

# Lição 6



## Algoritmos de Ordenação

# Objetivos

Ao final desta lição, o estudante será capaz de:

- Explicar os algoritmos utilizados em ordenação por inserção, ordenação por seleção, Merge Sort e Quick Sort
- Implementar seu próprio algoritmo utilizando essas técnicas



# Ordenação

- Organização de elementos em uma ordem particular
- Usado em diversos tipos de aplicações
- Vários algoritmos de ordenação foram inventados porque essa tarefa é fundamental e freqüentemente utilizada



# Ordenação por inserção

- Um dos algoritmos mais simples
- Totalmente intuitivo e é semelhante a maneira de organizar uma coleção de cartas
  - Objetivo: Dispor uma série de cartas do menor para o maior nível
  - Preparação: cartas , tabela 1, tabela 2
  - Início: Cartas desordenadas são posicionadas na tabela 1
  - Técnica: Cartas ordenadas serão posicionadas na tabela 2
  - Busca uma carta na tabela 1, compara com as que já estão na tabela 2 e coloca esta carta em uma posição adequada na tabela 2
  - Repete esse passo até que todas as cartas sejam posicionadas na tabela 2



# Ordenação por inserção: Algoritmo

- Dividir os elementos a serem ordenados em dois grupos:
  - Seção não ordenada
  - Seção ordenada
- Repetição dos seguintes passos até não existirem elementos na parte não ordenada da coleção (array)
  - Primeiro elemento disponível na seção não ordenada do array é selecionado
  - Coloca o elemento selecionado na posição adequada na seção ordenada do array



# Ordenação por inserção: Algoritmo

```
void insertionSort(Object array[], int startIdx,  
                    int endIdx) {  
    for (int i = startIdx; i < endIdx; i++) {  
        int k = i;  
        for (int j = i + 1; j < endIdx; j++) {  
            if (((Comparable) array[k]).compareTo(  
                array[j]) > 0) {  
                k = j;  
            }  
        }  
        swap(array[i], array[k]);  
    }  
}
```



# Ordenação por inserção: Exemplo

Dados	1º passo	2º passo	3º passo	4º passo
Manga	Manga	Maçã	Laranja	Banana
Maçã	Maçã	Manga	Maçã	Laranja
Pêssego	Pêssego	Pêssego	Manga	Maçã
Laranja	Laranja	Laranja	Pêssego	Manga
Banana	Banana	Banana	Banana	Pêssego

# Ordenação por seleção

- Intuitivo e fácil de implementar
- Mais uma maneira de organizar as cartas



# Ordenação por seleção: Algoritmo

- Selecionar o elemento com o menor valor
- Trocar elemento escolhido com o elemento da posição  $i$ 
  - Começa do 1 ao  $n$
  - Onde  $n$  é o total de elementos menos 1



# Ordenação por seleção: Algoritmo

```
void selectionSort(Object array[], int startIdx,  
                    int endIdx) {  
    int min;  
    for (int i = startIdx; i < endIdx; i++) {  
        min = i;  
        for (int j = i + 1; j < endIdx; j++) {  
            if (((Comparable) array[min]).compareTo(  
                array[j])>0) {  
                min = j;  
            }  
        }  
        swap(array[min], array[i]);  
    }  
}
```



# Ordenação por seleção: Exemplo

Dados	1º passo	2º passo	3º passo	4º passo
<i>Maricar</i>	Hannah	Hannah	Hannah	Hannah
Vanessa	<i>Vanessa</i>	Margaux	Margaux	Margaux
Margaux	<i>Margaux</i>	<i>Vanessa</i>	Maricar	Maricar
<i>Hannah</i>	Maricar	<i>Maricar</i>	<i>Vanessa</i>	Rowena
Rowena	Rowena	Rowena	<i>Rowena</i>	Vanessa

