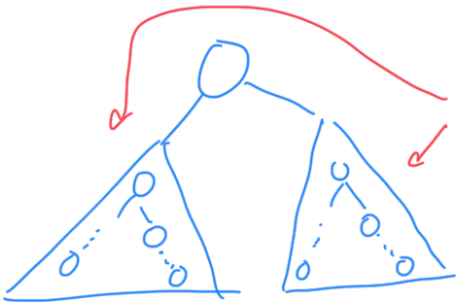
1.

BST pode ou não ter algoritmos de balanceamento

↳ para ter desempenho $O(\log_2 n)$, é necessário que a árvore BST se encontre balanceada

↳ desta forma, quando se vai para uma das subárvores, elimina-se metade do espaço de pesquisa



Na prática, usam-se árvores balanceadas

- árvores AVL
- a Red-black trees

{ Java/Kotlin:
- TreeSet (set)
- TreeMap (mapa)
baseados em árvores

⇓

pesquisa, remoção e inserção em $O(\log_2 n)$

Insert iterative

```
fun <E> insert(root: Node<E>, e: E,
               cmp: Comparator<E>): Node<E>
{
    val newNode = Node<E>()
    newNode.item = e
    if (root == null)
        return newNode

    var previous: Node<E>? = null
    var current:    "      = root
    while (current != null) {
        val c = cmp.compare(e, current.item)
                            "       "
                            7       20, 10, 5
        previous = current
        if (c < 0)
            current = current.left
        else
            current = current.right

    }
  → // current == null
if (cmp.compare(e, previous.item)
                              null <- - left
    previous.left = newNode
else previous.right = newNode
return root
}
```