



**INSTITUTO
FEDERAL**

Sudeste de
Minas Gerais

Campus
Manhuaçu

Estruturas de Dados I

Métodos de ordenação

Prof. Leonardo C. R. Soares - leonardo.soares@ifsudestemg.edu.br

Instituto Federal do Sudeste de Minas Gerais

10 de janeiro de 2024





Volta do recesso

Antes do recesso escolar, estudamos árvores binárias. Você lembra-se de todos os conceitos que foram apresentados? Se não, deve revisá-los. Caso lembre, indico abaixo alguns exercícios que irão ajudar a relembrar e praticar a aplicação dos conceitos no desenvolvimento de pequenos algoritmos. Para resolver os exercícios (sem precisar refazer toda a árvore), utilize a árvore binária apresentada como exemplo:

- ▶ Implemente um método que retorne a quantidade de elementos que uma árvore possui;
- ▶ Implemente um método que retorne a quantidade de folhas que uma árvore possui;
- ▶ Implemente um método que retorne a soma dos valores (atributo valor) de todos os nós da árvore.





Volta do recesso

Antes do recesso escolar, estudamos árvores binárias. Você lembra-se de todos os conceitos que foram apresentados? Se não, deve revisá-los. Caso lembre, indico abaixo alguns exercícios que irão ajudar a relembrar e praticar a aplicação dos conceitos no desenvolvimento de pequenos algoritmos. Para resolver os exercícios (sem precisar refazer toda a árvore), utilize a árvore binária apresentada como exemplo:

- ▶ Implemente um método que retorne a quantidade de elementos que uma árvore possui;
- ▶ Implemente um método que retorne a quantidade de folhas que uma árvore possui;
- ▶ Implemente um método que retorne a soma dos valores (atributo valor) de todos os nós da árvore.

Faça os exercícios!





Ordenação

Descrição

- ▶ Ordenação é o processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.





Ordenação

Descrição

- ▶ Ordenação é o processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- ▶ A ordenação visa facilitar a recuperação posterior dos itens do conjunto ordenado.





Ordenação

Descrição

- ▶ Ordenação é o processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- ▶ A ordenação visa facilitar a recuperação posterior dos itens do conjunto ordenado.
- ▶ As ordens mais comumente utilizadas são a numérica e a lexicográfica.





Ordenação

Problema de ordenação - Definição formal

Entrada: Uma sequência de n objetos $\{ a_1, a_2, \dots, a_n \}$

Saída: Uma permutação ordenada da sequência de entrada tal que
 $a_1, \leq a_2, \dots, \leq a_n$.





Ordenação

Características

- ▶ Cada elemento do conjunto a ser ordenado deve possuir uma **chave** para controlar a ordenação.





Ordenação

Características

- ▶ Cada elemento do conjunto a ser ordenado deve possuir uma **chave** para controlar a ordenação.
- ▶ Qualquer tipo de chave sobre o qual exista uma regra de ordenação bem-definida pode ser utilizada.





Ordenação

Características

- ▶ Cada elemento do conjunto a ser ordenado deve possuir uma **chave** para controlar a ordenação.
- ▶ Qualquer tipo de chave sobre o qual exista uma regra de ordenação bem-definida pode ser utilizada.
- ▶ Um método de ordenação é **estável** se a ordem relativa dos itens com chaves iguais é mantida durante a ordenação.





Ordenação

Os algoritmos de ordenação podem ser classificados em:

- ▶ Ordenação interna: Os dados a serem ordenados estão na memória principal.
- ▶ Ordenação externa: Os dados a serem ordenados não podem ser totalmente armazenados na memória principal, necessitando de armazenamento na memória auxiliar.

Dado o escopo da disciplina estudaremos apenas métodos de ordenação interna.





