

TAD

SimpleSplayTree

Estructura de Datos

Grado Informática/Software/Computadores

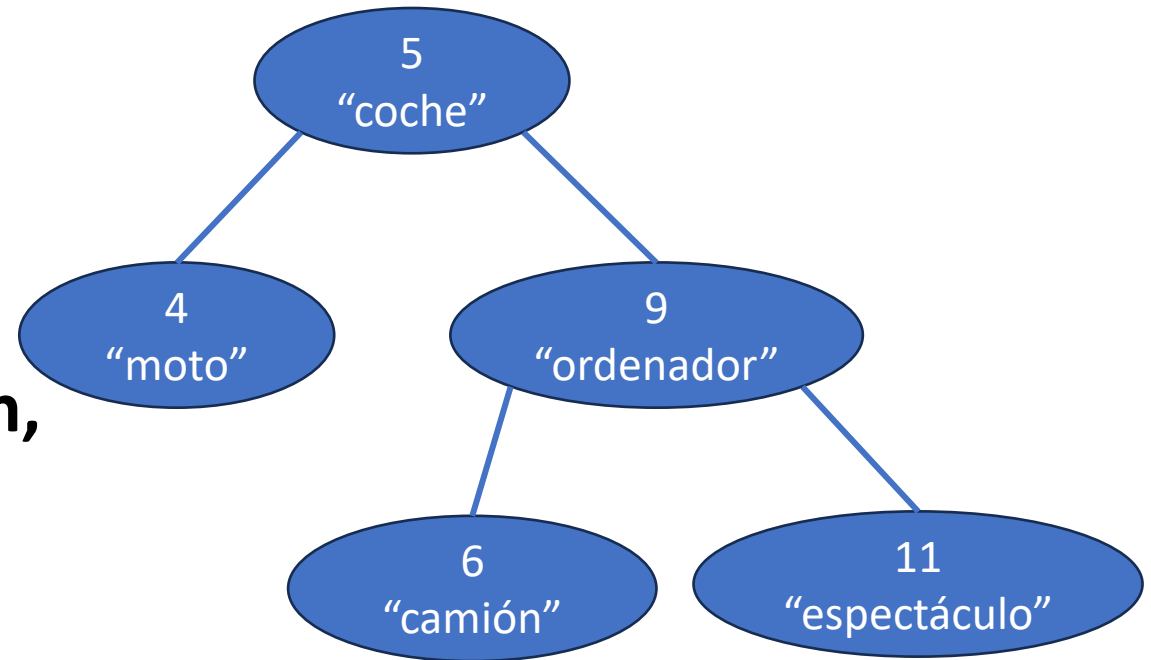
Universidad de Málaga

Especificación informal

- Un SimpleSplayTree (árbol biselado) es un tipo de árbol binario de búsqueda (BST)
- El SimpleSplayTree mejora el rendimiento del BST cuando se insertan o realizan consultas consecutivas de un elemento
 - **Realiza rotaciones (derecha/izquierda) en el árbol para que el último elemento insertado o buscado se convierta en la raíz del árbol**
- El resto de operaciones son iguales que en un BST

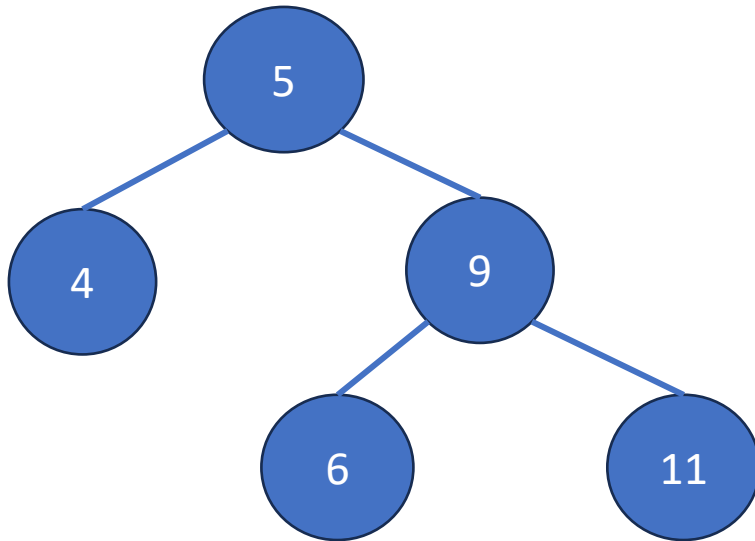
Especificación informal

- Cada nodo del árbol almacena una clave y un valor asociado (p.ej la clave es un entero y el valor un String)
- La propiedad de orden BST solo considera la clave
- **Para simplificar la representación, en las imágenes solo mostramos las claves**

