Árbol de intervalos (IntervalTree)

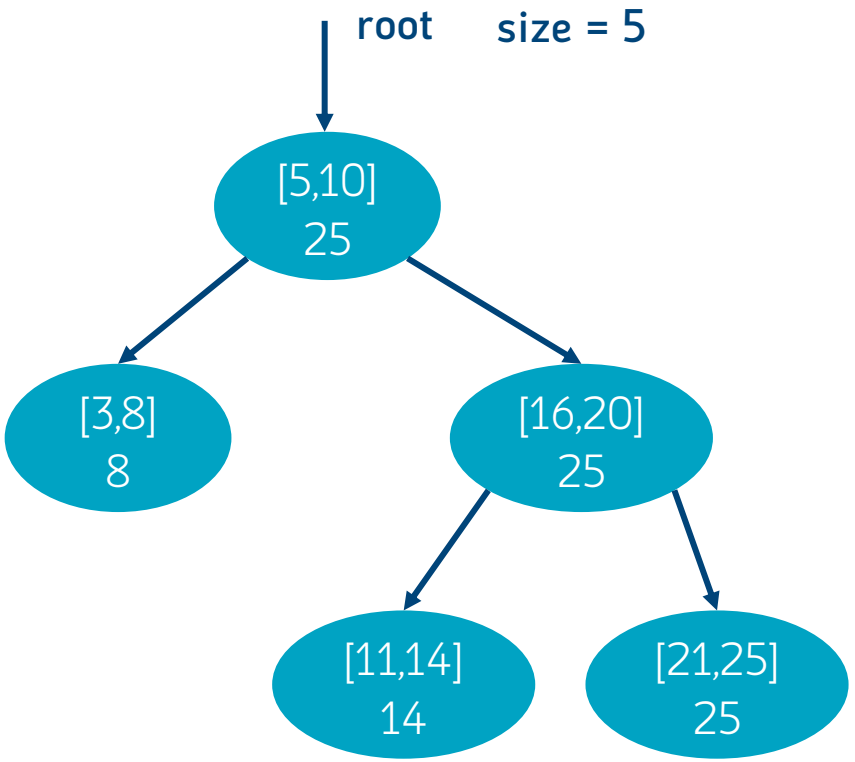
Estructura de Datos, 2º Ing. Informática, Ing. Software, Ing. Computadores

Curso 2022-2023

Universidad de Málaga

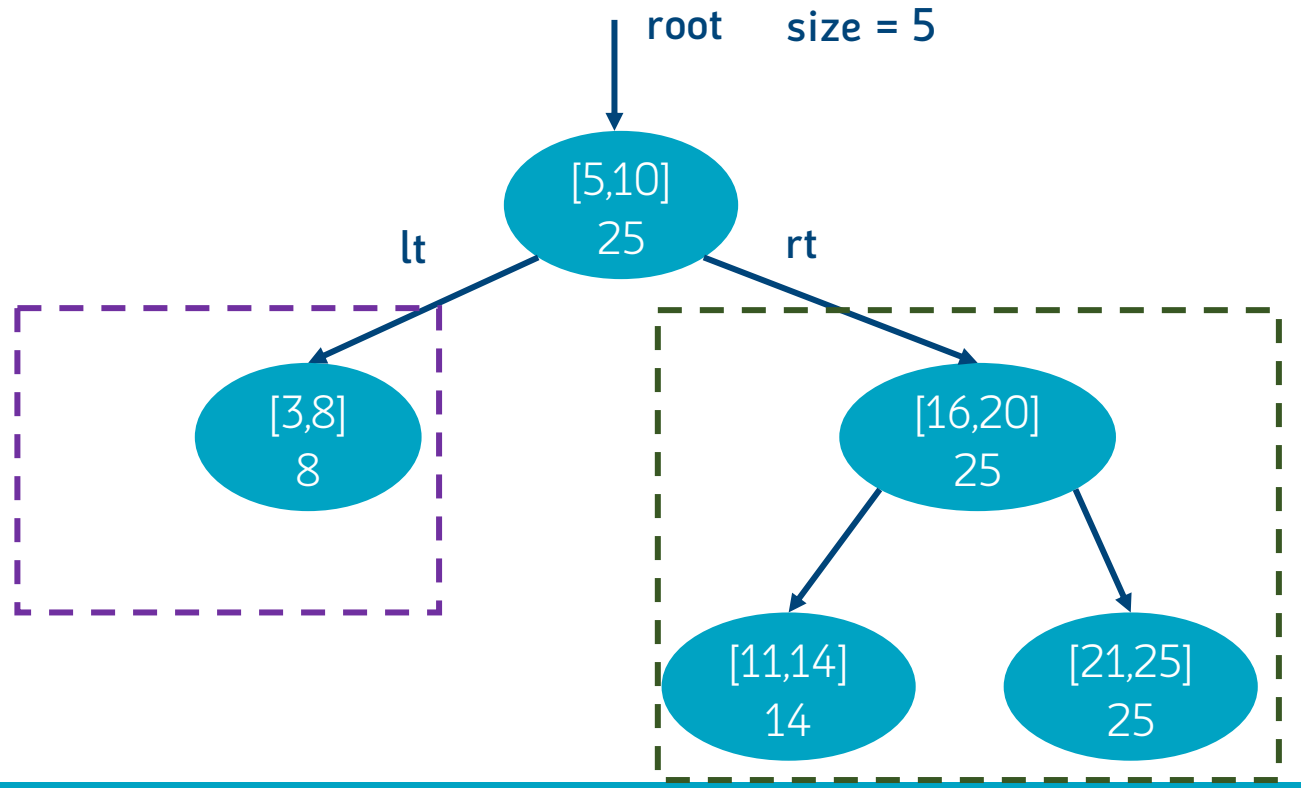
Árbol de intervalos

- El **árbol de intervalos** es un árbol binario de búsqueda (BST) que está adaptado para buscar eficientemente intervalos que se solapan.
  - Utiliza el **límite inferior de los intervalos** como clave para ordenar el BST.



Árbol de intervalos

- El **árbol de intervalos** es un árbol binario de búsqueda (BST) que está adaptado para buscar eficientemente intervalos que se solapan.
  - Propiedad de orden (suponemos que no hay intervalos con el mismo límite inferior)



- Los límites inferiores de todos los intervalos son menores que el de root
- lt es un IntervalTree

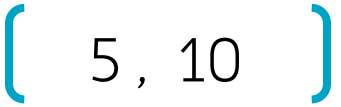
- Los límites inferiores de todos los intervalos son mayores que el de root
- rt es un IntervalTree

Clase Interval

- En Java, la clase **`Interval`** representa intervalos de enteros
  - `Interval` tiene como atributos `low` y `high` que representan respectivamente el límite inferior y superior
  - El método `overlap` devuelve true si el intervalo solapa con otro
  - El método `compareTo` considera únicamente low para determinar el orden de los intervalos

```
package dataStructures.interval;

public class Interval implements Comparable<Interval>{
     int high, low;
     public boolean overlap(Interval y) {/*…*/}
     public int compareTo(Interval o)
}
```



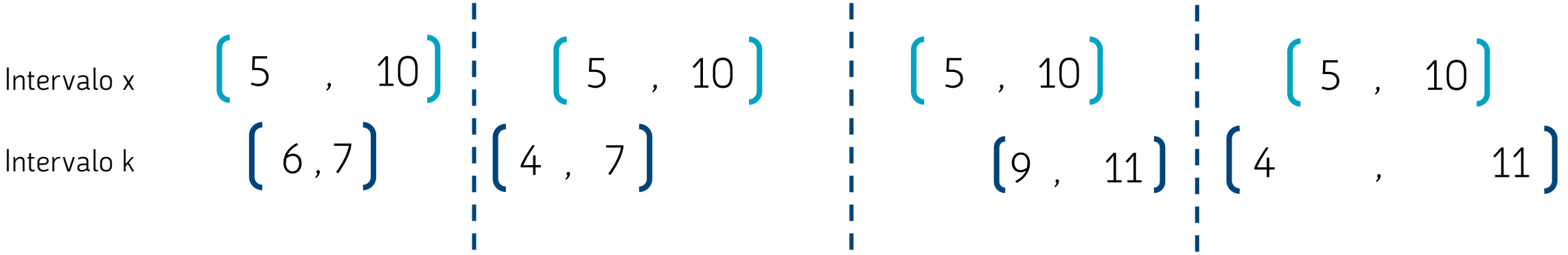
Clase Interval

- Dado dos intervalos x = [x_{low}, x_{high}] y k =[k_{low}, k_{high}], `x.overlap(k)` devuelve true si:

  x_{low} <= k_{high} && x_{high} >= k_{low}

- Dados dos intervalos x y k que solapan, x.union(k) devuelve un intervalo que es la unión de x y k

