DCC302 – ESTRUTURA DE DADOS I

Aula 15.2 – Bubble Sort

Bubble Sort

- Em comparação com outros métodos existentes, a classificação **Bubblesort** é, no geral, lenta, mas é o mais simples dos algoritmos de classificação, e encontra aplicação em determinadas situações, como a classificação de arquivos pequenos e conjuntos de dados que já se encontram semi-classificados.

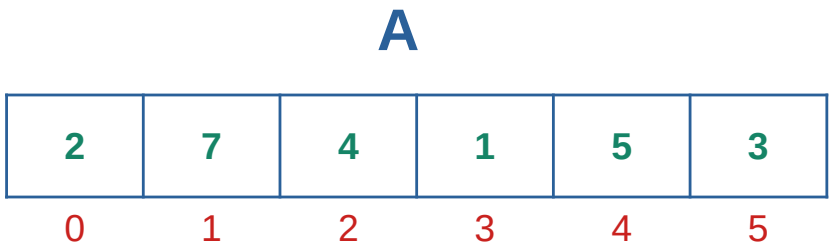
Bubble Sort

- O algoritmo **bubblesort** funciona, de forma simplificada, executando duas tarefas principais, que são executadas em loop até que os dados estejam totalmente ordenados (classificados). São elas:

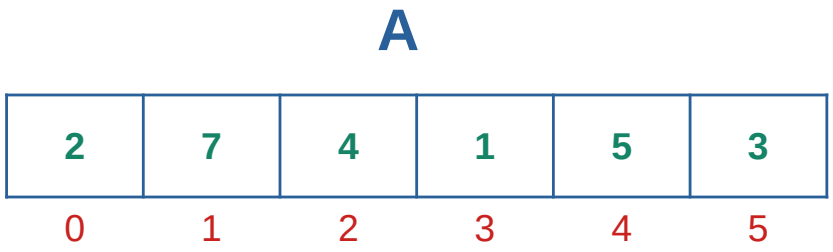
  → **Comparação de itens adjacentes**

  → **Troca de posição dos itens, quando for necessário**

Bubble Sort - Exemplo

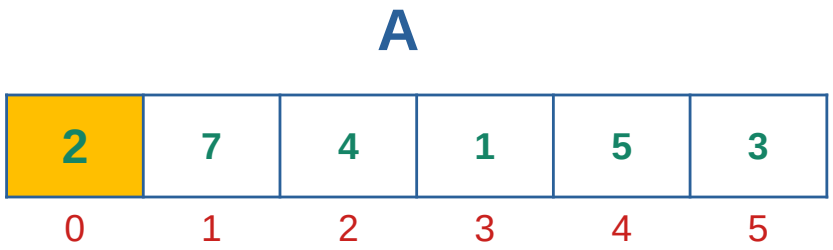


Bubble Sort - Exemplo



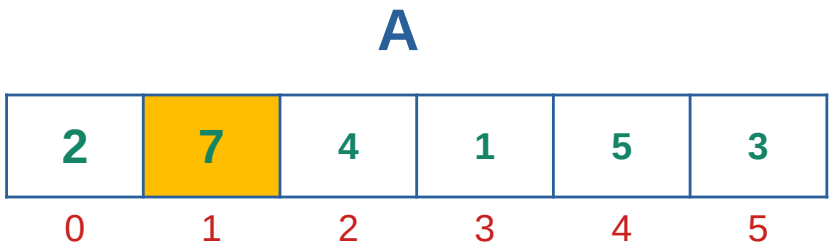
→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

Bubble Sort - Exemplo



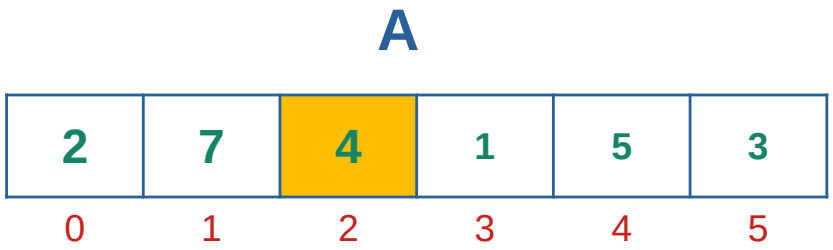
→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