# Merge Sort: Paradigma do dividir-e-conquistar

- Usa recursividade para resolver um problema
  - Problema original é dividido em subproblemas
  - Soluções para os subproblemas conduzem à solução do problema principal
- Passos:
  - Dividir
  - Conquistar
  - Combinar



# Merge Sort: Algoritmo

- Utiliza a abordagem do dividir-e-conquistar
  - Dividir
  - Conquistar
  - Combinar
- A repetição acaba quando a parte a ser ordenada possui exatamente um elemento



# Merge Sort: Algoritmo

```
void mergeSort(Object array[], int startIdx,  
                int endIdx) {  
    if (array.length != 1) {  
        mergeSort(leftArr, startIdx, midIdx);  
        mergeSort(rightArr, midIdx+1, endIdx);  
        combine(leftArr, rightArr);  
    }  
}
```



# Merge Sort: Exemplo

Dados:

7	2	5	6
---	---	---	---

Dividir o array de dados em dois:

7	2
---	---

5	6
---	---

*ArrayEsq    ArrayDir*

Dividir o ArrayEsq em dois:

7
---

2
---

*ArrayEsq    ArrayDir*

Combinar

2	7
---	---

Dividir *ArrayDir* em dois:

5
---

6
---

*ArrayEsq    ArrayDir*

Combinar

5	6
---	---

Combinar *ArrayEsq* e *ArrayDir*.

2	5	6	7
---	---	---	---



# Quick Sort: Algoritmo

- Criado por C.A.R. Hoare
- Baseado no paradigma de dividir-e-conquistar
  - Dividir
  - Conquistar
  - Sem a fase de “Combinar”





# Quick Sort: Algoritmo

```
void quickSort(Object array[], int leftIdx,  
                  int rightIdx) {  
    int pivotIdx;  
    if (rightIdx > leftIdx) {  
        pivotIdx = partition(array, leftIdx, rightIdx);  
        quickSort(array, leftIdx, pivotIdx-1);  
        quickSort(array, pivotIdx+1, rightIdx);  
    }  
}
```



# Quick Sort: Exemplo

Array dado:

3	1	4	1	5	9	2	6	5	3	5	8
---	---	---	---	---	---	---	---	---	---	---	---

Escolhe o primeiro elemento para ser o eixo = 3.

3	1	4	1	5	9	2	6	5	3	5	8
---	---	---	---	---	---	---	---	---	---	---	---

Inicializa a esquerda com o ponteiro no segundo elemento, e a direita com o ponteiro no último elemento.

left										right	
3	1	4	1	5	9	2	6	5	3	5	8

Mover o ponteiro da esquerda na direção da direita até ser localizado um valor maior do que o do eixo. Mover o ponteiro direito na direção esquerda até ser localizado um valor menor do que o eixo.

left						right					
3	1	<u>4</u>	1	5	9	2	6	5	<u>3</u>	5	8

Troca os elementos referidos com os ponteiros esquerdo e direito.

left						right					
3	1	3	1	5	9	2	6	5	4	5	8



# Quick Sort: Exemplo

Mover os ponteiros direito e esquerdo novamente.

left				right							
3	1	3	1	<u>5</u>	9	<u>2</u>	6	5	4	5	8

Trocar os elementos.

left				right							
3	1	3	1	2	9	5	6	5	4	5	8

Mover os ponteiros esquerdos e direitos novamente.

right				left							
3	1	3	1	<u>2</u>	9	5	6	5	4	5	8

Observe que os ponteiros da esquerda e direita se cruzaram e o direito < esquerdo. Neste caso, troca o eixo pelo valor do ponteiro direito.

pivot											
2	1	3	1	3	9	5	6	5	4	5	8



# Sumário

- Técnicas simples de ordenação
  - Ordenação por inserção
  - Ordenação por seleção
- Paradigma do dividir-e-conquistar
  - Merge Sort
  - Quick Sort



# Parceiros

- Os seguintes parceiros tornaram JEDI<sup>TM</sup> possível em Língua Portuguesa:

