



# Algoritmo e Estrutura de Dados II

## COM-112

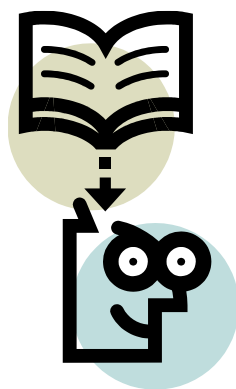
Vanessa Souza



# Ordenação

# Problema

---





# Ordenação

---

- ▶ Processo bastante utilizado na computação de uma estrutura de dados
- ▶ Ordenar significa colocar em ordem, segundo algum critério
- ▶ Alterar a ordem na qual os elementos de uma estrutura de dados aparece nessa estrutura
  - ▶ Rearranjar a estrutura





## Ordenação

---

- ▶ A ordenação de uma coleção de valores é um dos problemas computacionais mais estudados.
- ▶ É importante porque muitos problemas definidos sobre coleções de valores se tornam fáceis, se os elementos da coleção estiverem ordenados.



# Ordenação

---

5

3

1

2

4

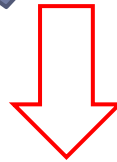
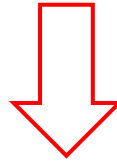




# Ordenação

---

5	3	1	2	4
---	---	---	---	---



1	2	3	4	5
---	---	---	---	---

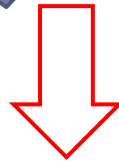
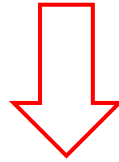




# Ordenação

---

5	3	1	2	4
---	---	---	---	---



5	4	3	2	1
---	---	---	---	---



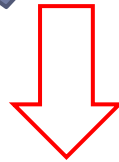
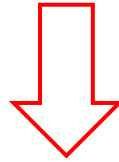




# Ordenação

---

5	3	1	2	4
---	---	---	---	---



2	4	1	3	5
---	---	---	---	---





## Por que ordenar?

---

- ▶ Dados ordenados garantem uma melhor performance de pesquisa a uma ED

- ▶ Recuperar Informação

- ▶ Busca Sequencial



$$\Theta(n)$$





## Por que ordenar?

---

- ▶ Dados ordenados garantem uma melhor performance de pesquisa a uma ED

- ▶ Recuperar Informação

- ▶ Busca Binária





## Por que ordenar?

---

- ▶ Dados ordenados garantem uma melhor performance de pesquisa a uma ED

- ▶ Recuperar Informação

- ▶ Busca Binária



$$\Theta(\log_2 n)$$

Só é possível quando os dados estão ordenados





## Ordenação

---

- ▶ Os algoritmos frequentemente usam a ordenação como sub-rotina.
- ▶ *“A complexidade da ordenação da ED não deve exceder a complexidade da computação a ser feita na ED sem o processo de ordenação”*





# Ordenação

---

## ► Tipo de Dado

- Em geral a ordenação é feita sobre um campo chave de um registro de dados
- O restante do registro é chamado de dados satélite
- Na prática, quando um algoritmo permuta uma chave, ele deve permutar também os dados satélites.

```
typedef struct vinho
{
    unsigned int codigo; chave
    unsigned int safra;
    char tipo[30];
    struct vinho *prox;
} vinho;
```

