Fernanda Pereira Caetano

proffernanda.caetano@fiap.com.br

Computational Thinking

2º semestre 2019

www.fiap.com.br

Agenda

1. Métodos de Ordenação

Ordenação

"Ordenar corresponde ao processo de rearranjar um conjunto de objetos e uma ordem específica"

- **Objetivo:** facilitar a recuperação posterior de elementos do conjunto ordenado.

→ possibilidade se acessar seus dados de modo mais eficiente.

- **Exemplos:** Dicionários e Sumários.

Ordenação

Ordenar é uma operação fundamental em computação e por isso inúmeros bons algoritmos foram desenvolvidos.

Qual o melhor algoritmo?

Depende, por exemplo, do número de itens a serem ordenados, se os números já estão mais ou menos ordenados e outros.

Métodos de Ordenação de vetores

- Bubblesort
- Quicksort
- Mergesort
- Selectionsort
- Heapsort
- Insertionsort
- Shellsort
- Radixsort
- Bucketsort

Bubblesort

Para leitura:

http://www.devmedia.com.br/entendendo-o-algoritmo-bubble-sort-em-java/24812

Acesso em 09/09/2019 – 22:45

Selectionsort

É um algoritmo que ordena itens verificando repetidamente os itens restantes para encontrar o menor deles e movê-lo para uma posição final.

Selectionsort

Selectionsort ou ordenação por seleção

- A ideia por trás do selectionsort é que para ordenar N itens você tem que passar por todos eles.

- No primeiro passo você encontra o maior valor, e então troca ele pelo último item. Assim o maior item está agora na posição N.

- No segundo passo você faz uma varredura apenas nos N-1 elementos.

Selectionsort

O maior dos itens é trocado de posição com o item na posição N-1.

Assim o maior de todos os itens está agora na última posição; o segundo maior na segunda maior posição.

Este processo é repetido, com um item sendo colocado na sua posição correta a cada vez.

Selectionsort

Depois de N passos, a coleção inteira de dados está ordenada.

Uma variação simples é encontrar o menor item a cada vez e colocá-lo na frente.

Para ordenar em ordem decrescente, o maior item é encontrado a cada vez e movido para a frente.

Selectionsort

```
import javax.swing.*;

public class Teste1 {
    public static void main(String[] args) {
        int vetor[] = {2,4,1,5,10,7,3};
        int menor_valor, menor_indice;
        String ordenados = "";

        for (int i = 0; i < vetor.length - 1; i++) {
            menor_valor = vetor[i];
            menor_indice = i;

            for (int j = i + 1; j < vetor.length; j++){
                if (vetor[j] < menor_valor){
                    menor_valor = vetor[j];
                    menor_indice = j;
                }
            }
            vetor[menor_indice] = vetor[i];
            vetor[i] = menor_valor;
        }
```

Selectionsort

```
        for (int n=0; n<vetor.length; n++){
            ordenados += vetor[n]+"  ";
          }
          JOptionPane.showMessageDialog(null,ordenados);
      }
  }
```

Faça o teste de mesa para:

vetor[] = {2,4,1,5,10,7,3}

Insertionsort

O algoritmo de inserção funciona da mesma maneira com que muitas pessoas ordenam cartas em um jogo de baralho como o pôquer.

Insertionsort

Insertionsort ou ordenação por inserção

É um simples algoritmo de ordenação, eficiente quando aplicado a um pequeno número de elementos.

Em termos gerais, ele percorre um vetor de elementos da esquerda para a direita e à medida que avança vai deixando os elementos mais à esquerda ordenados.

Insertionsort

Insertionsort parte um vetor em duas regiões: a região ordenada e a não ordenada.

Inicialmente, o vetor corresponde à região não ordenada.

Em cada passo, pegamos no primeiro elemento da região não ordenada e colocamo-lo na ordem correta da região ordenada.

Insertionsort

A inserção do item em uma posição adequada na sequencia de destino é realizada com a movimentação das chaves maiores para a direita e então é feita a inserção do item na posição vazia.

Insertionsort

Resumo:

Método de ordenação, na qual são procurados sucessivos elementos que se encontram fora de ordem, retira o elemento da lista e depois insere o elemento de forma ordenada. Este tipo de ordenação em pequenas listas é rápido, sendo extremamente lento para grandes listas.

Insertionsort

```
import javax.swing.*;
public class Teste1 {
    public static void main(String[] args) {
        int vetor[] = {2,4,1,5,10,7,3};
        int valor, j=0;
        String ordenados = "";

        for (int i = 1; i < vetor.length; i++) {
           valor = vetor[i];
           j=i-1;
           while(j >=0 && vetor[j] > valor){
                    vetor[j+1] = vetor[j];
            j=j-1;
           }
           vetor[j+1] = valor;
        }
```

Insertionsort

```
            for (int n=0; n<vetor.length; n++){
                  ordenados += vetor[n]+"  ";
            }
            System.out.println(ordenados);
      }
}
```

Faça o teste de mesa para:

vetor[] = {2,4,1,5,10,7,3}

- **Exercício**

Faça um programa em Java para controle de peças.