Bubble Sort - Exemplo



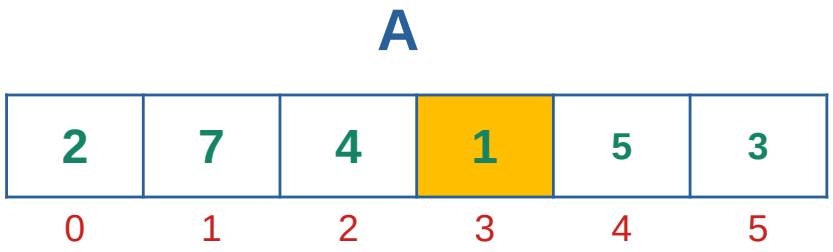
→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

Bubble Sort - Exemplo



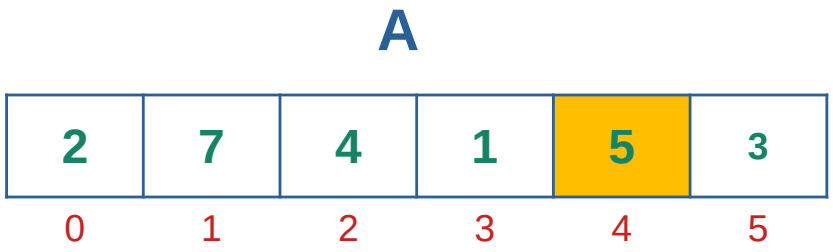
→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

Bubble Sort - Exemplo



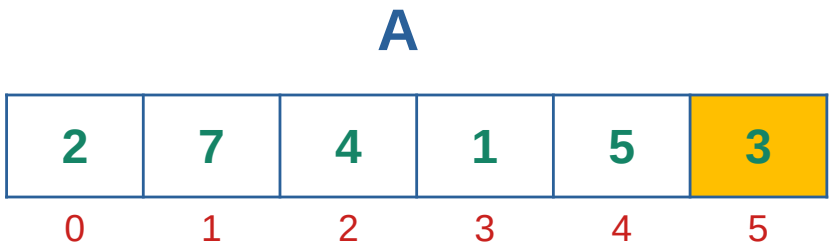
→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

Bubble Sort - Exemplo



→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

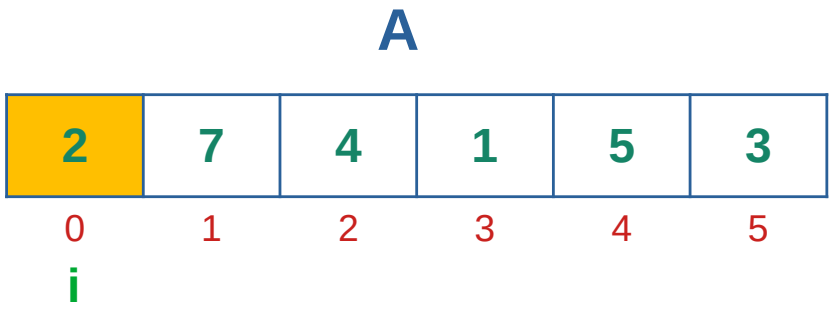
Bubble Sort - Exemplo



→ Basicamente vamos ter um laço que vai **percorrer** o array da esquerda para a direita

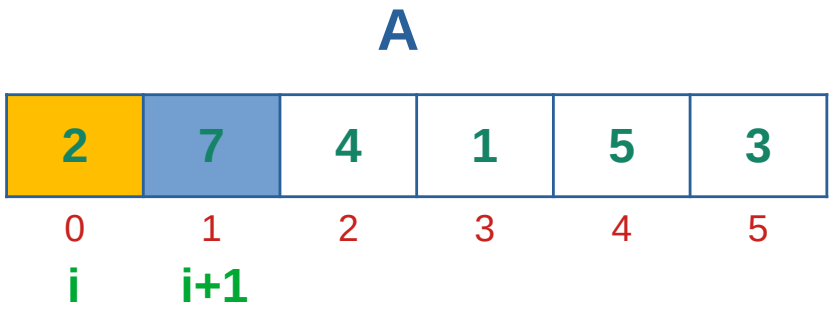
→ Lógico que não vamos fazer o escaneamento sem motivo

Bubble Sort - Exemplo



→ Pega o elemento da primeira posição e compara ele com o seu próximo.