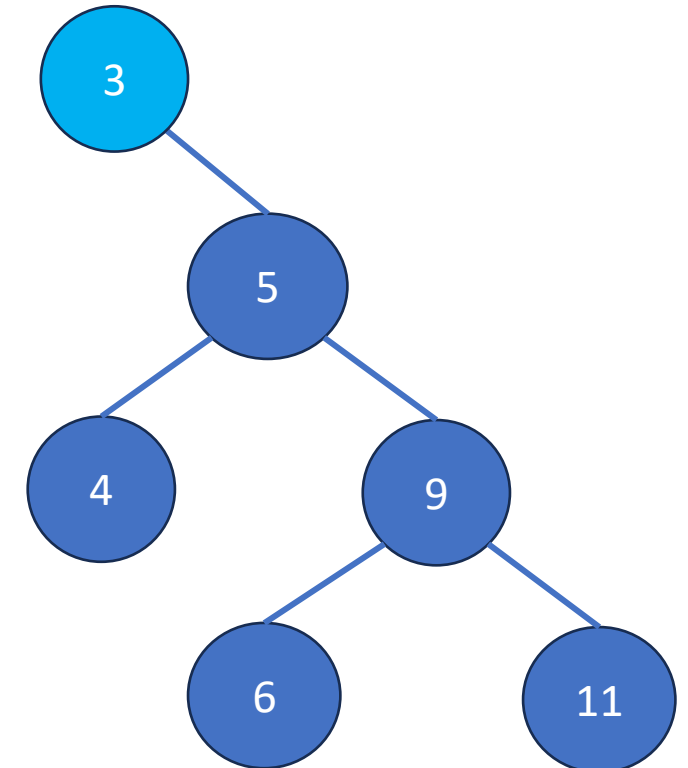
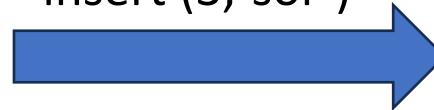


