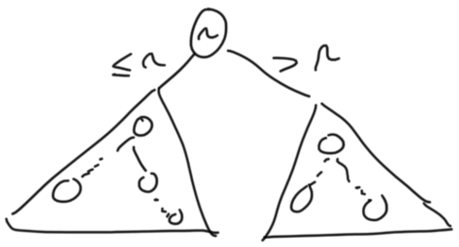
BST são árvores ordenadas:



Pesquisa, inserção e remoção, caso a árvore BST esteja balanceada, são $O(\log_2 N)$

↳ porque, quando vai para a subárvore esquerda, por exemplo, elimina metade do espaço de pesquisa.

Nem todas as árvores são BST.
A árvore BST apresentadas não fazem balanceamento

é necessário fazer código extra

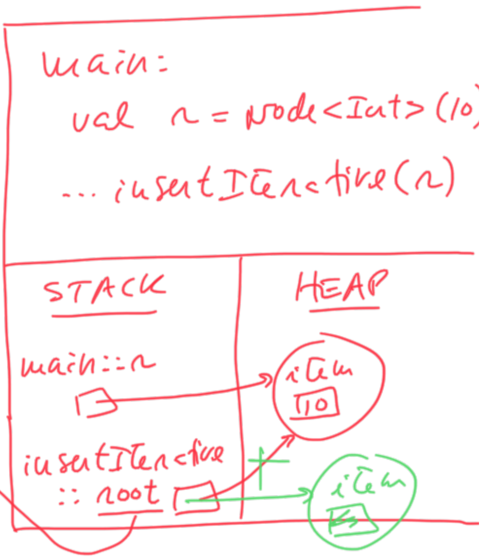
Exemplos de árvores BST balanceadas:

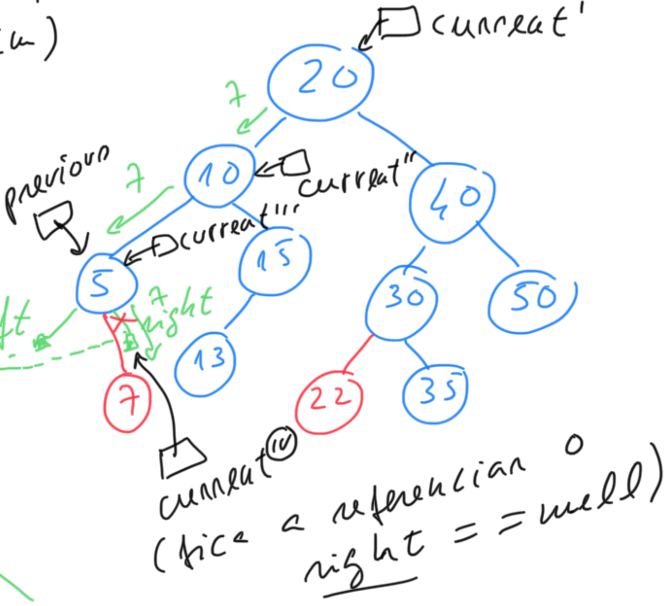
- árvores AVL
- " Red-Black → Kotlin/Java:
  - TreeSet (set)
  - TreeMap (mapa)

BST - inserção de forma iterativa

```
fun <E> insertIterative(root: Node<E>?, e: E,
                    val cmp: Comparator<E>): Node<E>
{
        // root = Node<E>(e)
        val newNode = Node<E>()
        newNode.item = e
        if (root == null)
                return newNode
        var previous: Node<E>? = null
        var current: Node<E>? = root
        while (current != null) {
                val c = cmp.compare(e,    = 7
                          current.item)
                previous = current
                if (c < 0)
                    current = current.left
                else
                    current = current.right
        }
   // → current == null   referências a null
   if (cmp.compare(e,
             previous.item) < 0)
        previous.left = newNode
   else
        previous.right = newNode

   return root
}
```


main:
val r = Node<Int>(10
... insertIterative(r)
STACK
HEAP
main::r
insertIterative :: root
item
10
item


current'
20
7
10
current''
40
previous
7
5
current'''
15
30
50
left
7 right
13
7
22
35
current⁽ⁱᵛ⁾
(fica a referenciar o right == null)