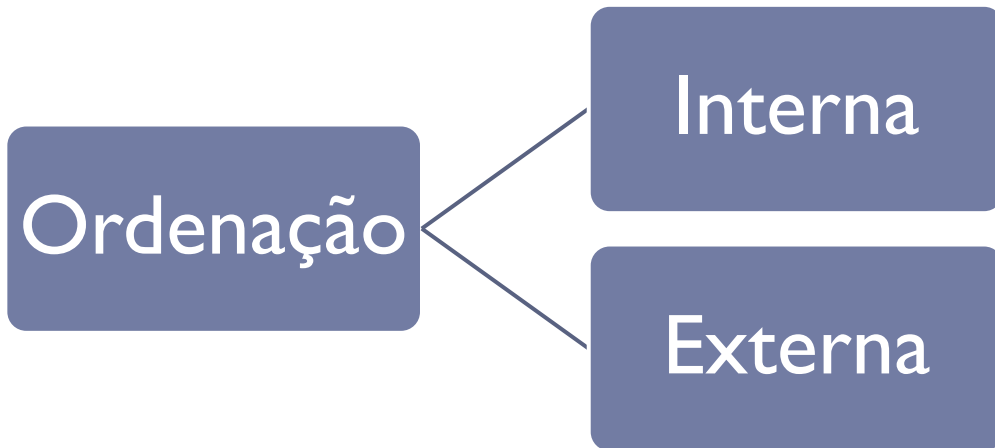
} Dados Satélite





# Classificação dos Métodos de Ordenação

---

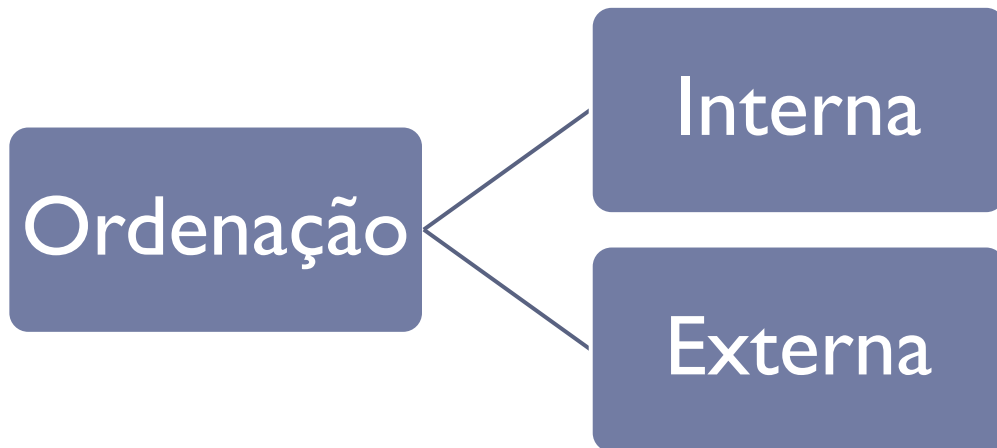


Os registros que serão ordenados cabem todos na memória principal.



# Classificação dos Métodos de Ordenação

---



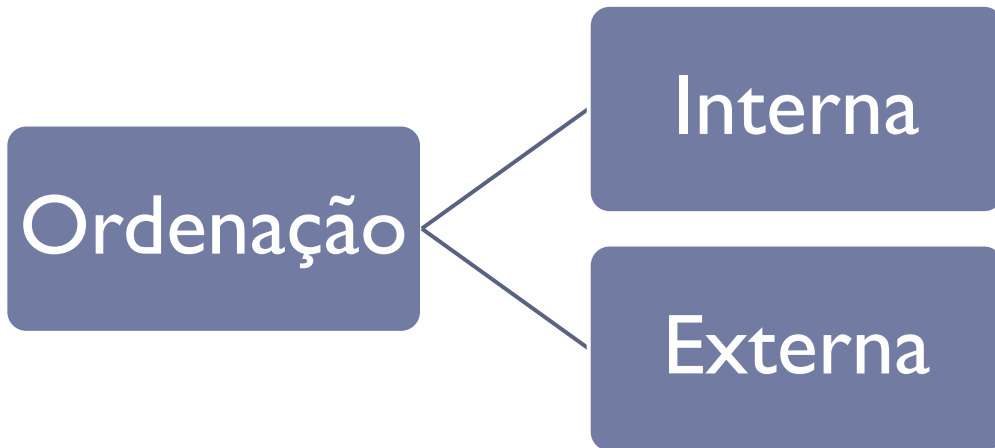
Os registros que serão ordenados não cabem na memória principal.





