# Ordenação

# Ordenação


Vetor com elementos não ordenados
Algoritmo de ordenação
Vetor com elementos ordenados

# Características do vetor a ser ordenado

- Pode conter dados primitivos ou objetos
- Quando armazena objetos, deve-se estipular um ou mais atributos que vão se comportar como “chave de ordenação”
  - A chave de ordenação identifica unicamente o objeto
- Não pode haver mais deu um objeto com a mesma chave de ordenação
- Operações manipulam apenas endereços dos objetos

# Algoritmos

- Existem vários algoritmos de ordenação
  - Bolha
  - Mergesort
  - Quicksort
  - Heapsort
  - Por Seleção
  - Etc

# Algoritmo bolha
# *Bubblesort*

FURB
UNIVERSIDADE DE BLUMENAU

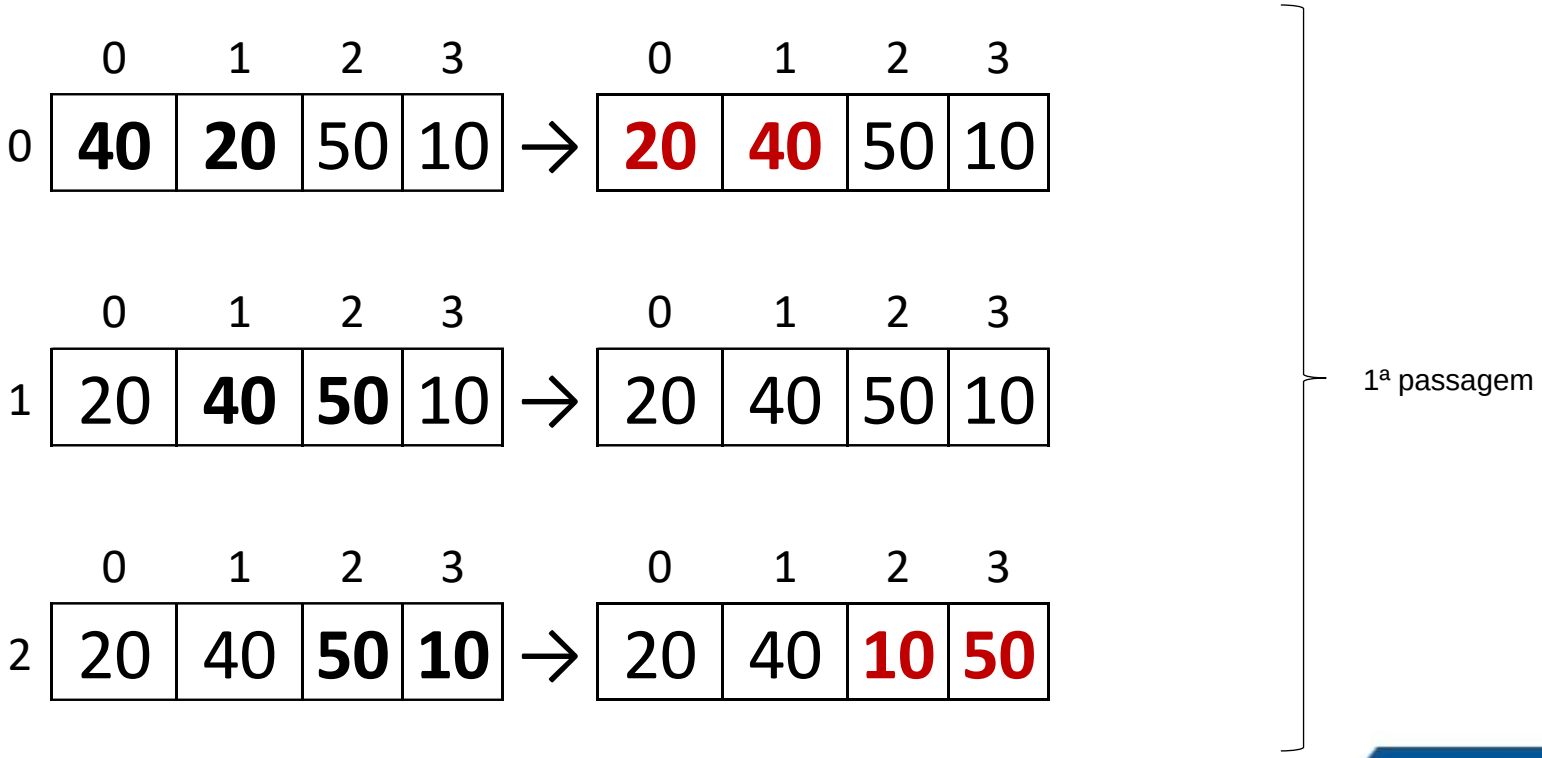
# Algoritmo de Ordenação Bolha

- Também conhecido como “bubblesort”
- Utiliza o seguinte procedimento:
  - Compare dois elementos
  - Se o da esquerda for maior, troque-os de posição
  - Desloque-se para a direita
  - Quando conseguir ordenar um elemento (colocando-o mais à direta) reinicie novamente na extremidade à esquerda