Ejemplo de intervalos que solapan

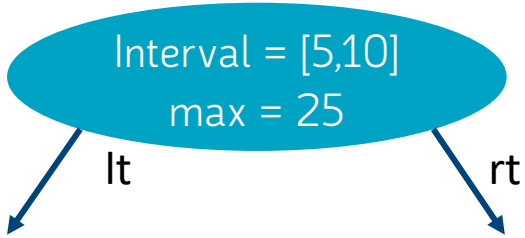


Clase Node

- En Java, la clase **Node** representa los nodos del árbol de intervalos
  - **Node** tiene como atributos el intervalo de enteros (`interval`) que representa y un valor (`max`) que es límite superior más alto de todos los intervalos que hay en el árbol que parte de este nodo. Además, `lt` y `rt` que son referencias a los hijos izquierdo y derecho del nodo

```
private static class Node{
    Interval interval;
    int max;
    Node lt, rt;

    /* constructores*/
}
```



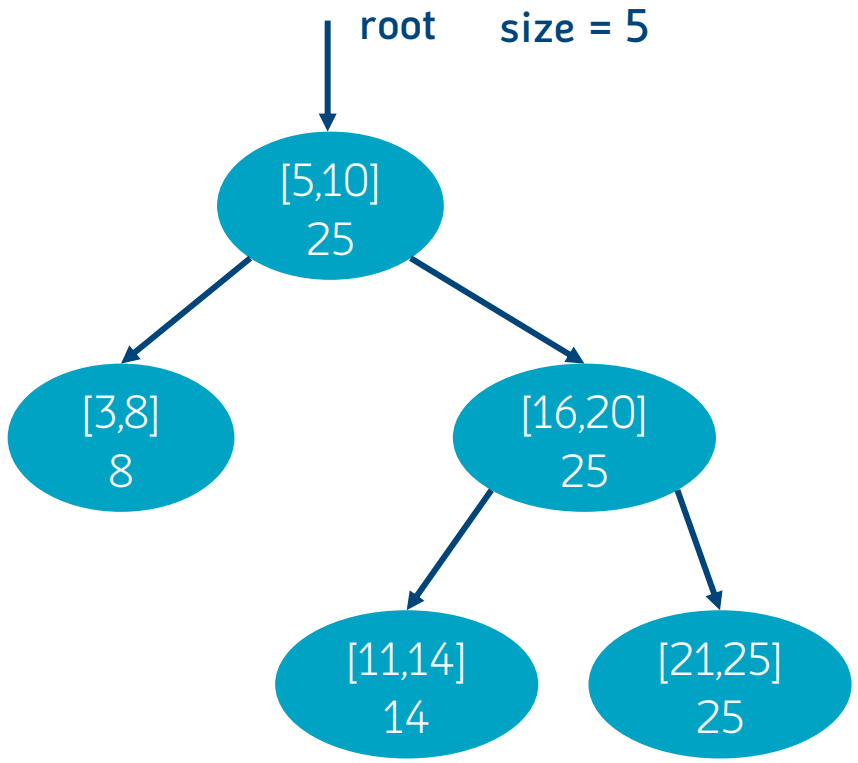
Nota: esta clase está anidada en la clase IntervalTree

Clase IntervalTree

- En Java, la clase `IntervalTree` representa un árbol de intervalos.
  - `IntervalTree` tiene como atributos una referencia al nodo raíz (`root`) y el número de nodos que tiene el árbol (`size`)

```
package dataStructures.interval;

public class IntervalTree {
    private static class Node{
       /* implementación de Node*/
    }
    /* Atributos de IntervalTree */
    Node root;
    int size;
   /* métodos de IntervalTree */
}
```



Ejercicios (I)

Se pide implementar:

- (1.25 pto) Constructor de Node

```
public Node (Interval interval, Node lt, Node rt){ /*…*/}
```

- (0.75 ptos) constructor del árbol de intervalos vacío (no almacena ningún intervalo)

```
public IntervalTree() {/*…*/}
```

- (0.5 ptos) true si el árbol está vacío

```
public boolean isEmpty()
```

- (0.5 ptos) devuelve el número de intervalos almacenados en el árbol de intervalos