Menu:
1- Cadastrar
2- Vender
3- Comprar
4- Listar
5- Sair

Método cadastrar
Obrigatório cadastrar a peça e voltar para o menu
- **cod_peca (inteiro)**
- **desc_peca (string)**
- **qtde_estoque (inteiro)**
- **valor_unit (float)**
- **est_mínimo (inteiro)**

Método vender

- **Obrigatório cadastro prévio da peça que deseja vender**
- **O usuário digitará o código da peça. Se existir no cadastro de peças, mostrar a descrição e solicitar quantas peças deseja vender.**
- **Verificar se tem quantidade suficiente de peças (qtde_estoque). Se possível a venda, deverá ser subtraído a quantidade solicitada de qtde_estoque. Se não tiver quantidade suficiente mostrar mensagem para o usuário com a quantidade em estoque.**
- **Considerar estoque mínimo**

Método comprar

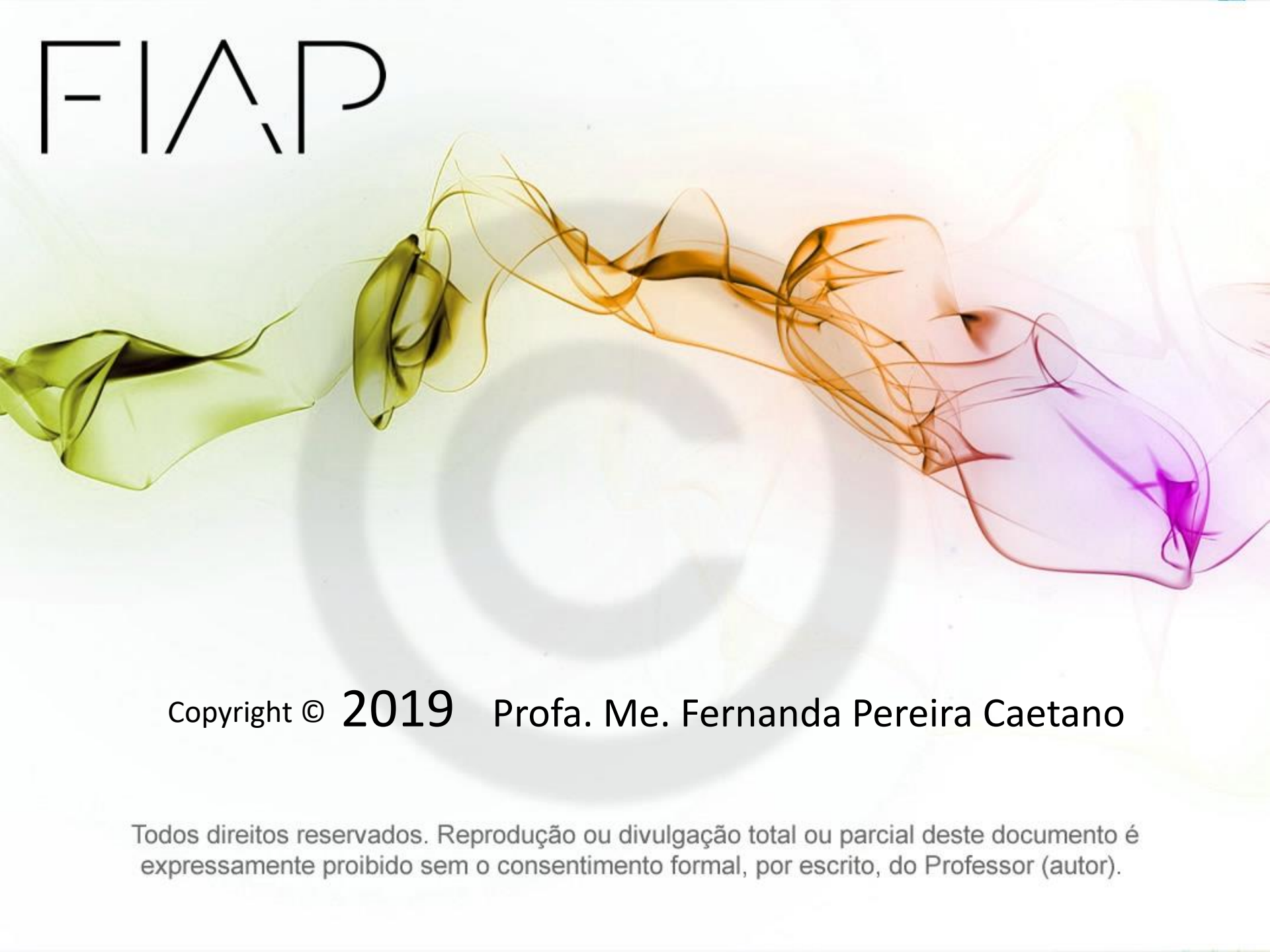
- **Obrigatório cadastro prévio da peça que deseja comprar**
- **O usuário digitará o código da peça. Se existir no cadastro de peças, mostrar a descrição e solicitar quantas peças deseja comprar.**
- **Deverá ser somada a quantidade solicitada com a qtde_estoque.**

Vetores, Ordenação

Método listar

•O usuário poderá escolher se quer o relatório por ordem de:
-Código da peça
-Nome da peça
-Valor unitário

Dúvidas ?



Copyright © 2019 Profa. Me. Fernanda Pereira Caetano

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).