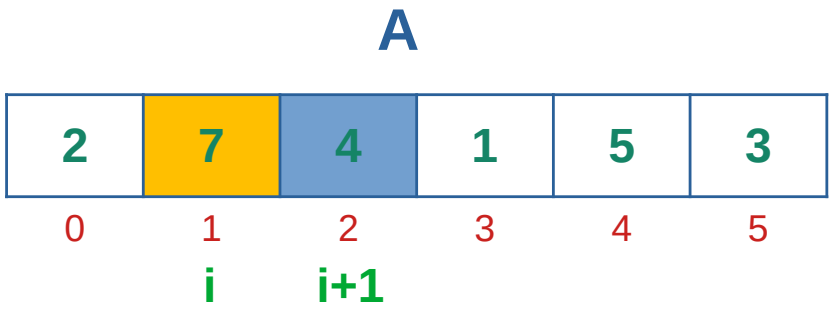
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

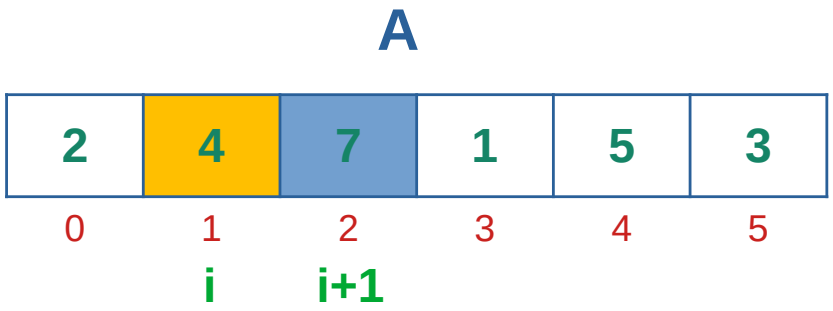
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

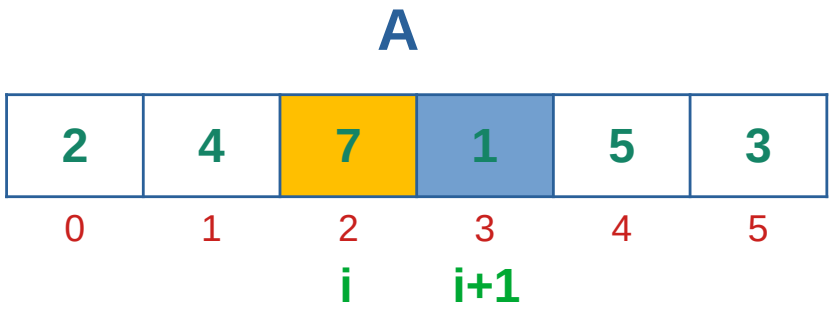
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

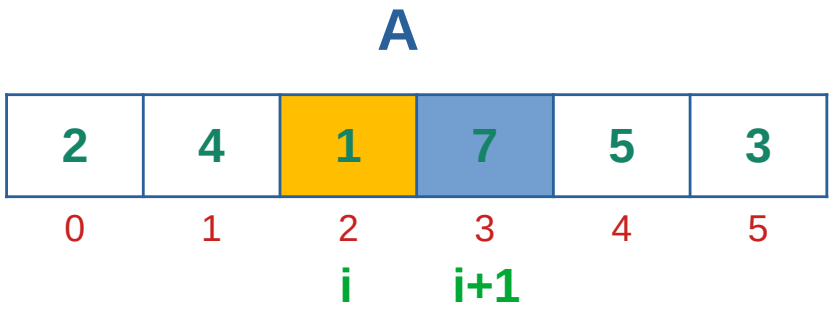
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