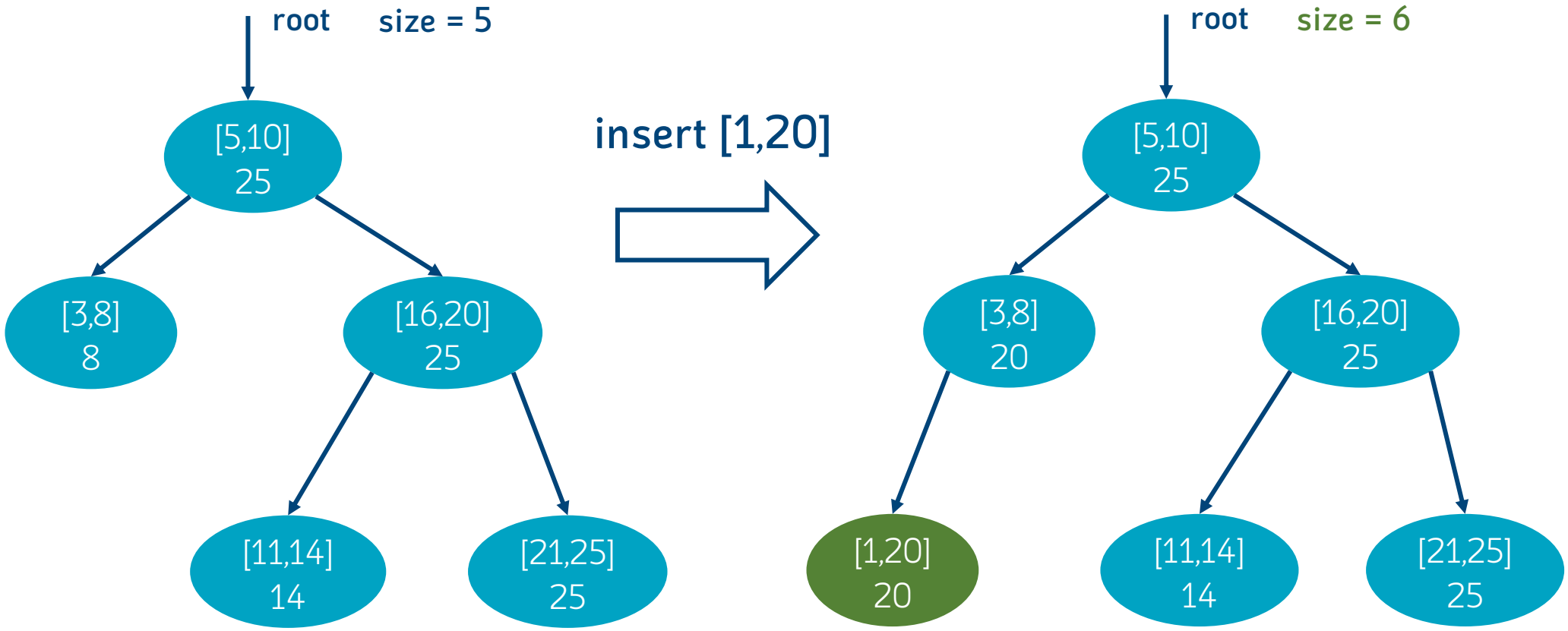
```
public int size()
```

- (2 ptos) inserta el intervalo x en el árbol. El árbol debe mantener la propiedad de orden (páginas 9-10)

```
public void insert(Interval x)
```

