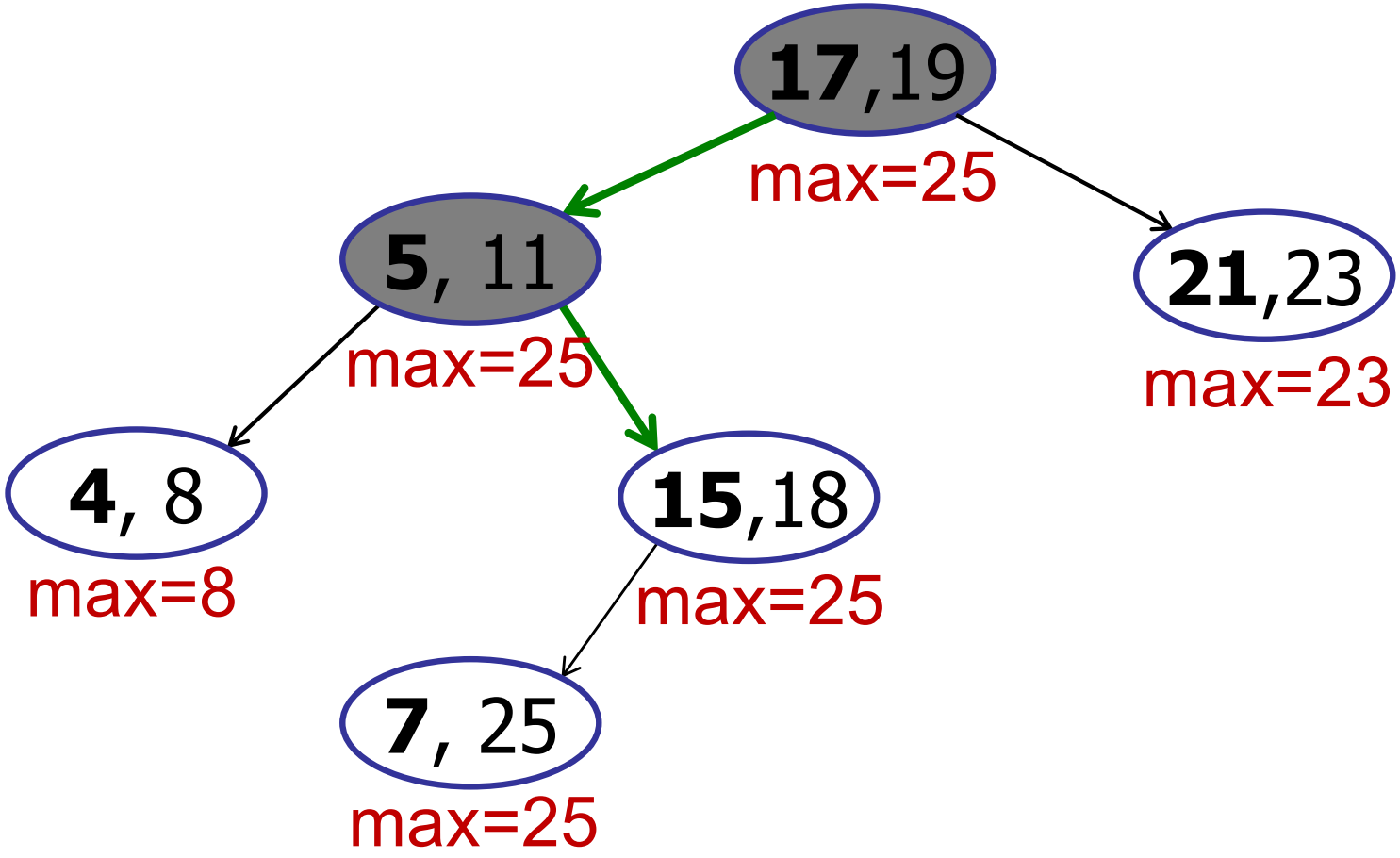
Interval Trees

Searching: interval-search(22)



