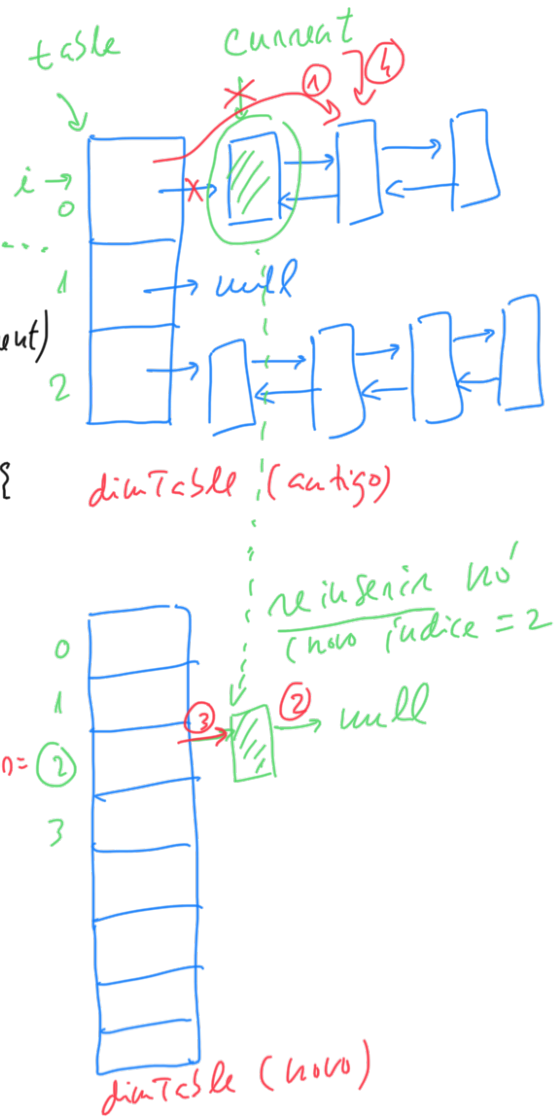
```
private fun resize() {
    // Aumentar a dimensão da tabela e recalcular a
    // posição dos elementos na nova tabela
// dimTable = 2 * dimTable   → Problema: dim. não é um nº primo
    dimTable = getSizeMersenne(x, delta)
                ↓ devolve próximo primo de Mersenne → ver livro

    val newTable = arrayOfNulls<
            Node<Any??>(dimTable)
        as Array<Node<E>??>

    for(i in table!!.indices) {
        var current = table!![i]
        while(current != null) {
            ① table!![i] = table!![i].next
            val newPos = index(current.element)
            ② current.next = newTable[newPos]
            if(newTable[newPos] != null) {
                newPos[newPos]!!.previous =
                                    current
            }
            ③ newTable[newPos] = current
            ④ current = table!![i]
        } // while
    } // for
    table = newTable
}
```



```
class HashSet<E> : MutableSet<E> {
...

override fun iterator<E> {
    return MyIterator()
}

private inner class MyIterator: Iterator<E>{
    var currentPos: Int = -1
    var nodeIt: Node<E>? = null
    var currentElement : Node<E>? = null

    override fun hasNext() : Boolean {
        if (currentElement != null)
            return true
        // senão, efeta currentElem com próximo elemento válido
        while (currentPos < table!!.size) {
            if (nodeIt == null) {
                currentPos ++
                if (currentPos < table!!.size)
                    nodeIt = table!![currentPos]
            }
            else {
                currentElement = nodeIt
                nodeIt = nodeIt!!.next
                return true
            }
        }
        return false
    }
}
```

override fun iterator() → cria instância que tem acesso ao this da outer class

inner class MyIterator → cria instância que tem acesso ao this da outer class

currentPos → índice do bucket atual

nodeIt → nó a iterar

currentElement → nó a iterar

andar de bucket em bucket

Iterar nós do bucket atual

hasNext() → se tem elemento → true

```
...
override fun next(): E {
        if (!hasNext())
            throw NoSuchElementException()
        val aux = currentElement.element
        currentElement = null
        return aux!!
    }
} // HashSet
```