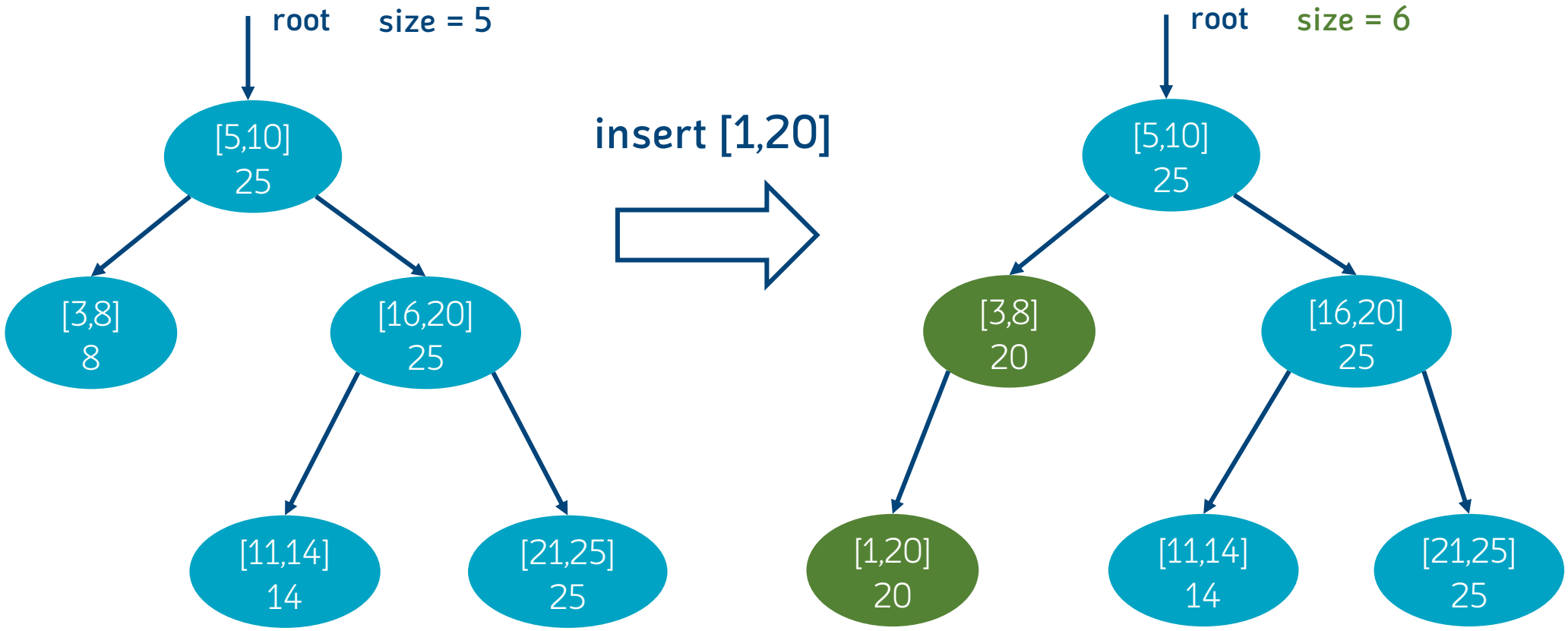
Árbol de intervalos: insert

- Busca la posición que le corresponde al nuevo intervalo utilizando el límite inferior como clave