# Classificação dos Métodos de Ordenação

---



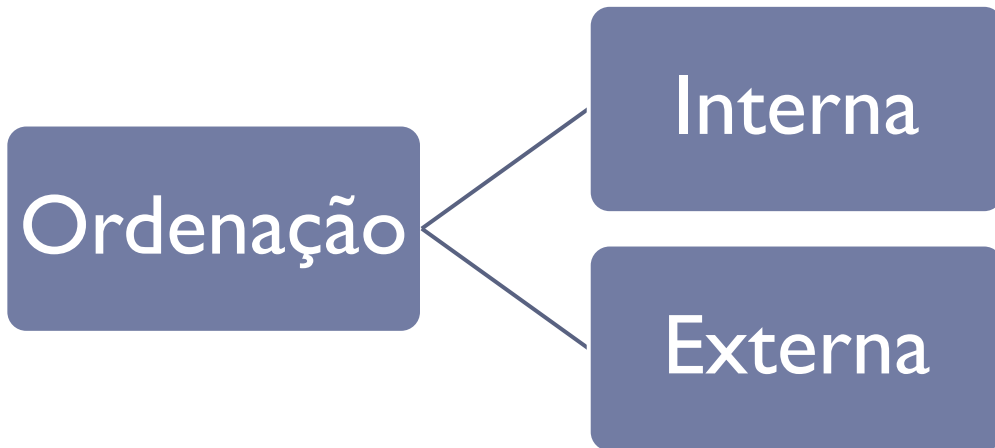
Os registros que serão ordenados não cabem na memória principal.





# Classificação dos Métodos de Ordenação

---



Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.

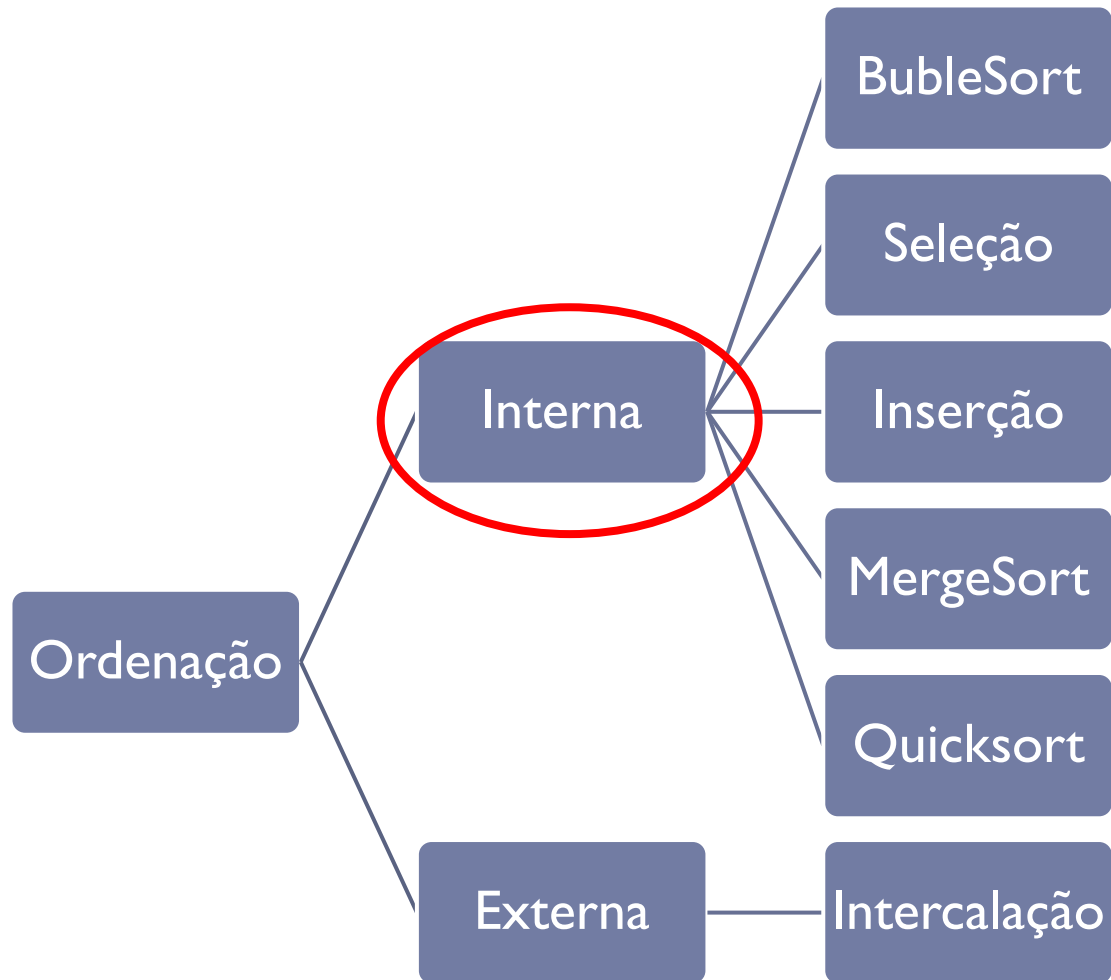
Na ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.