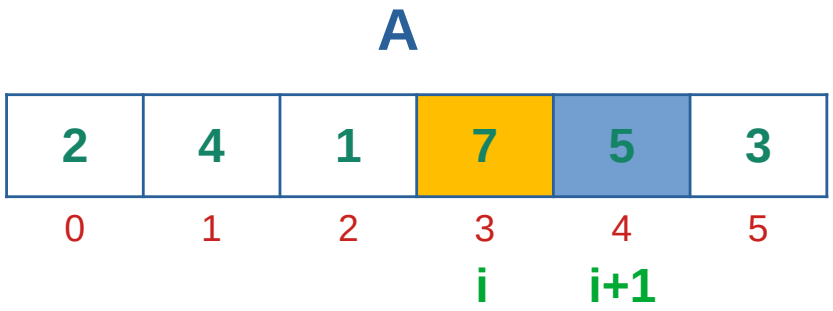
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

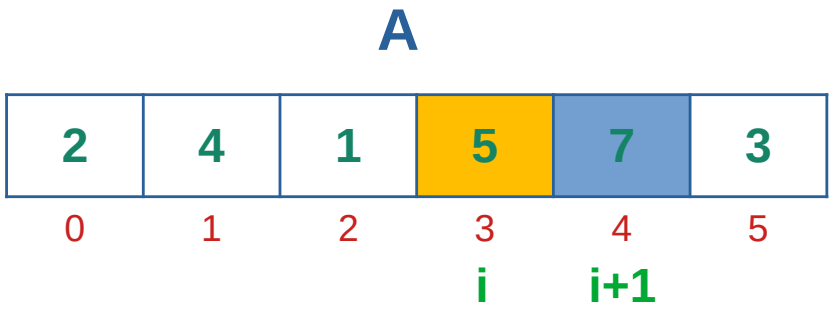
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

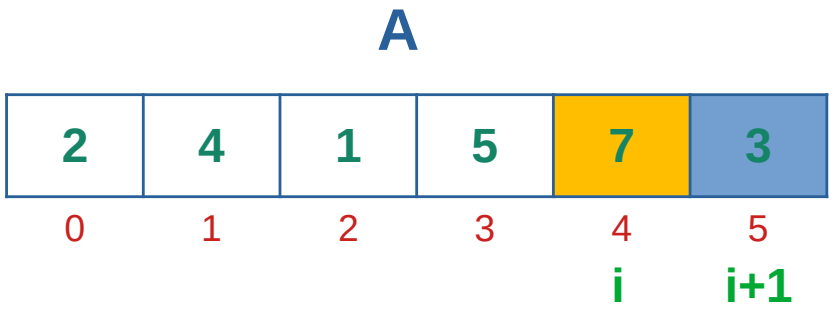
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

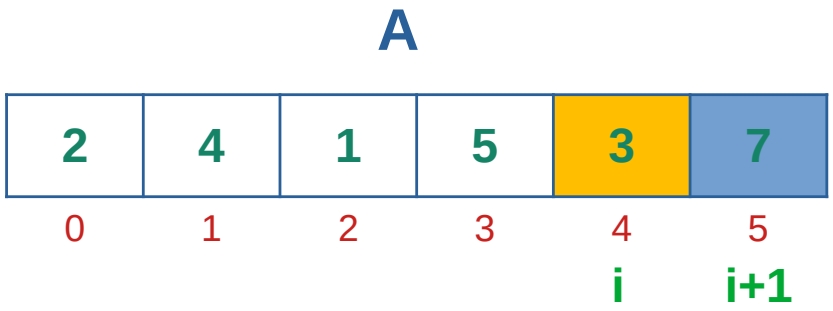
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