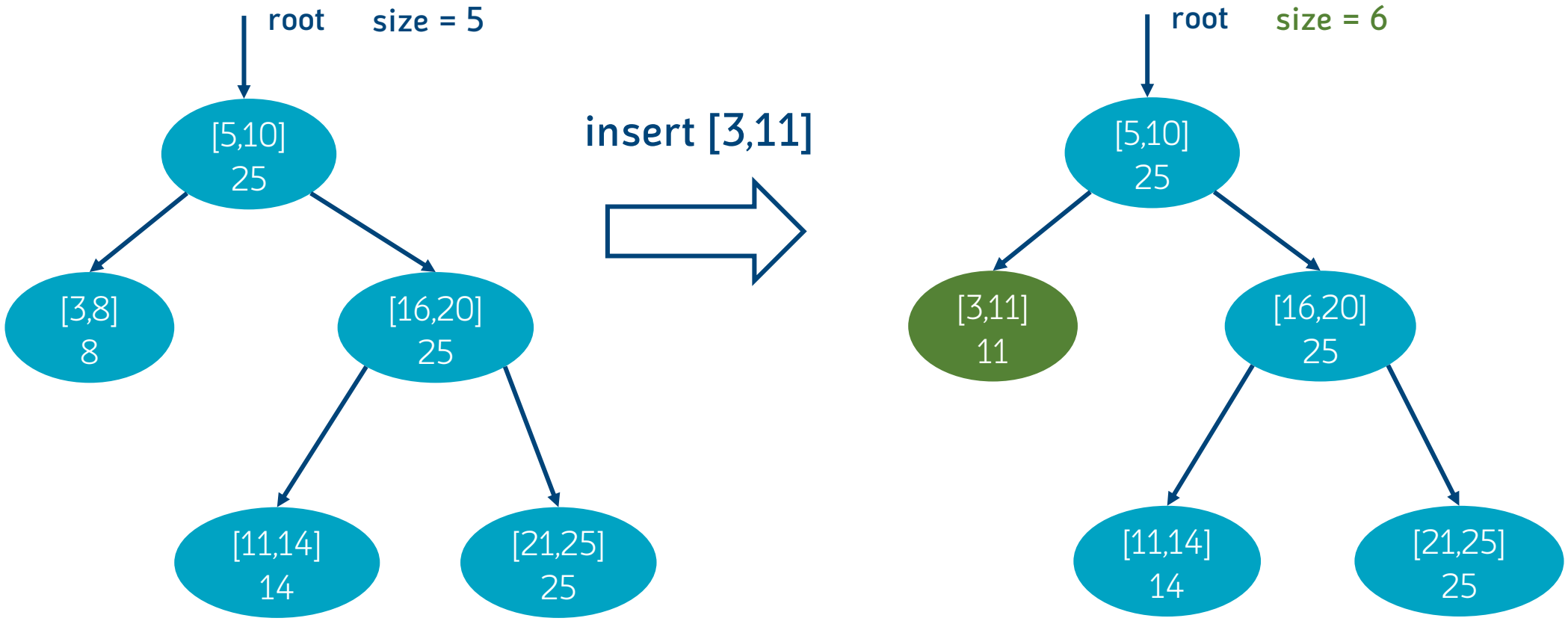
Árbol de intervalos: insert

- Observa que **debe quedar actualizado el atributo *max*** de los nodos que están en el camino de la raíz al nuevo nodo



Árbol de intervalos: insert

- Si existe un intervalo con el mismo límite inferior, actualiza el límite superior y max



Ejercicios (II)

Se pide implementar (continuación):

- (0.5 ptos) true si satisfice el siguiente predicado : el nodo n no es nulo y su atributo max es mayor o igual que el límite inferior del intervalo x

```
private boolean condicionC1(Node n, Interval x)
```

- (0.5 ptos) true si satisfice el siguiente predicado : el hijo derecho de n no es nulo y el límite inferior de n es menor o igual que el límite superior del intervalo x

```
private boolean condicionC2(Node n, Interval x)
```

- (2 ptos) Devuelve un intervalo del árbol que solapa con x (páginas 13 –16)

```
public Interval searchOverlappingInterval(Interval x)
```

- (2 ptos) Devuelve una lista con todos los intervalos del árbol que solapan con x (página 17)

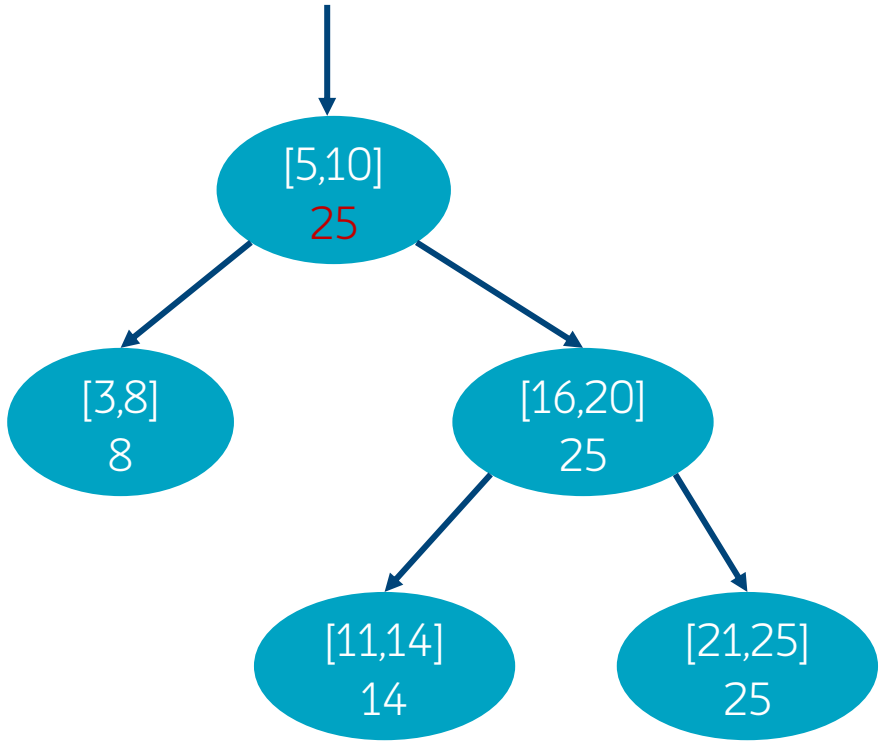
```
public List<Interval> allOverlappingIntervals(Interval x)
```