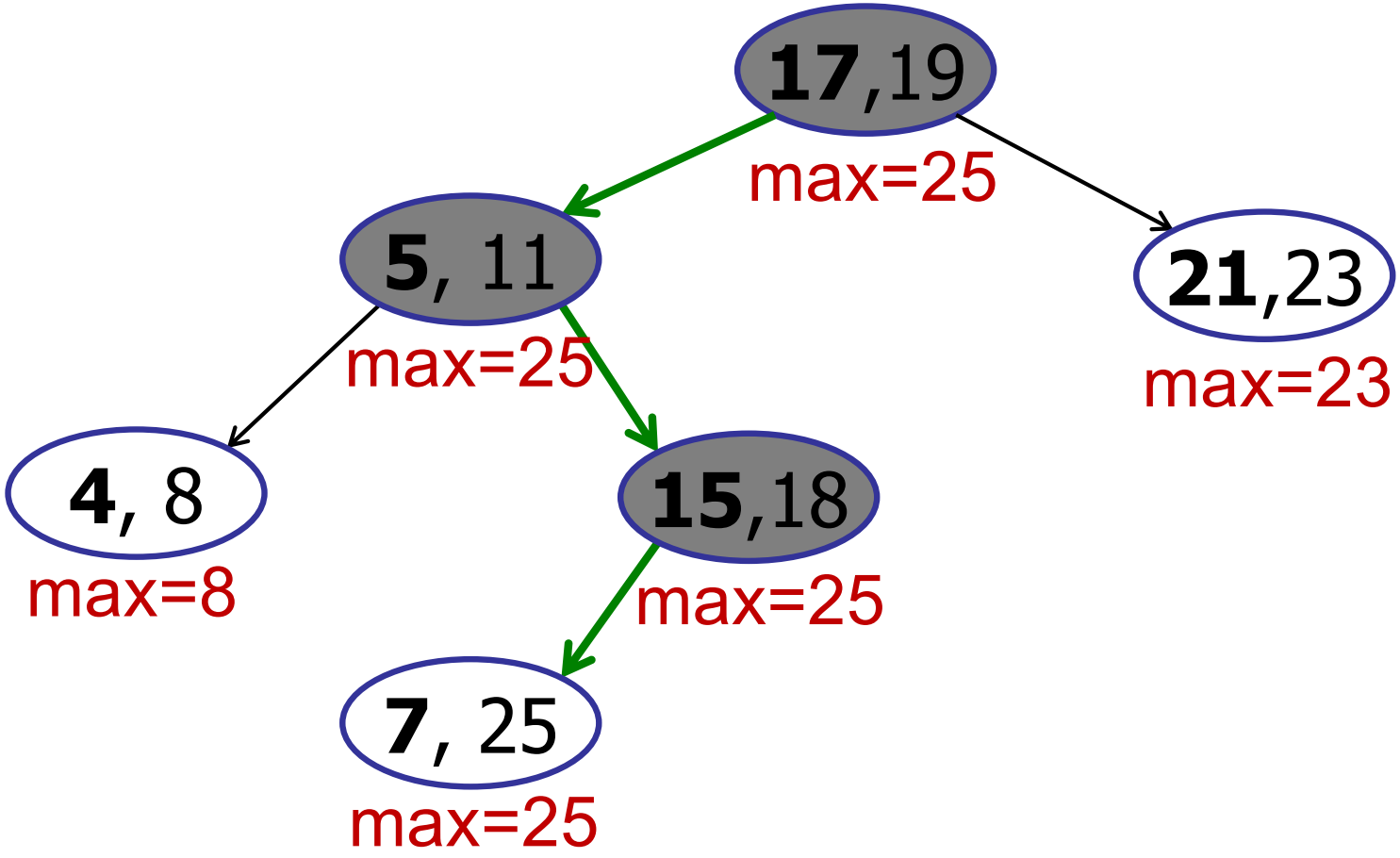
Interval Trees

Searching: interval-search(22)



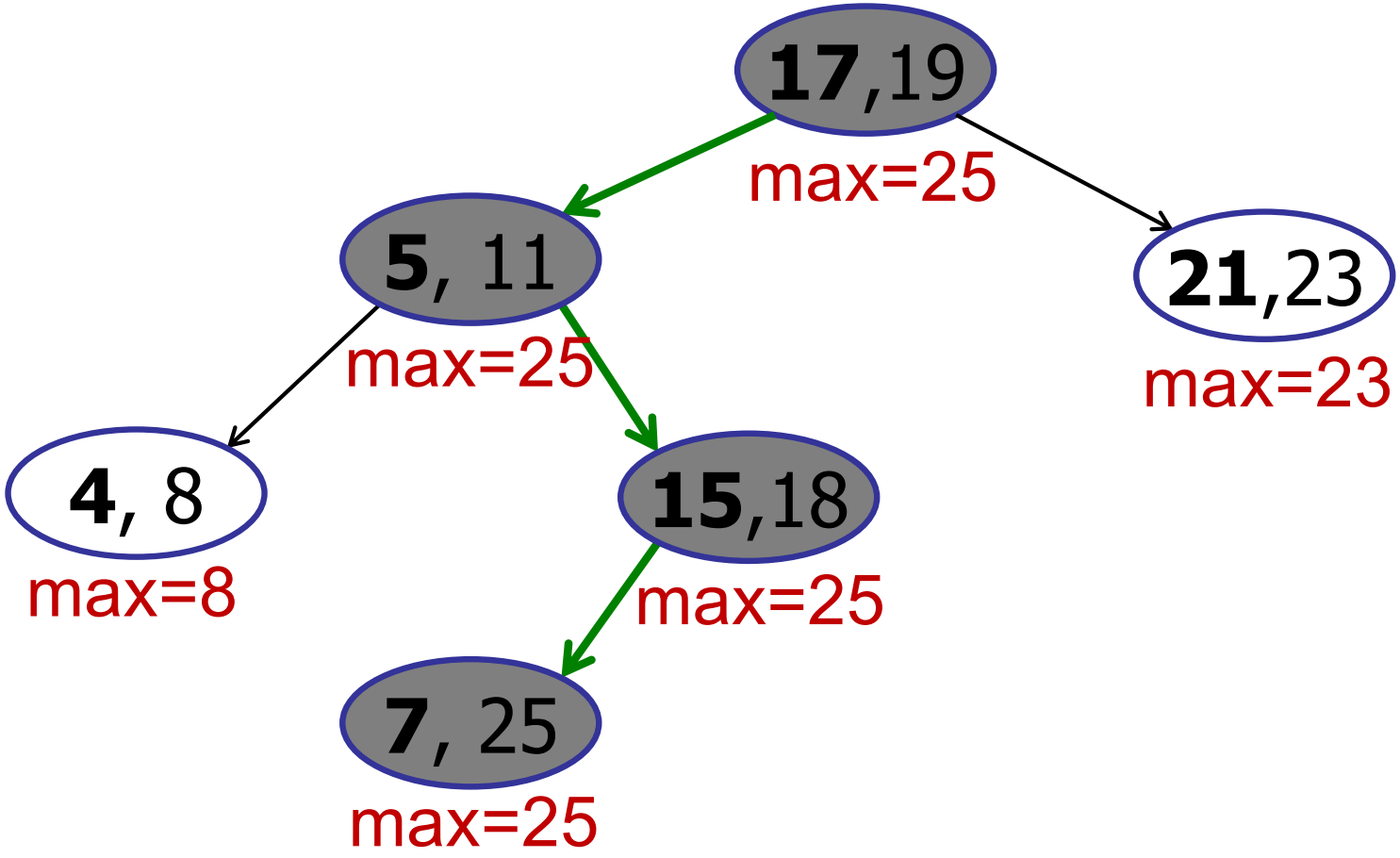
Interval Trees

Searching: interval-search(22)