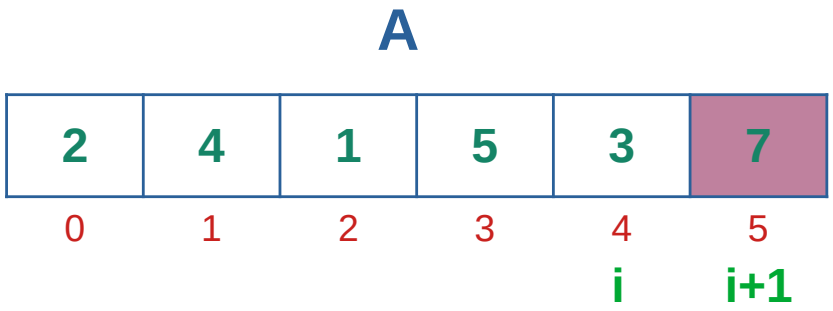
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

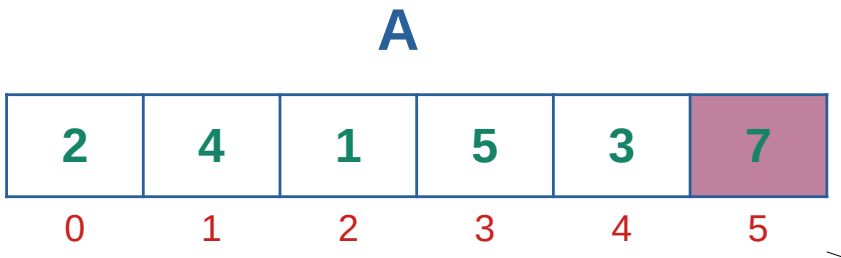
Bubble Sort - Exemplo



→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

Bubble Sort - Exemplo

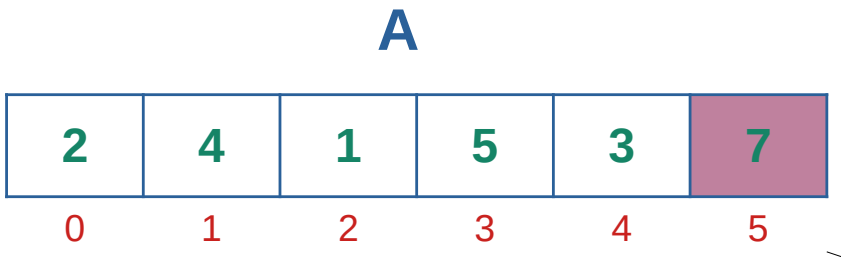


→ Pega o elemento da posição corrente e compara ele com o seu próximo.

→ **i > i+1** caso positivo, troca

```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Bubble Sort - Exemplo

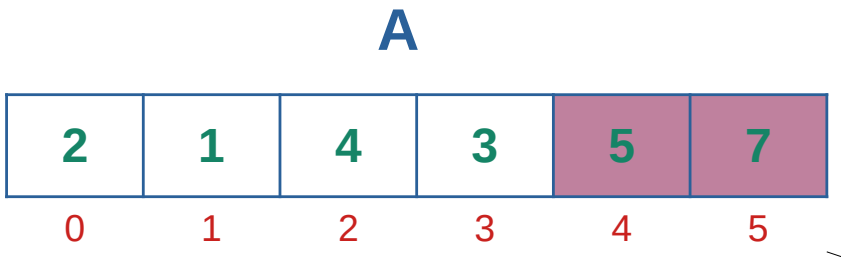


Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - 7

```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Bubble Sort - Exemplo



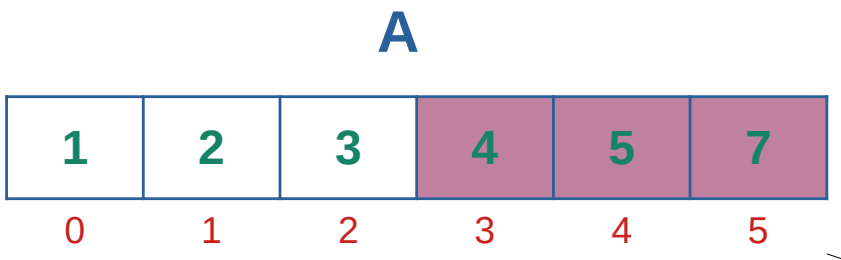
Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - 7