Árbol de intervalos: searchOverlappingInterval

- Dado un intervalo x = [x_{low}, x_{high}] el objetivo es explorar el árbol para encontrar un intervalo con el que haya solapamiento. Establecemos dos condiciones para la búsqueda en un árbol:
  - **Condición C1: se puede explorar un árbol si no es nulo y si su atributo max es mayor o igual que el límite inferior de x (x_{low})**
  - **Condición C2: se puede explorar el subárbol derecho de un árbol si no es nulo y el valor mínimo de su intervalo es menor o igual que el límite superior de x (x_{high})**
- Si se da la condición C1, el árbol se explora de la siguiente forma:
  - Si hay solape con el intervalo raíz → se devuelve el intervalo raíz
  - En otro caso, si el subárbol izquierdo no es nulo → se explora el subárbol izquierdo
  - Si no hay solape y el subárbol derecho verifica la condición C2 → se explora el subárbol derecho

Árbol de intervalos: searchOverlappingInterval

- **nota**: recuerda que la clase `Interval` proporciona un método `overlap`

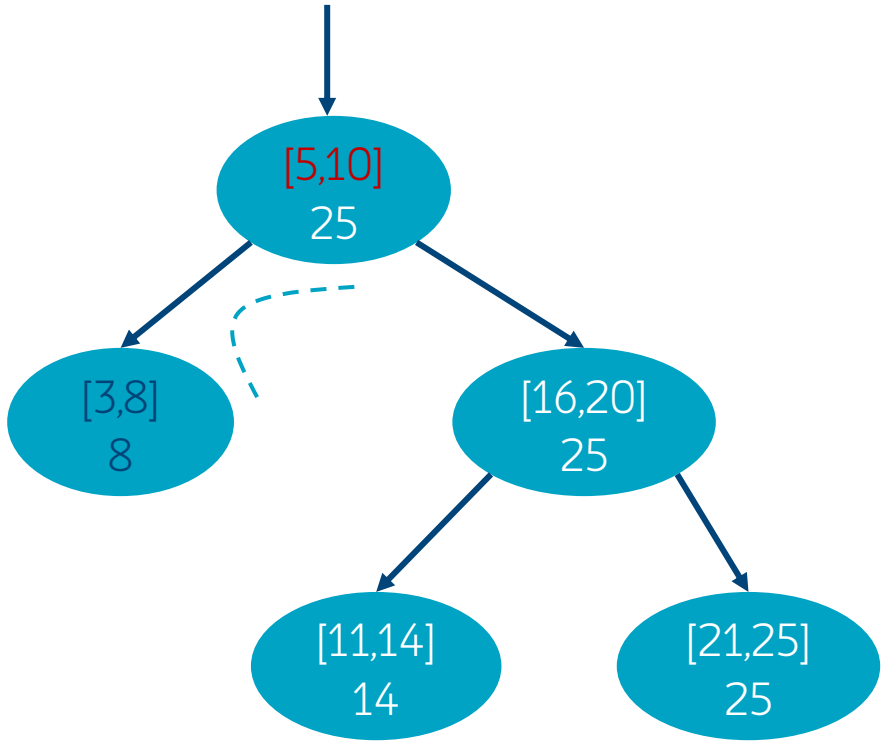


searchOverlappingInterval **x = [26,30]** → null

- No puede haber solape con ningún intervalo (no cumple C1)