Interval Trees

Searching: *interval-search(22)*



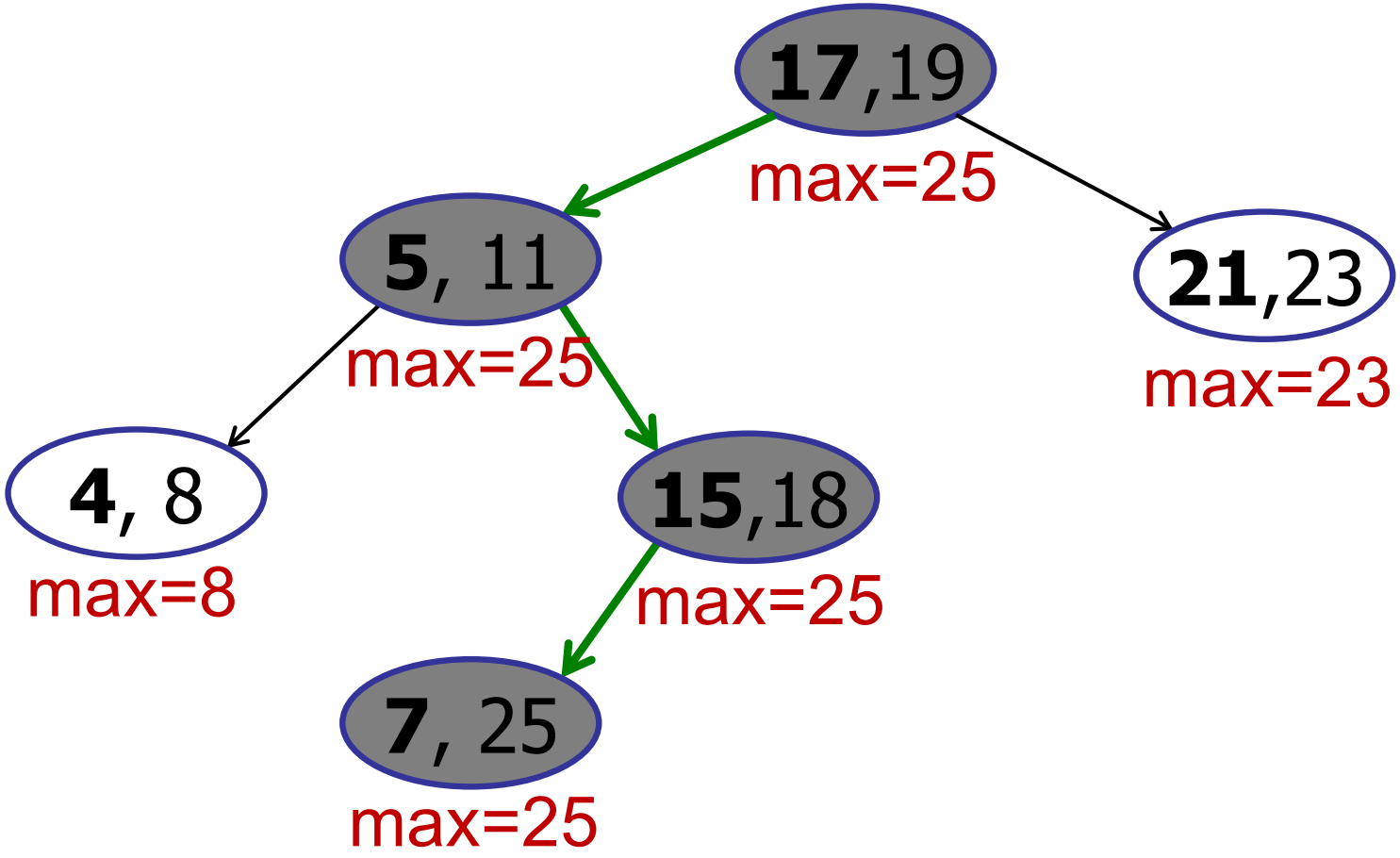
Interval Trees

interval-search(x) : find interval containing x

```
interval-search(x)
        c = root;
        while (c != null and x is not in c.interval) do
                if (c.left == null) then
                        c = c.right;
                else if  (x > c.left.max)  then
                        c = c.right;
                else c = c.left;
        return c.interval;
```

