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

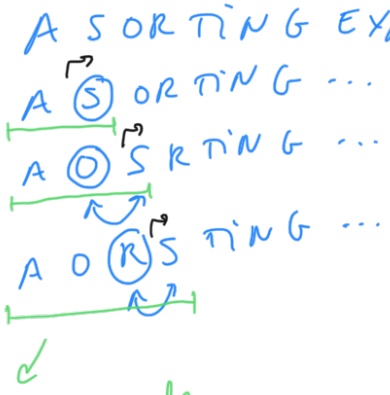
↳ para dados distribuídos de forma aleatória, não é possível ter um tempo de execução inferior ao tempo linearítmico $O(N \times \log_2 N)$

→ N é o número de elementos do array a ordenar

Alg. não baseados em comparações

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

Insertion sort



Zona onde vai inserir elemento corrente por ordem. No fim de cada iteração, esta zona fica ordenada

Funções auxiliares

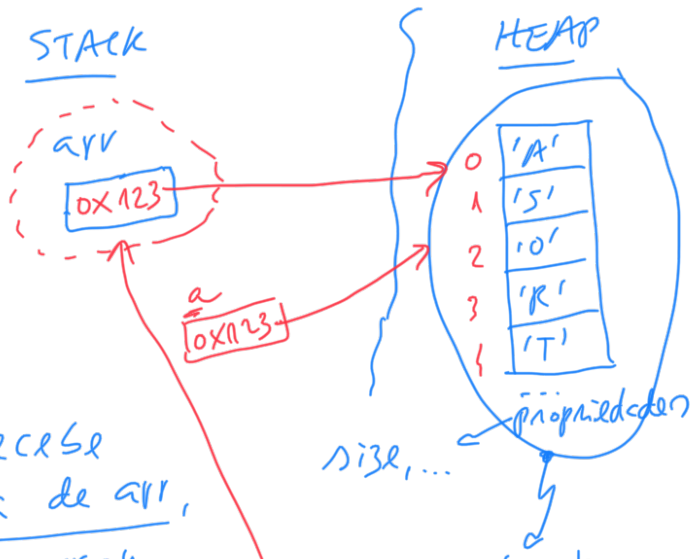
```
fun less(x:Char, y:Char):
    Boolean = x < y

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

main

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

Memória



print(arr) → Não altera o array

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

Objeto que contém o array (reside no endereço 0x123)

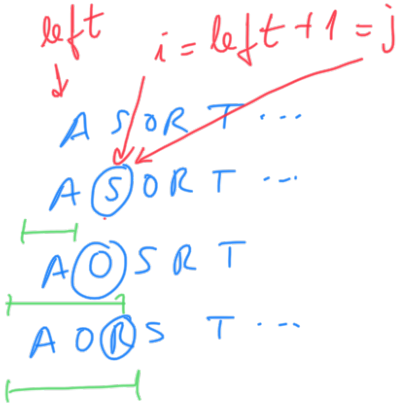
```
fun print(a: CharArray)
{ for ...
}
```

Recebe o array original passado uma nova referência ou ponteiro

endereço do array na memória heap

```
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
    }
}
```



Un optimizesaf: insertionSort2

Complexitate temporal: $O(n^2)$

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

ordenado de forma decrescente

1 comparação para trás → janela já ordenada
+
2 " " "
+
3 " " "
+
⋮
+
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

1 comparação para trás

\+

1 " " " "

\+

1 " " " "

\+

⋮

1

" $\Omega(n)$