# Algoritmo Bolha

- Ordenar o vetor

| 40 | 20 | 50 | 10 |
|---|---|---|---|



# Algoritmo Bolha

- Continuação da ordenação do vetor – 2ª passagem

| 20 | 40 | 10 | 50 |
|---|---|---|---|

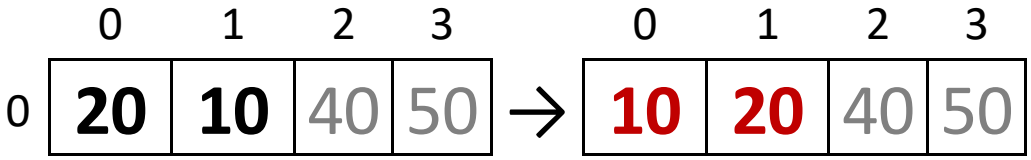
| | 0 | 1 | 2 | 3 | | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | **20** | **40** | 10 | 50 | → | 20 | 40 | 10 | 50 |

| | 0 | 1 | 2 | 3 | | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 20 | **40** | **10** | 50 | → | 20 | **10** | **40** | 50 |

# Algoritmo Bolha

- Continuação da ordenação do vetor – 3ª passagem

| 20 | 10 | 40 | 50 |
|---|---|---|---|



# Algoritmo bolha

```
Algoritmo: ordenarPorBolha(int[] info)

int i,j;
int n ← size(info);
para i ← n-1 até 1 faça
   para j ← 0 até i-1 faça
      se info[j] > info[j+1] então
         trocar(info, j, j+1);
      fim-se
   fim-para
fim-para
```

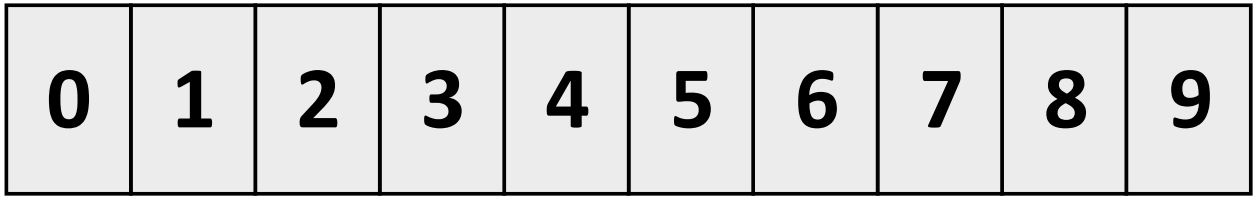
Laço que controla a quantidade de passagens
Laço decrescente

Executa uma passagem

```
Algoritmo: trocar(int[] info, int pos1, int pos2)

int temp ← info[pos1];
info[pos1] ← info[pos2];
info[pos2] ← temp;
```

# Eficiência do algoritmo




- Comparações:
  - Na 1ª passagem: 9 comparações
  - Na 2ª passagem: 8 comparações
  - Na 3ª passagem: 7 comparações
  - ...
  - Na 9ª passagem: 1 comparação
  - Total de comparações: 9 + 8 + 7 + 6 + .. 1 = 45

# Eficiência do algoritmo - Comparações

- Sendo N o número de itens do vetor, há N-1 comparações na primeira passagem, N-2 na segunda e assim sucessivamente.
- A soma de tal série é:
  - (N-1) + (N-2) + (N-3) + ... + 1 = N*(N-1)/2
- Para vetor de 10 elementos:
  - N*(N-1) / 2 é = 10 * (10-1) / 2 => 45 comparações
- Quando o vetor é grande, o “-1” é irrelevante.
  - Podemos escrever a expressão: N*N/2
  - Ou $N^2/2$ – desempenho $O(N^2)$

# Situação de melhoria

- Ordenar este vetor:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 5 | 8 | 90 | 12 | 15 | 22 | 29 | 34 | 44 |

- Após a primeira passagem:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 5 | 8 | 12 | 15 | 22 | 29 | 34 | 44 | 90 |

# Algoritmo bolha melhorado

```
Algoritmo: ordenarPorBolha(int info[])
int i,j;
int n ← size(info);
boolean trocou;

para i ← n-1 até 1 faça
   trocou ← falso;
   para j ← 0 até i-1 faça
      se info[j] > info[j+1] então
         trocar(info, j, j+1);
         trocou ← verdadeiro;
      fim-se;
   fim-para;
   se (não trocou) então
      retornar;
   fim-se;
fim-para;
```

Interrompe o algoritmo quando não ocorre nenhuma troca

# Eficiência - Trocas

- Provavelmente deve haver menos trocas do que comparações
- Com dados aleatórios, o número de trocas é mais ou menos igual à metade de comparações ($N^2/4$) ou $O(N^2)$