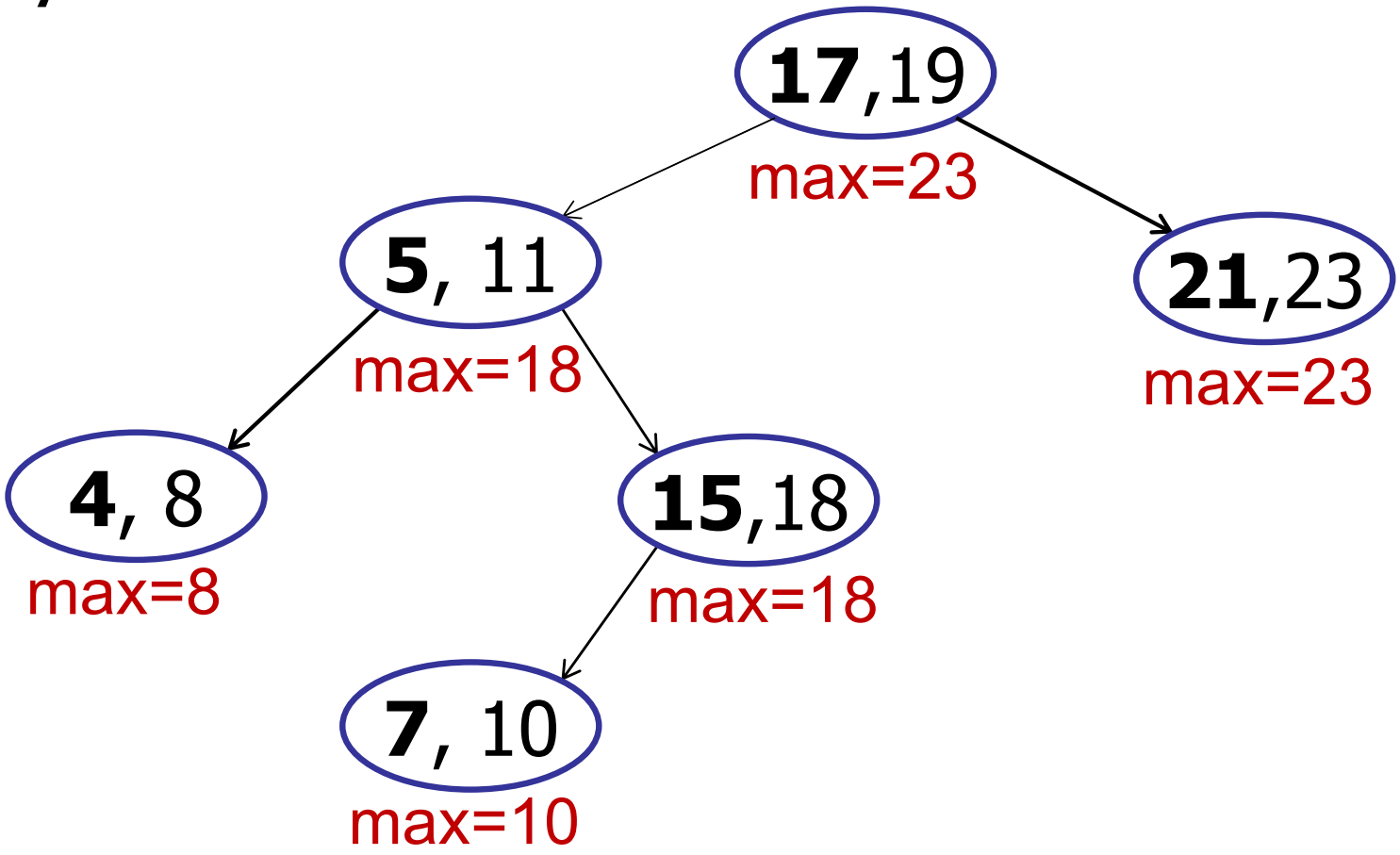
Interval Trees

Will any search find (21, 23)?



Interval Trees

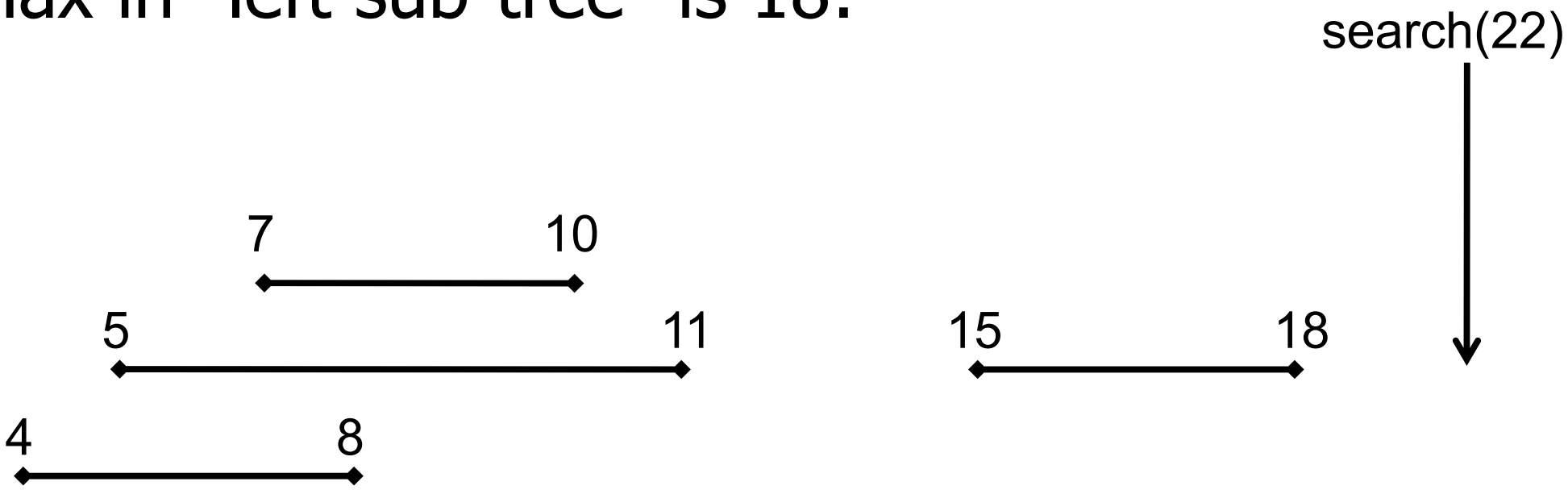
Why does it work?



Claim: If search goes right, then no overlap in left subtree.