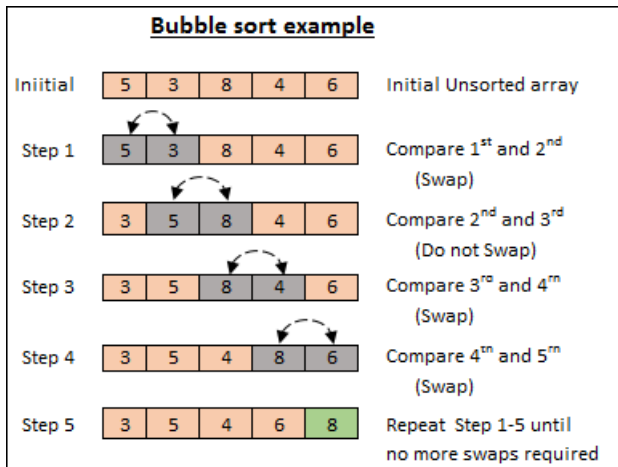
BubbleSort

BubbleSort (ou método de bolha) é um algoritmo de ordenação extremamente simples. Ele trabalha trocando repetidamente pares de elementos adjacentes que estejam na ordem errada. Possui complexidade assintótica limitada por $\mathcal{O}(n^2)$.





BubbleSort





BubbleSort

Podemos ver uma animação demonstrando passo-a-passo o funcionamento do algoritmo aqui.





BubbleSort

```
public static void bubbleSort(final int arr[]){  
    int temp;  
    boolean trocou;  
    final int n = arr.length;  
    for (int i = 0; i < n - 1; i++){  
        trocou = false;  
        for (int j = 0; j < n - i - 1; j++){  
            if (arr[j] > arr[j + 1]){  
                temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
                trocou = true;  
            }  
        }  
        if (!trocou) // Se não trocou, está ordenado  
            break;  
    }  
}
```





Exercícios práticos

1. Implemente uma lista simplesmente encadeada de Alunos. Cada aluno deverá ter um número de matrícula e um nome. O número de matrícula será utilizado como chave. Implemente um método bubbleSort que ordene a lista.





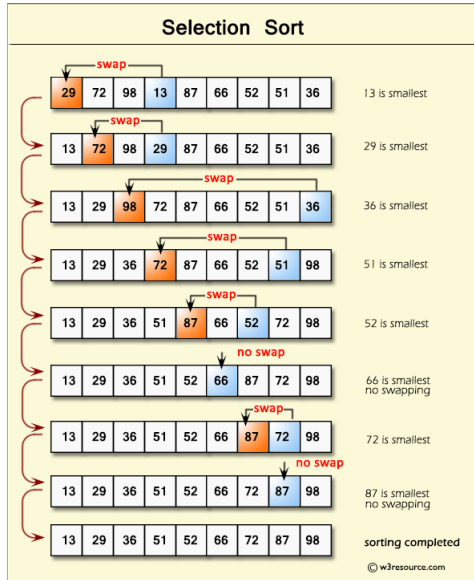
SelectionSort

SelectionSort (ou ordenação por seleção) é um algoritmo de ordenação baseado na seleção contínua do menor valor (ou maior, dependendo da ordenação desejada) do conjunto. Na primeira iteração, o elemento de menor valor vai para a primeira posição, trocando de lugar com o elemento original. Na segunda iteração, o elemento de menor valor vai para a segunda posição trocando de lugar com o segundo elemento. O processo se repete com todos os $n - 1$ elementos. A complexidade assintótica do algoritmo é limitada por $\mathcal{O}(n^2)$.





Selection Sort





SelectionSort

Podemos ver uma animação demonstrando passo-a-passo o funcionamento do algoritmo aqui.