Inserción

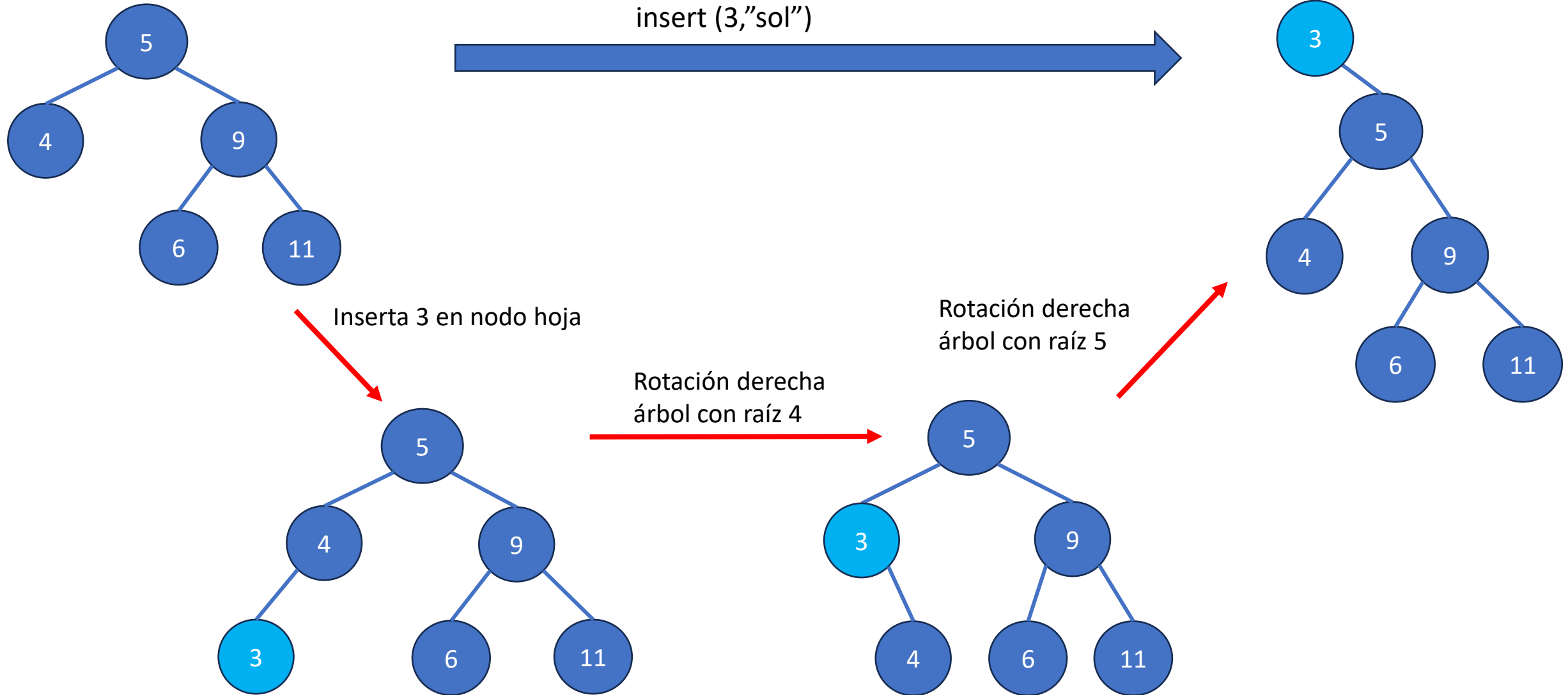
- Se inserta como en un BST → Si la clave no está, se inserta la clave y el valor asociado. En otro caso se actualiza el valor asociado a la clave
- Se sube el elemento a la raíz mediante rotaciones



insert (3,"sol")