Interval Trees

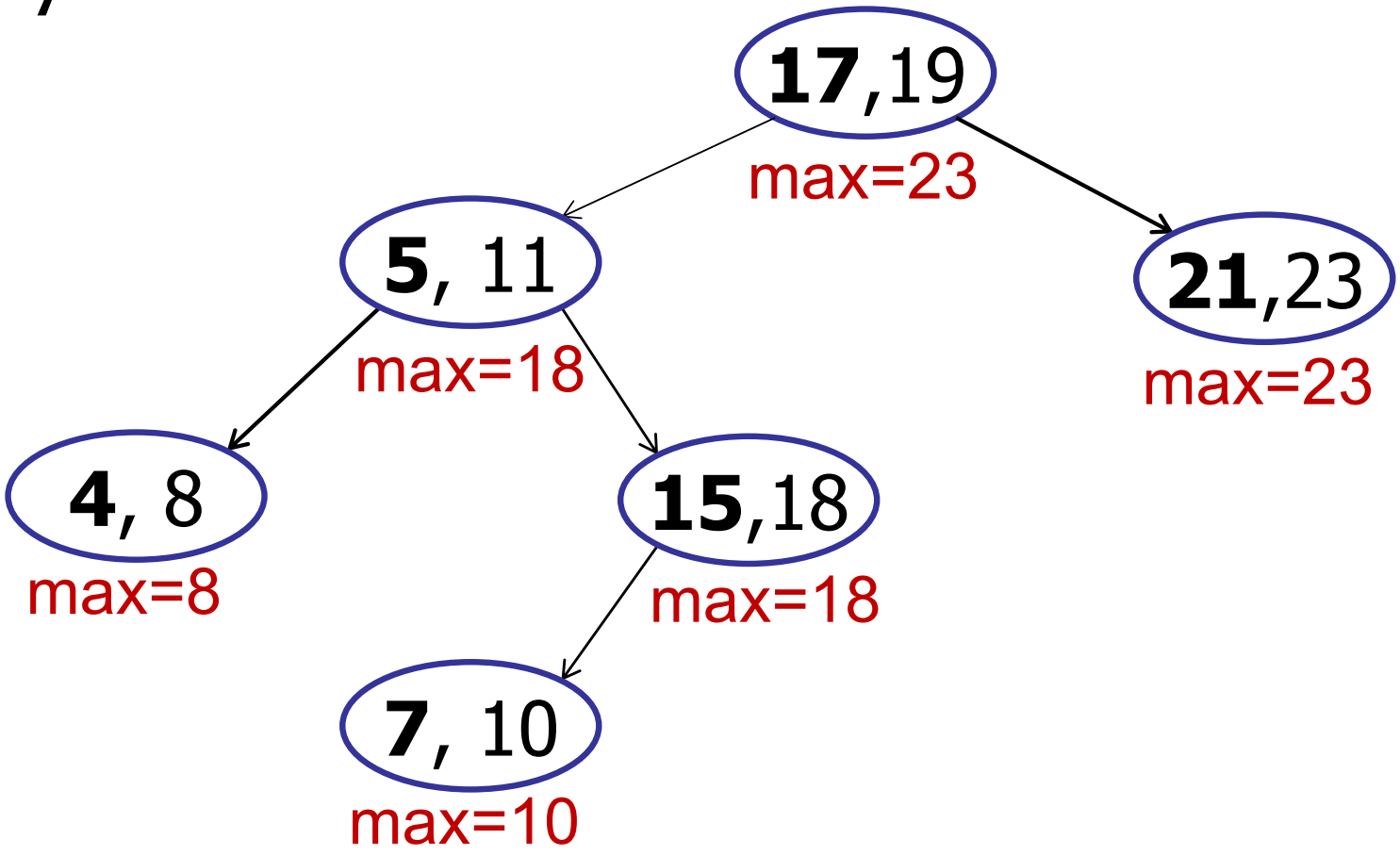
Max in "left sub-tree" is 18:



Safe to go right: 22 is not in the left sub-tree.