2ª passada: 2 - 1 - 4 - 3 - 5 - 7

```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Bubble Sort - Exemplo




```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

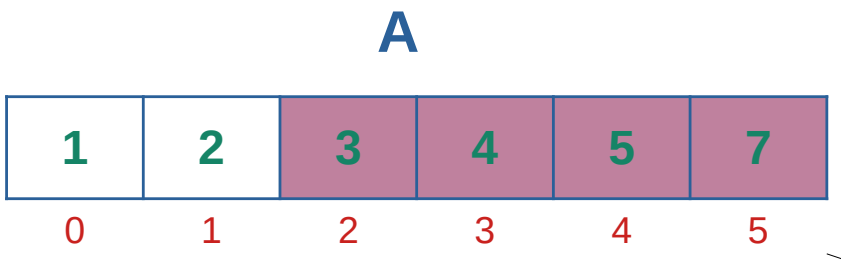
Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - 7

2ª passada: 2 - 1 - 4 - 3 - 5 - 7

3ª passada: 1 - 2 - 3 - 4 - 5 - 7

Bubble Sort - Exemplo




```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Entrada: 2 - 7 - 4 - 1 - 5 - 3

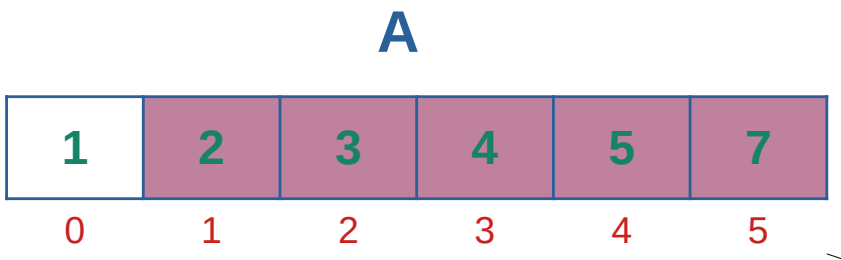
1ª passada: 2 - 4 - 1 - 5 - 3 - 7

2ª passada: 2 - 1 - 4 - 3 - 5 - 7

3ª passada: 1 - 2 - 3 - 4 - 5 - 7

4ª passada: 1 - 2 - 3 - 4 - 5 - 7

Bubble Sort - Exemplo




