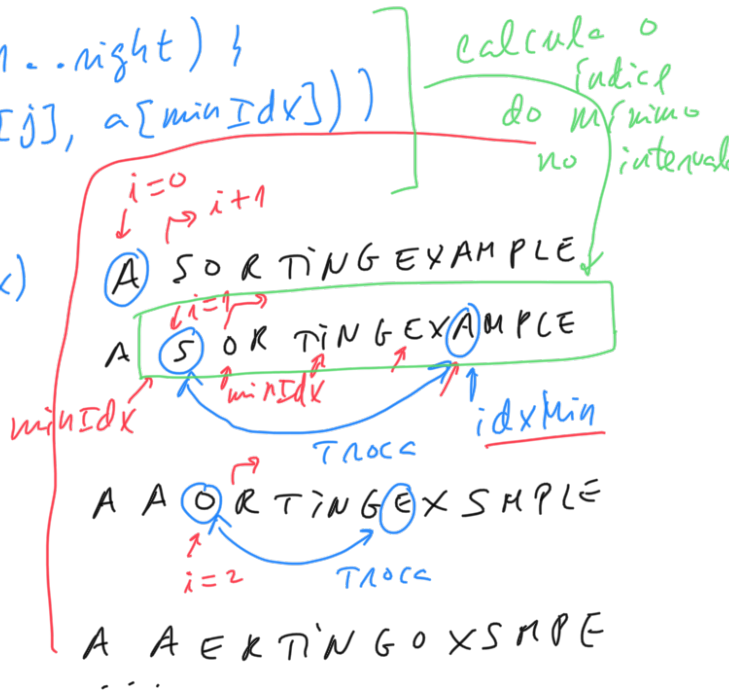
Selection Sort

```
fun selectionSort(a: CharArray, left: Int, right: Int)
{
    for(i in left until right) {
        var minIdx = i
        // Find current minimum from i on
        for(j in i+1..right) {
            if (less(a[j], a[minIdx]))
                minIdx = j
        }
        exch(a, i, minIdx)
    }
}
```

right exclusive



Complexidade Temporal do Selection Sort

n = right - left + 1 (dimensão do array)

cálculo de mínimos em subarrays cada vez menores:

$$
\begin{aligned}
&n \rightarrow 1^{\circ} \text{ mínimo} \\
&+ \\
&n-1 \rightarrow 2^{\circ} \text{ mínimo} \\
&+ \\
&n-2 \rightarrow 3^{\circ} \text{ "} \\
&+ \\
&\vdots \\
&+ \\
&1 = \text{série aritmética} = \delta(n^2)
\end{aligned}
$$

custos do pior caso e melhor caso ↙

array ordenado ↑ array ordenado decrescentemente

Será que Selection Sort é estável?

Input: 4_A 5 3 2 4_B 1

Output:

↓ se não altera a ordenação feita com chave anterior, i.e., se ordenar por nº não altera ordenações por strings.

← Está só a ver a chave numérica, não a chave string

$4_B == 4_A$ em termos de chave numérica

Selection Sort não é estável pois alterou ordem por string: 4B ficou antes de 4A

T.P.C. Fazer versão estável do selection sort (stable selection sort)

↓

tem que fazer procedimento semelhante ao insertion sort

↓

estável

Bubble sort

```
fun bubbleSort(a: CharArray, left: Int, right: Int)
{
    // Put n-1 elements in order
    for (i in left until right) {
        for (j in right downTo i+1) {
            lessExch(a, j, j-1)
        }
    }
}
```

Complexidade Temporal: $O(n^2)$

É possível otimizar caso o array estivesse ordenado?

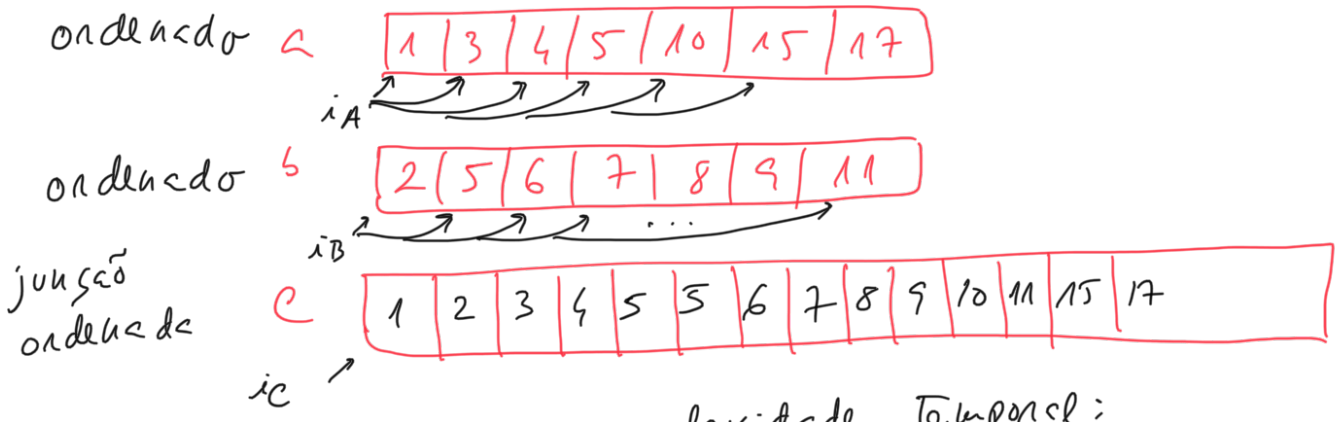
T.P.C. Implementar bubble sort que faz apenas uma passagem no caso de estar ordenado (uma vez booleano)

Merge sort

Uma técnica de "Divide to Conquer" ou "Dividir para conquistar"

Merge - Junção

↳ Aplicar se ambos os arrays estiverem ordenados

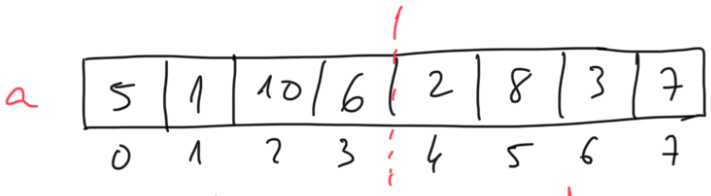


Complexidade Temporal:

n_A e n_B → dimensões arrays a e b

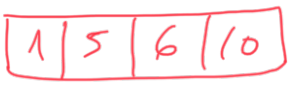
custo: $O(n_A + n_B)$

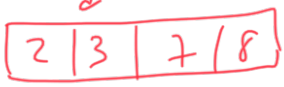
Merge sort



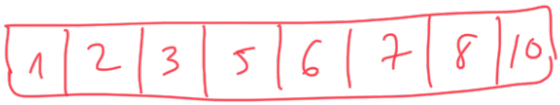
ordenar recursivamente sub array esquerdo

ordenar recursivamente sub array direito





junção ordenada (merge)




```
mergesort(a, left, right):
if (left < right)
{
    mid = (left + right) / 2
    mergesort(a, left, mid)
    mergesort(a, mid+1, right)
    merge(a, left, mid, right)
}
```