Árbol de intervalos: searchOverlappingInterval

- **nota**: recuerda que la clase `Interval` proporciona un método `overlap`

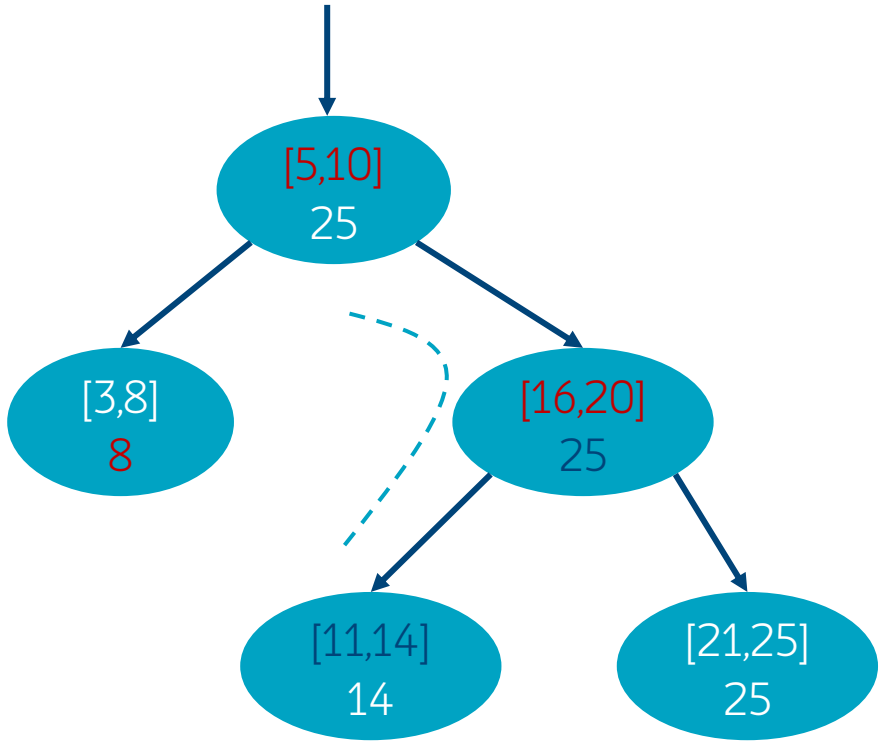


searchOverlappingInterval **x = [2,4]** → [3,8]

- No hay solape con [5,10]
- Se explora el subárbol izquierdo
  - Hay solape con [3,8]

Árbol de intervalos: searchOverlappingInterval

- **nota**: recuerda que la clase `Interval` proporciona un método `overlap`

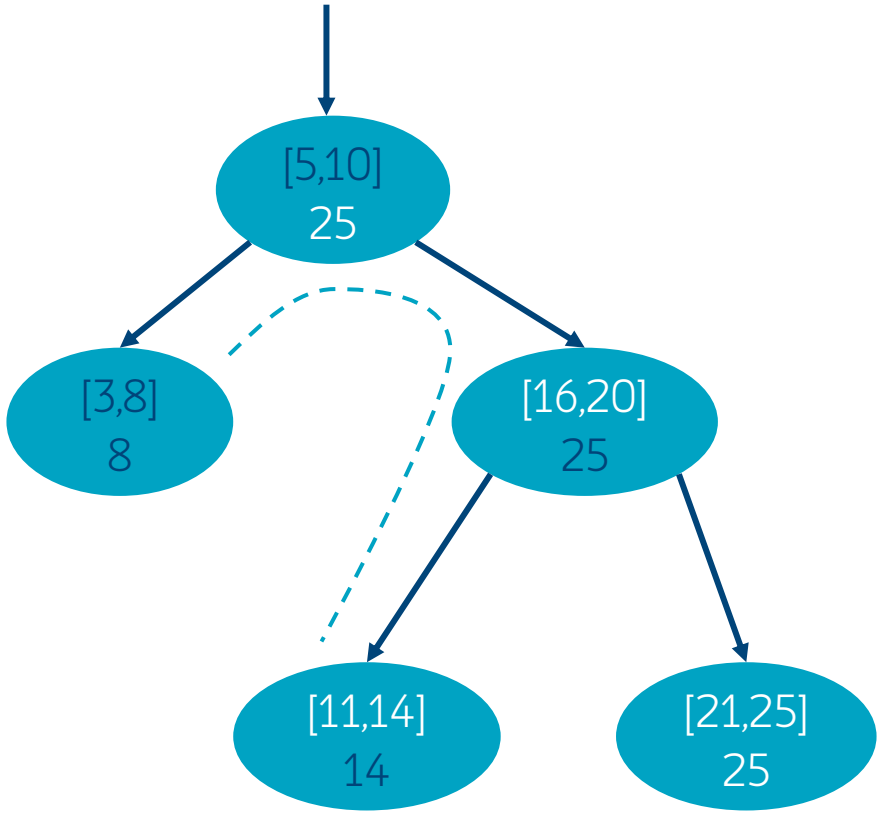


searchOverlappingInterval **x = [11,13]** → [11,14]

- No hay solape con [5,10]
- No hay solape en el subárbol izquierdo
- Puede haber solape en el subárbol derecho (cumple C2)
  - No hay solape con [16,20]
  - Se explora el subárbol izquierdo
    - Hay solape con [11,14]

Árbol de intervalos: searchAllOverlappingInterval

- Búsqueda de **TODOS** los intervalos con solape



searchAllOverlappingInterval x = **[4,7]** → { [5, 10] , [3, 8] }

- Si hay solape con [5,10] → añade al resultado
- Se explora el árbol izquierdo
  → añade todos los solapes del subárbol izquierdo
- Si cumple C2 se explora subárbol derecho
  → añade todos los solapes del subárbol derecho