Interval Trees

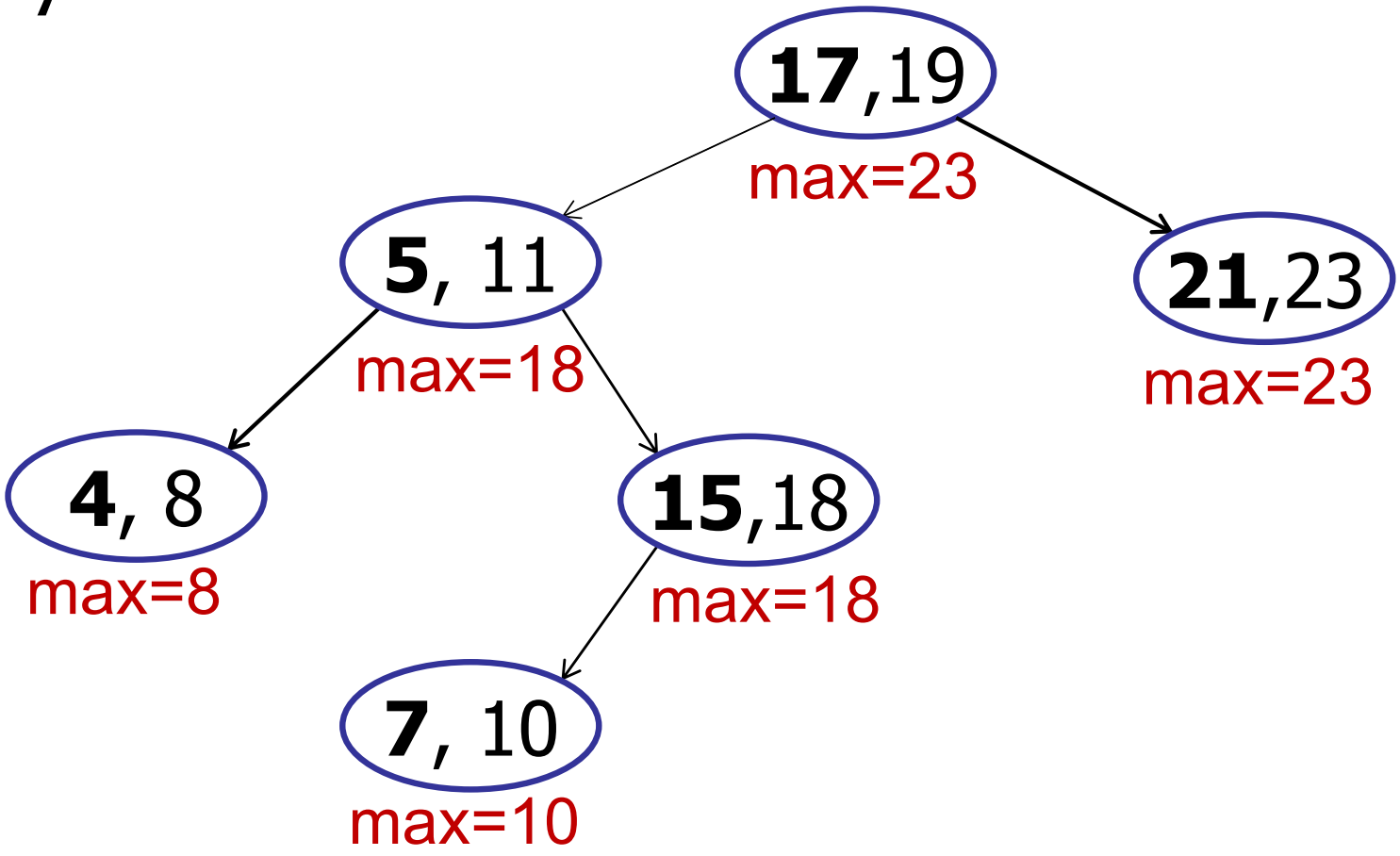
Why does it work?



Claim: If search goes left and there is no overlap in the left subtree…

Interval Trees

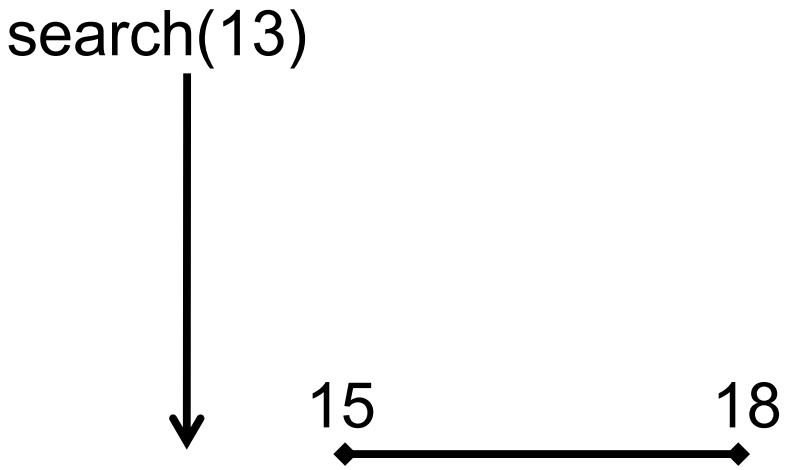
Why does it work?



Claim: If search goes left, then safe to go left.

Interval Trees

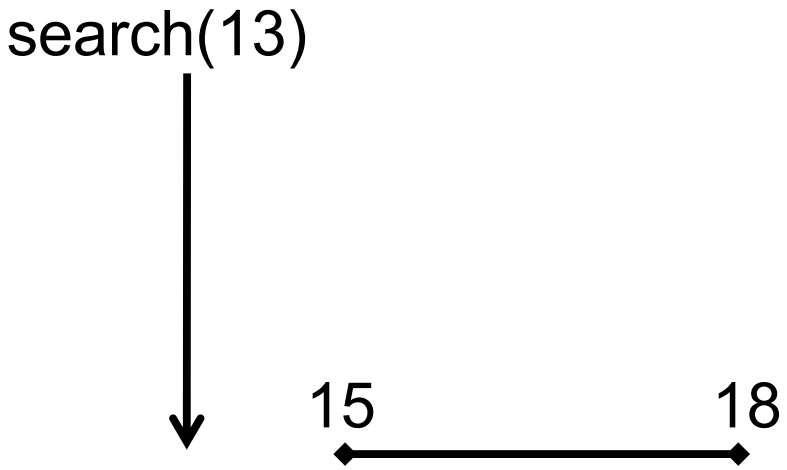
Max in "left sub-tree" is 18:



Go left: search(13) < 18

Interval Trees

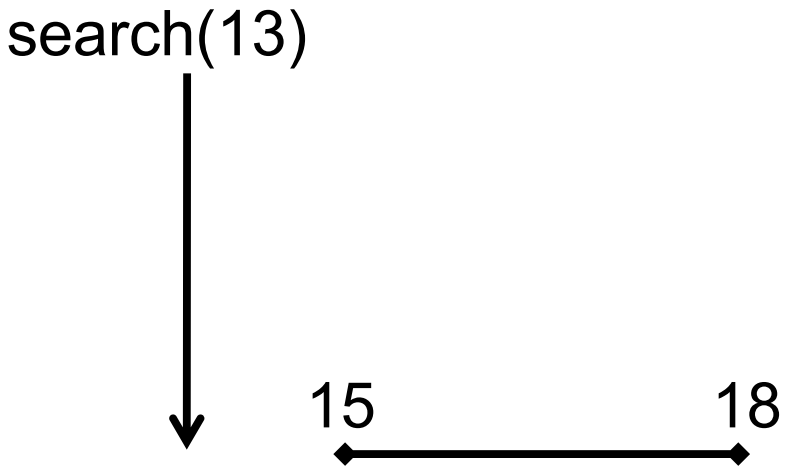
Max in “left sub-tree” is 18:



Go left: search(13) < 15 < 18

Interval Trees

Max in "left sub-tree" is 18:

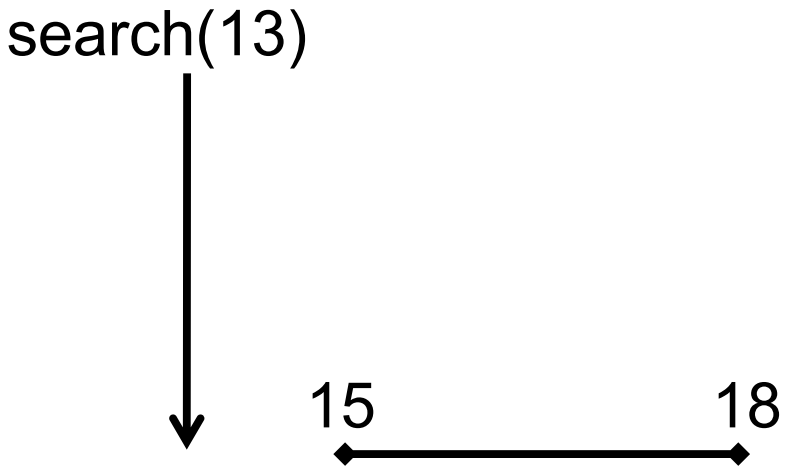


Go left: search(13) < 15 < 18

Tree sorted by left endpoint.

Interval Trees

Max in "left sub-tree" is 18:



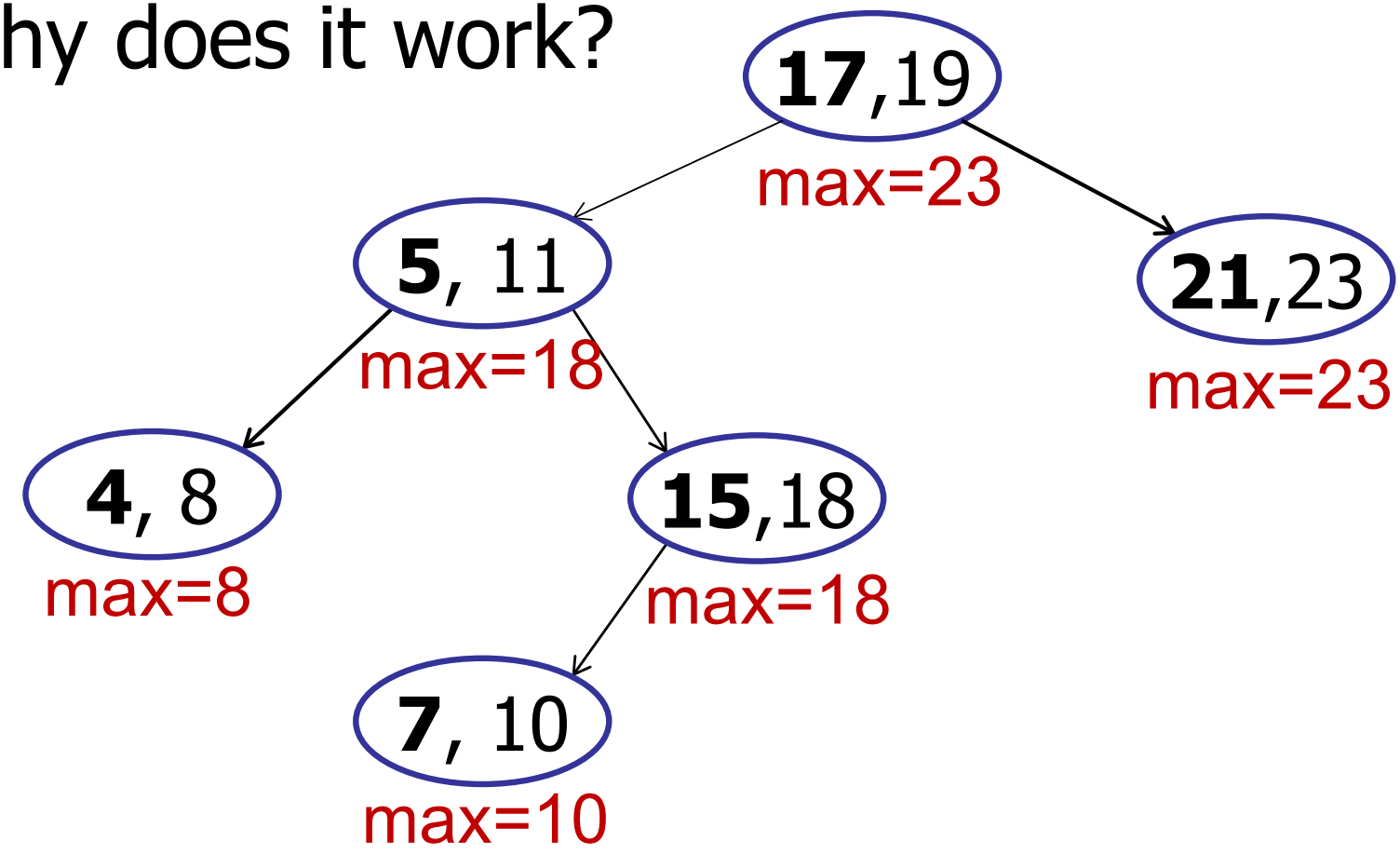
Go left: search(13) < 15 < 18

Tree sorted by left endpoint.