```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - 7

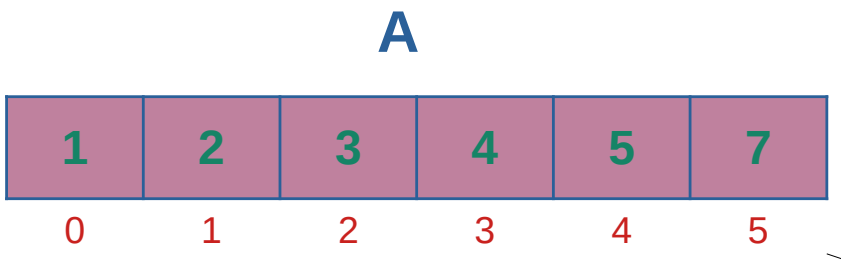
2ª passada: 2 - 1 - 4 - 3 - 5 - 7

3ª passada: 1 - 2 - 3 - 4 - 5 - 7

4ª passada: 1 - 2 - 3 - 4 - 5 - 7

5ª passada: 1 - 2 - 3 - 4 - 5 - 7

Bubble Sort - Exemplo




```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - **7**

2ª passada: 2 - 1 - 4 - 3 - **5 - 7**

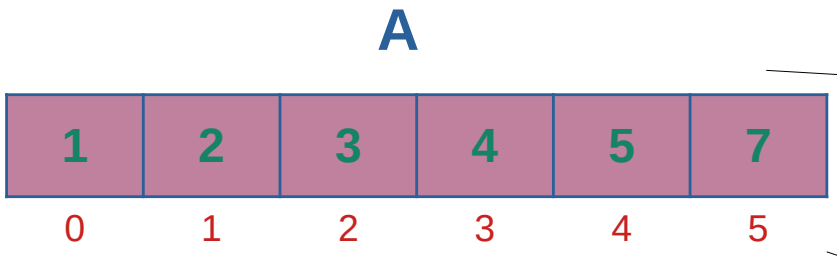
3ª passada: 1 - 2 - 3 - **4 - 5 - 7**

4ª passada: 1 - 2 - **3 - 4 - 5 - 7**

5ª passada: 1 - **2 - 3 - 4 - 5 - 7**

6ª passada: **1 - 2 - 3 - 4 - 5 - 7**

Bubble Sort - Exemplo



Precisamos de um laço externo que controle os elementos que já estão em ordem

```
for i = 0 to n-2 do:
    /* compare the adjacent elements */
    if list[i] > list[i+1] then
       swap( list[i], list[i+1] )
    end if
 end for
```

Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - **7**

2ª passada: 2 - 1 - 4 - 3 - **5 - 7**

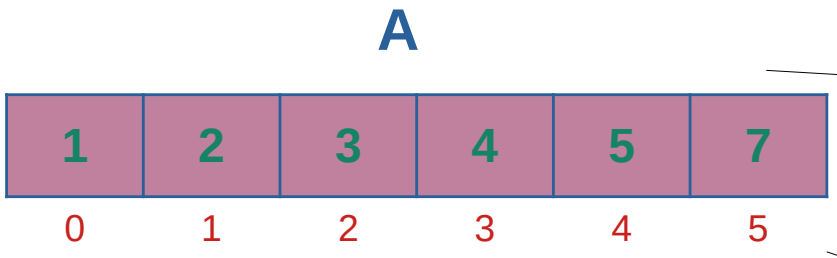
3ª passada: 1 - 2 - 3 - **4 - 5 - 7**

4ª passada: 1 - 2 - **3 - 4 - 5 - 7**

5ª passada: 1 - **2 - 3 - 4 - 5 - 7**

6ª passada: **1 - 2 - 3 - 4 - 5 - 7**

Bubble Sort - Exemplo



Precisamos de um laço externo que controle os elementos que já estão em ordem

```
for k = 0 to n-1 do:
     for i = 0 to n-2 do:
         /* compare the adjacent elements */
         if list[i] > list[i+1] then
            swap( list[i], list[i+1] )
         end if
      end for
end for
```

Entrada: 2 - 7 - 4 - 1 - 5 - 3

1ª passada: 2 - 4 - 1 - 5 - 3 - **7**

2ª passada: 2 - 1 - 4 - 3 - **5 - 7**

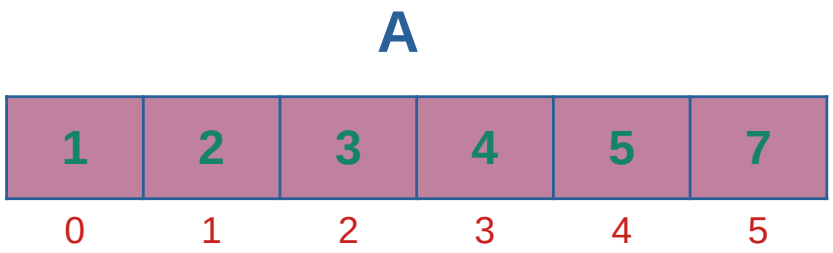
3ª passada: 1 - 2 - 3 - **4 - 5 - 7**

4ª passada: 1 - 2 - **3 - 4 - 5 - 7**

5ª passada: 1 - **2 - 3 - 4 - 5 - 7**

6ª passada: **1 - 2 - 3 - 4 - 5 - 7**

Bubble Sort - Exemplo



Complexeidade:

$T(n) = (n-1) * (n-1) * c$

$= n^2 - 2cn + 1$

$= \mathbf{O(n^2)}$

```
for k = 0 to n do:
     for i = 0 to n-1 do:
         /* compare the adjacent elements */
         if list[i] > list[i+1] then
            swap( list[i], list[i+1] )
         end if
      end for
end for
```

Bubble Sort

- Implementação