# Classificação dos Métodos de Ordenação

---





# Ordenação Interna



## Ordenação Interna

---

- ▶ Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- ▶ Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.
  - ▶ Número de movimentações  $M(n)$  de itens do arquivo.





## Ordenação Interna

---

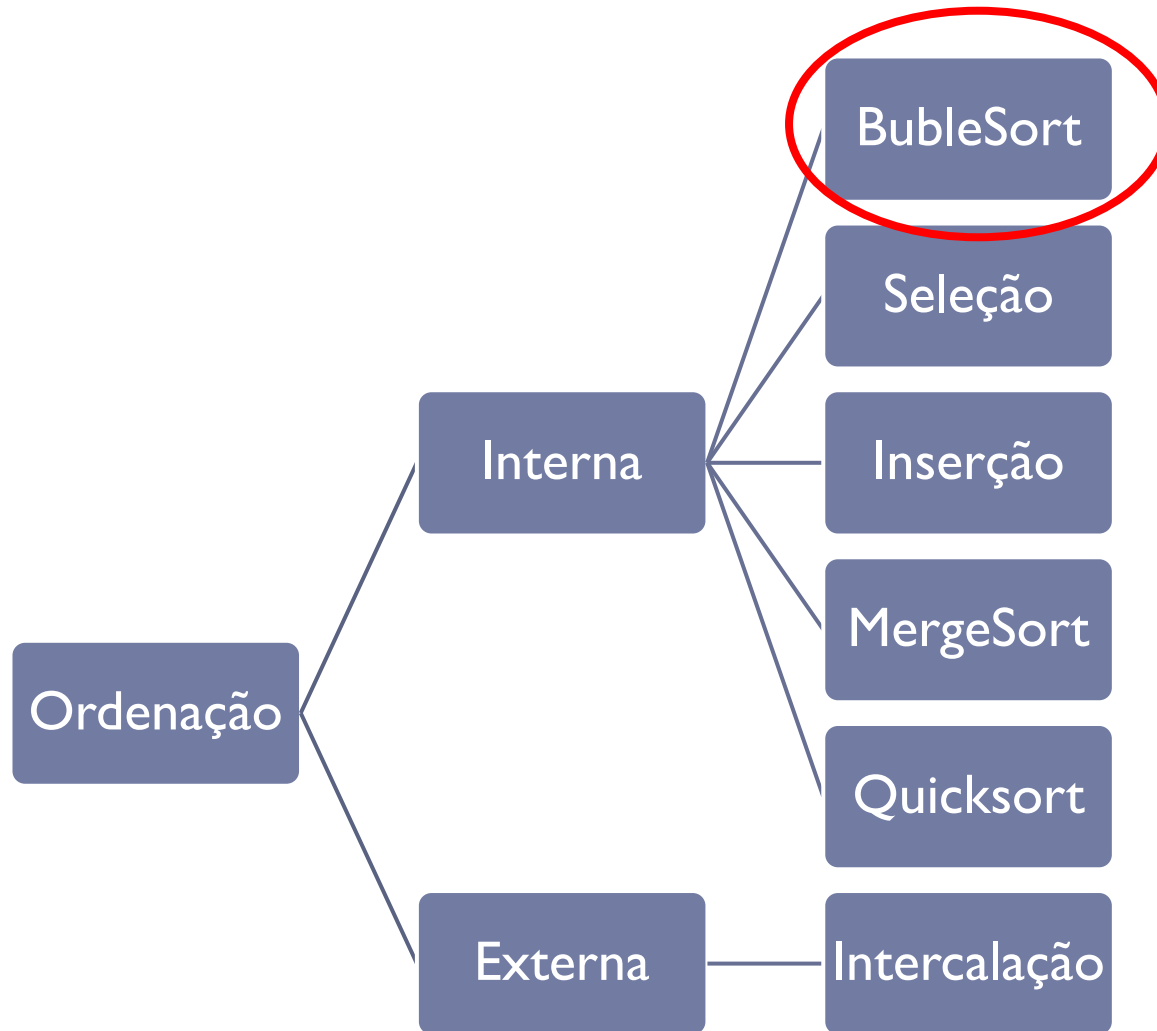
- ▶ Nesse curso, veremos algoritmos de ordenação baseados em comparação de chaves apenas.
- ▶ Consideraremos que a coleção é implementada usando a estrutura de dados do tipo vetor.
  - ▶ Mas os algoritmos podem ser usados em estruturas do tipo lista também!