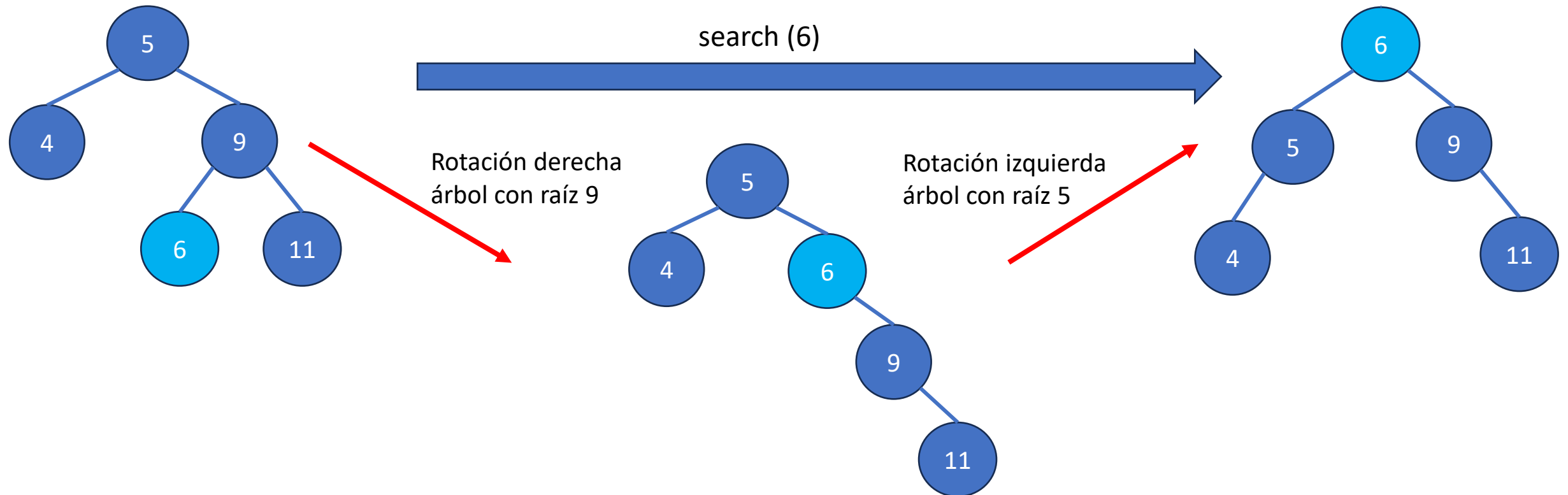


Inserción



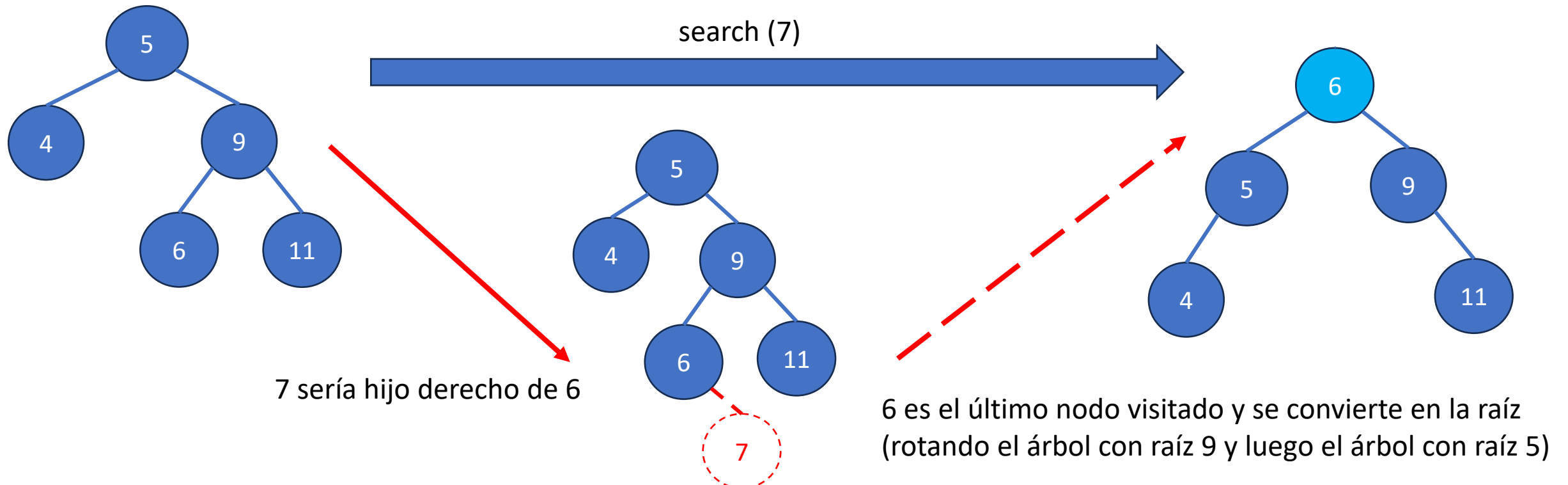
Búsqueda un elemento

- Busca como en un BST → **Si la clave está:**
 - El elemento se convierte en la raíz (mediante rotaciones) y se devuelve el valor asociado



Búsqueda de un elemento

- Busca como en un BST → **Si la clave NO está:**
 - El último elemento visitado en la búsqueda se convierte en la raíz (mediante rotaciones) y se devuelve null



Implementación de SimpleSplayTree

```
public class SimpleSplayTree<K extends Comparable<? super K>, V>
    implements Iterable<Tuple2<K,V>>{

    private static class Node<K, V> { // Clase para representar un nodo
        K key;
        V value;
        Node<K, V> left;
        Node<K, V> right;

        public Node(K k, V v) { /*...*/ }
    }

    private Node<K, V> root; // raiz del arbol
    private int size; // num. elementos en el arbol

    /* ... */
}
```


Especificación de un SimpleSplayTree (I)

Operaciones **públicas** de SimpleSplayTree:

- Constructor: crea un SimpleSplayTree vacío
- **boolean** isEmpty(): true si el árbol está vacío
- **int** size(): devuelve el número de elementos almacenados
- **void** insert(**K** k, **V** v): inserta el par (k,v) si la clave k no está, en otro caso actualiza el valor asociado. El BST resultante tiene en la raíz el nodo con el par (k,v)

