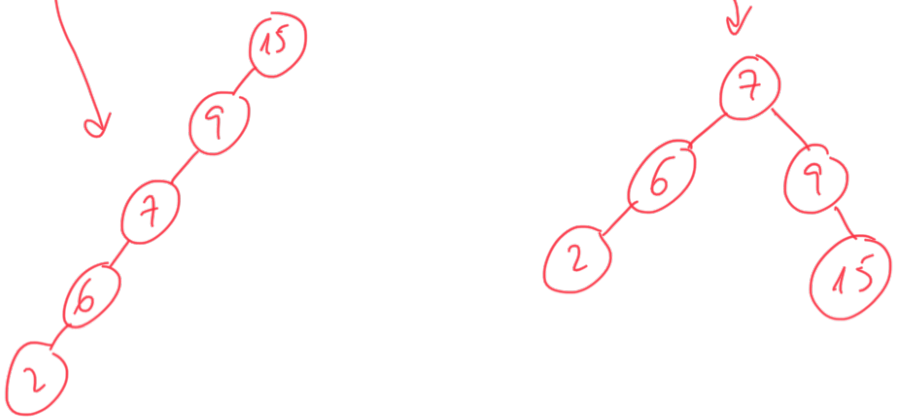
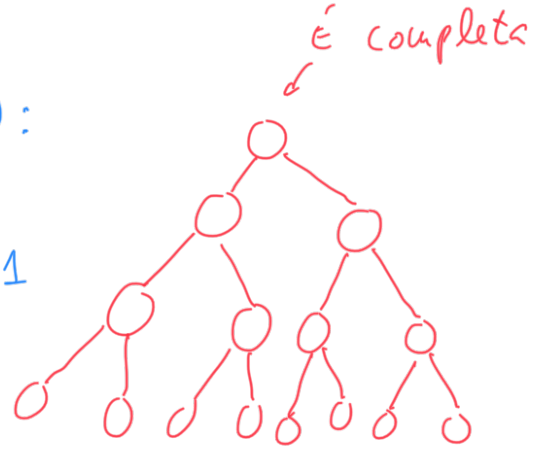
Cálculo do menor elemento duma BST


```
fun min(h: Node<Int>): Int? {
    if (h == null)
        return null
    while (h.left != null) {
        h = h.left
    }
    // h.left == null
    return h.item // retorna valor da raiz
}
```

Custo temporal é $O(\log_2 n) = O(\text{altura})$ se estiver balanceada. Se a BST degenerar numa lista (caso extremo), custo é $O(n)$

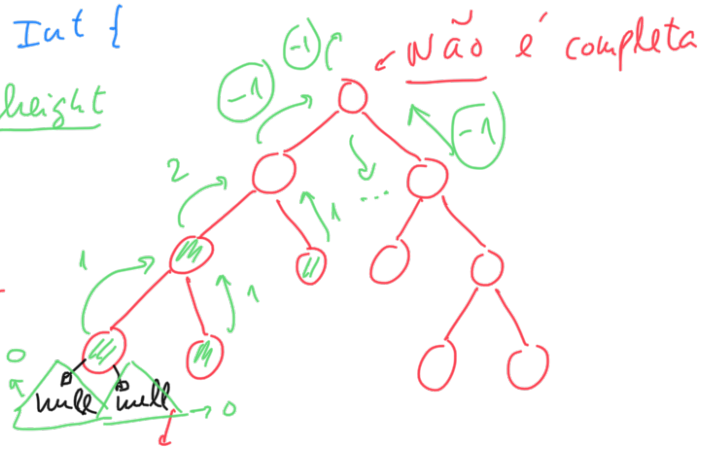


EX: Ver se uma árvore (BST ou não) é completa (todos os níveis encontram-se preenchidos) (2.)




```
fun <E> isComplete(root: Node<E>?):
    Boolean {
    return isCompleteAux(root) != -1
}

fun <E> isCompleteAux(root:
        Node<E>?): Int {
    // variante da função height
    if (root == null)
        return ~~2 0, -1 ?~~ 0
    val lh = isCompleteAux(
        root.left)
    val rh = isCompleteAux(
        root.right)
    if (lh == rh)
        return lh + 1
    // lh != rh
    if (lh == -1 || rh == -1)
        return -1
    return -1
}
```



não dá para ver se apenas tem os 2 filhos ou se são nós folha.

-1 sinaliza que a subárvore não é completa

→ continua a sinalizar, para cima

return -1 → sinaliza que não é completa