13 < every interval in right subtree

Interval Trees

Why does it work?



Claim: If search goes left and no overlap, then key < every interval in right sub-tree.

Interval Trees

If search goes right: then no overlap in left subtree.

➔ Either search finds key in right subtree or it is not in the tree.

If search goes left: if there is no overlap in left subtree, then there is no overlap in right subtree either.

➔ Either search finds key in left subtree or it is not in the tree.

Conclusion: search finds an overlapping interval, if it exists.

The running time of interval-search is:

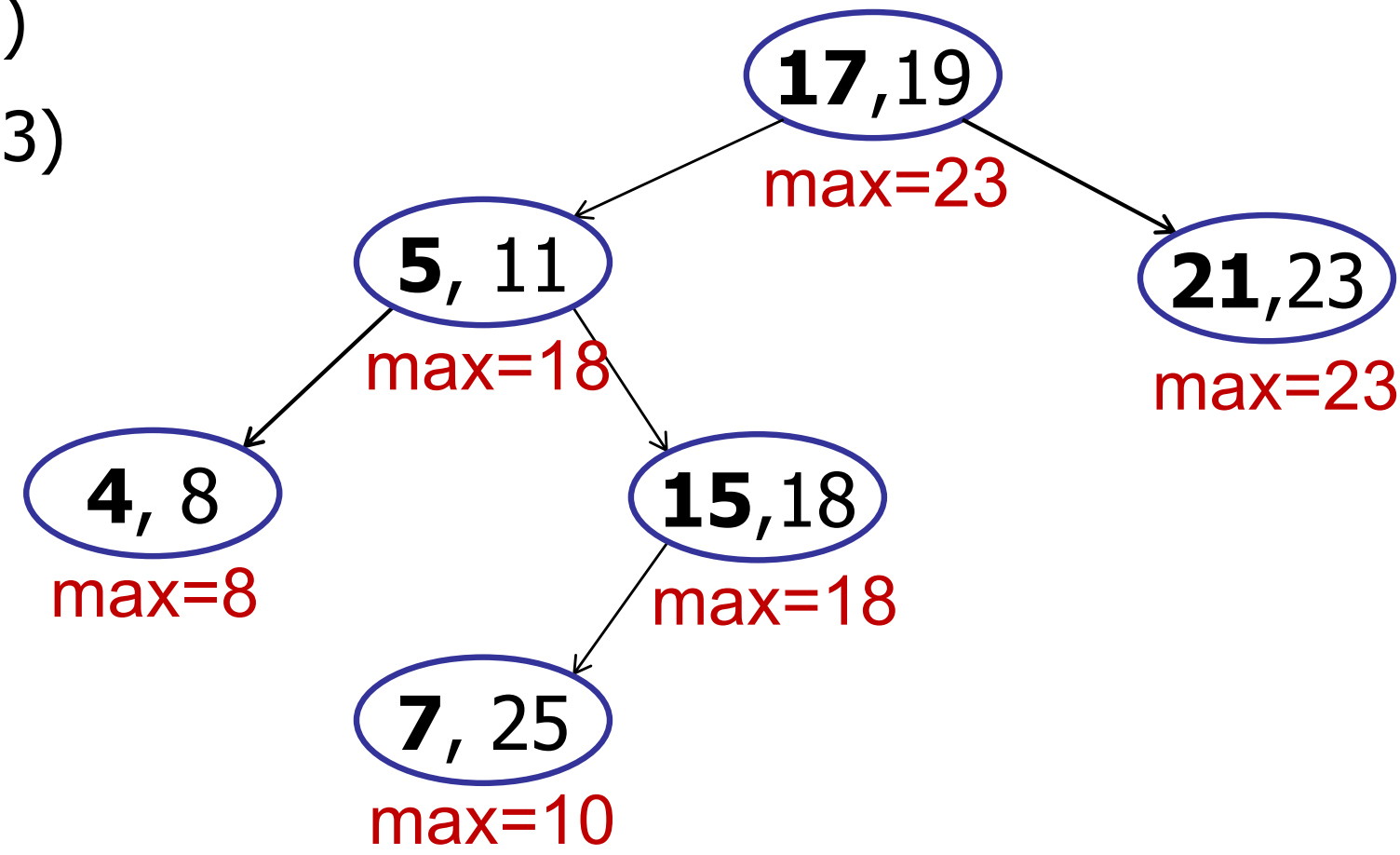
1. O(1)
2. O(log n)
3. O(n)
4. O(n log n)
5. $O(n^2)$
6. Can't say.

Interval Trees

Extension: List all intervals that overlap with point?

E.g.: search(22) returns:

- (7,25)
- (21,23)



Interval Trees

Extension: List all intervals that overlap with point?

All-Overlaps Algorithm:

Repeat until no more intervals:

- Search for interval.
- Add to list.
- Delete interval.

Repeat for all intervals on list:

- Add interval back to tree.

The running time of All-Overlaps, if there are k overlapping intervals?

1. O(1)
2. O(k)
3. O(k log n)
4. O(k + log n)
5. O(kn)
6. O(kn log n)

Interval Trees

Extension: List all intervals that overlap with point?

All-Overlaps Algorithm: O(k log n)

Repeat until no more intervals:

- Search for interval.
- Add to list.
- Delete interval.

Repeat for all intervals on list:

- Add interval back to tree.

Best known solution: O(k + log n)

Today

Three examples of augmenting BSTs

1. Order Statistics

2. Intervals

3. Orthogonal Range Searching