Especificación de un SimpleSplayTree (II)

Operaciones **públicas** de SimpleSplayTree:

- **V** search(**K** k): Busca la clave k en el árbol. Si está, devuelve el valor asociado y el BST resultante tiene en la raíz el nodo con clave k. En otro caso, devuelve null y el BST resultante tiene en la raíz el último nodo que se ha visitado durante la búsqueda
- **boolean** isElem(**K** k): true si la clave está en el árbol. Este método modifica el árbol del mismo modo que search
- Iterator<Tuple2<**K**, **V**>> iterator(): devuelve un iterador para recorrer los pares (clave, valor) del árbol en in-orden (implementado)

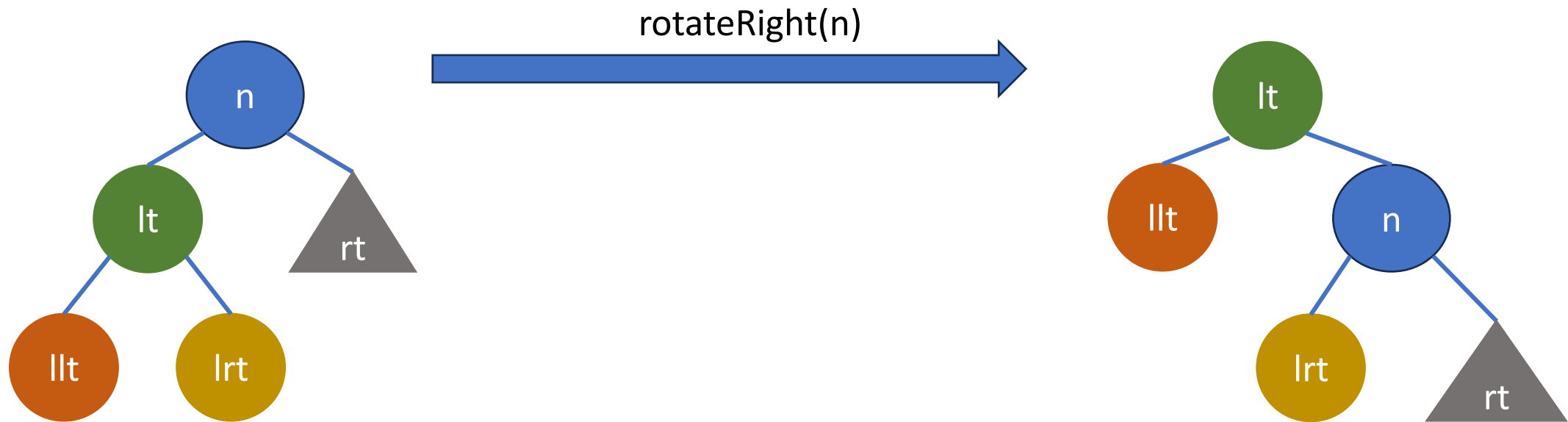
Especificación de un SimpleSplayTree (III)

Operaciones **privadas** de SimpleSplayTree:

- `Node<K,V> rotateLeft(Node<K,V> node)`: Rota el árbol con raíz `node` a la izquierda, el hijo derecho de `node` pasa a ser la raíz. Solo se aplica si `node` y su hijo derecho no son nulos. En otro caso devuelve el árbol original
- `Node<K,V> rotateRigth(Node<K,V> node)`: Rota el árbol con raíz `node` a la derecha, el hijo izquierdo de `node` pasa a ser la raíz. Solo se aplica si `node` y su hijo izquierdo no son nulos. En otro caso devuelve el árbol original
- `List<Tuple2<K,V>> inOrderRec(Node<K,V> node)`: Devuelve una lista con los pares (clave,valor) del árbol en recorrido in-orden

Rotación derecha

- Si el nodo no es nulo y el hijo izquierdo no es nulo
- En otro caso no modifica el árbol



Rotación izquierda

- Si el nodo no es nulo y el hijo derecho no es nulo
- En otro caso no modifica el árbol