# Classificação dos Métodos de Ordenação

---





## Bolha

---

- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

5	3	1	2	4
---	---	---	---	---







- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

5	3	1	2	4
---	---	---	---	---

Compara o 5 com o 3. Se  
for maior, troca





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	5	1	2	4
---	---	---	---	---

Compara o 5 com o 3. Se  
for maior, troca





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	4	5
---	---	---	---	---

Repete o procedimento  
até o fim do vetor







- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	4	5
---	---	---	---	---

Ao fim da primeira  
passada, volta ao começo  
novamente





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	4	5
---	---	---	---	---

Qual elemento está  
sabidamente ordenado???





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Reinicia o processo





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

1	2	3	4	5
---	---	---	---	---

Ao fim da segunda  
passada, volta ao começo  
novamente





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.

3	1	2	4	5
---	---	---	---	---

Quais elementos estão  
sabidamente ordenados???







- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





- ▶ Ideia: percorrer o vetor sequencialmente e comparar cada elemento com seu sucessor, efetuando uma troca se esses elementos não estiverem na ordem correta.



Repete o procedimento  
até o fim do vetor





## Bolha

---

- ▶ O BubbleSort repete a comparação entre pares  $n$  vezes.

1	2	3	4	5
---	---	---	---	---





- ▶ <https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/visualize/>
- ▶ <https://visualgo.net/en/sorting>





# Bolha – Algoritmo

---

---

## Algoritmo: ORDENAÇÃO - BOLHA

---

**Entrada:** **Vet:** Vetor de números naturais; **Tam:** Tamanho do vetor (inteiro)

**Saída:** Vet ordenado em ordem crescente

```
1 início
2   para  $i \leftarrow 0$  até  $Tam$  faça
3     para  $j \leftarrow 1$  até  $Tam$  faça
4       se  $Vet[j] < Vet[j - 1]$  então
5          $aux \leftarrow Vet[j - 1]$ 
6          $Vet[j - 1] \leftarrow Vet[j]$ 
7          $Vet[j] \leftarrow aux$ 
8       fim
9     fim
10  fim
11 fim
```

---





## ► Exercício

- Computar a função de complexidade do algoritmo Bolha

---

**Algoritmo:** ORDENAÇÃO - BOLHA

---

**Entrada:** **Vet:** Vetor de números naturais; **Tam:** Tamanho do vetor  
(inteiro)

**Saída:** Vet ordenado em ordem crescente

```
1 início
2   para  $i \leftarrow 0$  até  $Tam$  faça
3     para  $j \leftarrow 1$  até  $Tam$  faça
4       se  $Vet[j] < Vet[j - 1]$  então
5          $aux \leftarrow Vet[j - 1]$ 
6          $Vet[j - 1] \leftarrow Vet[j]$ 
7          $Vet[j] \leftarrow aux$ 
8       fim
9     fim
10  fim
11 fim
```

---





- ▶ O bolha possui uma versão mais “inteligente”.
- ▶ Quais alterações podem ser feitas no código para melhorar sua performance?





# Bolha vs Bolha Inteligente

---

Buble sort tradicional

1 2 3 4 5

Executou 5 vezes

1 2 3 4 5

Buble sort inteligente

1 2 3 4 5

Executou 1 vezes

1 2 3 4 5





## Bolha vs Bolha Inteligente

---

Buble sort tradicional

5 3 1 2 4

Executou 5 vezes

1 2 3 4 5

Buble sort inteligente

5 3 1 2 4

Executou 3 vezes

1 2 3 4 5





# Bolha

---

Buble sort tradicional

5 4 3 2 1

Executou 5 vezes

1 2 3 4 5

Buble sort inteligente

5 4 3 2 1

Executou 5 vezes

1 2 3 4 5





- ▶ Uma das principais vantagens do algoritmo BubbleSort é sua simplicidade.
- ▶ O algoritmo tem complexidade assintótica  $O(n^2)$  e funciona muito bem em vetores quase ordenados.
- ▶ BubbleSort x BubbleSortInteligente

