Algoritmos de Ordenação

- Baseados em comparações
  - Elementares
    - → insertion sort
    - → selection sort
    - → bubble sort
    - → merge sort
  - Não elementares
    - → quick sort
    - → heap sort
- Não baseados em comparações

Algoritmos baseados em comparações

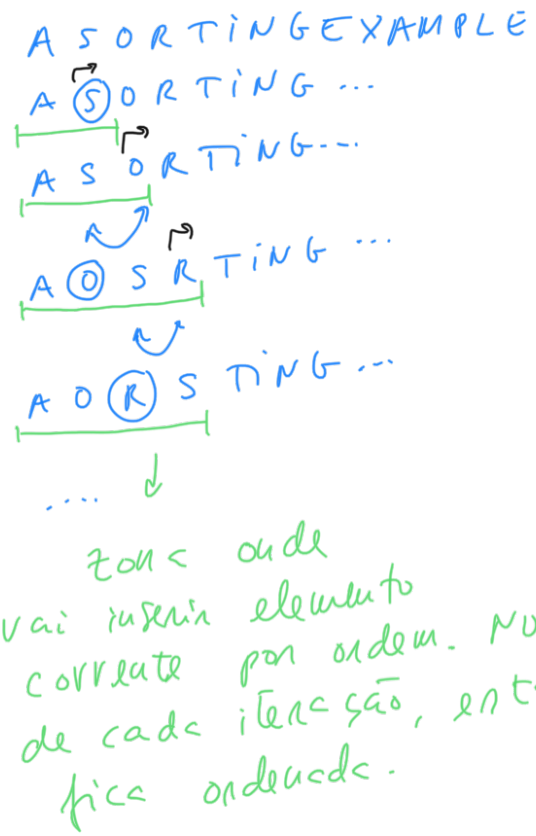
↳ para dados distribuídos de forma aleatória, não é possível em um tempo de execução inferior ao tempo linearítmico $O(N \times \log_2 N)$

→ N é o número de elementos do array a ordenar

Alg. não baseados em comparados

↳ embora não se apliquem sempre, quando se podem aplicar, sob determinadas condições, conseguem executar em tempo linear $O(N)$

Insertion sort



Funções auxiliares

```
fun less (x: Char, y: Char):
    Boolean = x < y

fun exch (a: CharArray,
          i: Int, j: Int) {
    val aux: Char = a[i]
    a[i] = a[j]
    a[j] = aux
}
```

main

```
val arr = CharArray(5)
arr[0] = 'A'
arr[1] = '5'
arr[2] = 'O'
arr[3] = 'R'
arr[4] = 'T'
print(arr)
```

Não altera o array

Irá alterar para 'z'

```
fun print(a: CharArray) {
    for (....)
    ⇒ a[0] = 'z'
}
```

a função recebe uma cópia de arr, ou recebe o array original?

Recebe o array original, passado numa nova referência

ou ponteiro

endereço do array na memória heap (array reside no endereço 0x123)

stackframe de main

Memória

STACK

HEAP

onde são alojados objetos

referência arr

ou ponteiro

0x123

a

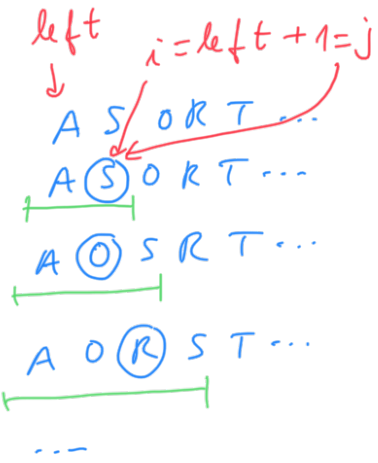
0x123

0	'A' 'z'
1	'5'
2	'O'
3	'R'
4	'T'

propriedades: size, ...

```
fun lessExch(a: CharArray, i: Int, j: Int) {
    if (less(a[i], a[j]))
        exch(a, i, j)
}

fun insertionSort1(a: CharArray, left: Int, right: Int)
{
    var i = left + 1
    while (i <= right) {
        var j = i
        while (j > left) {
            lessExch(a, j, j-1)
            --j
        }
        ++i
    }
}
```



T.P.C. insertionSort2 → otimização nas f(x) lessExch, mas apenas move o elem. anterior para a frente. É pára assim que estiver ordenado.

Complexidade
Temporal: $O(n^2)$

Pior caso: $n = right - left + 1$

ordenado de forma decrescente

1 comparação + troca para trás → janela já ordenada

$+ 2$ " " "

$+ 3$ " " "

$+ \dots$

$+ n$ → soma dos n termos duma série aritmética

$$= \sum_{i=1}^{n} i = \frac{n \times (n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} \approx n^2$$

$n \to \infty$

melhor caso: ordenado de forma crescente

Assumindo o insertionSort2 ↓ otimizado

1 comparação para trás
\+
1 " " " "
\+
1 " " " "
\+
⋮
\+
1

" $n \rightarrow \Omega(n)$