SelectionSort

```
public static void selectionSort(int arr[]){  
    int tam = arr.length;  
    for (int i=0; i < tam - 1; i++){  
        int min = i;  
        for (int j=i+1;j<tam; j++){  
            if (arr[j]<arr[min])  
                min=j;  
        }  
        int aux = arr[i];  
        arr[i] = arr[min];  
        arr[min] = aux;  
    }  
}
```





SelectionSort

```
public static void selectionSort(int arr[]){  
    int tam = arr.length;  
    for (int i=0; i < tam - 1; i++){  
        int min = i;  
        for (int j=i+1;j<tam; j++){  
            if (arr[j]<arr[min])  
                min=j;  
        }  
        int aux = arr[i];  
        arr[i] = arr[min];  
        arr[min] = aux;  
    }  
}
```

Este algoritmo é estável?





Exercícios

- Desenvolva uma aplicação que permita cadastrar uma lista de animais de estimação utilizando vetor. Cada animal deverá possuir, código, nome e raça. O campo código será utilizado como campo chave. Faça um programa que cadastre dez animais, ordene os mesmos utilizando *selectionsort* e imprima os animais em ordem.





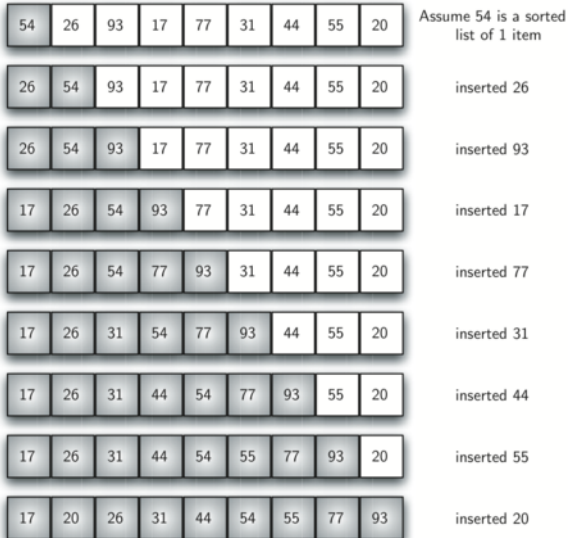
InsertionSort

InsertionSort (ou ordenação por inserção) é um algoritmo de ordenação baseado na ideia de manter uma sublista ordenada nas posições inferiores da lista. Cada novo elemento é *inserido* na sublista anterior de modo que a sublista ordenada fique com um item a mais. A complexidade assintótica do algoritmo é limitada por $\mathcal{O}(n^2)$.





InsertionSort





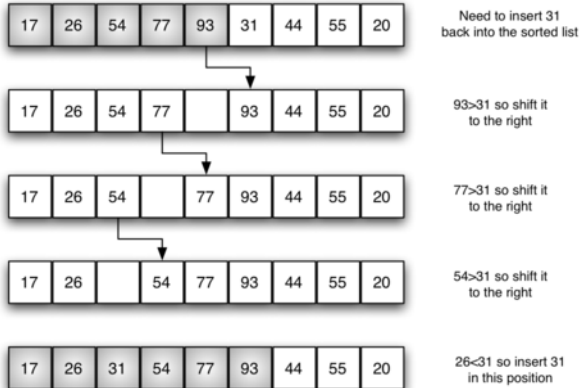
InsertionSort

A Figura a seguir mostra a quinta passagem do algoritmo em detalhe. Nesse ponto do algoritmo, há uma sublista com cinco itens: 17, 26, 54, 77 e 93. Queremos inserir 31 nesse conjunto já ordenado. A primeira comparação faz com que o item 93 vá para a direita. Os itens 77 e 54 também são deslocados para frente. Quando o item 26 é encontrado, o processo de deslocamento para e o 31 é colocado na lacuna aberta. Temos agora uma sublista ordenada com seis itens.





InsertionSort





InsertionSort

Podemos ver uma animação demonstrando passo-a-passo o funcionamento do algoritmo aqui.





Exercícios

1. Suponha que você tenha a seguinte lista de números para ordenar: [15, 5, 4, 18, 12, 19, 14, 10, 8, 20] como estará a lista parcialmente ordenada depois três passagens completas da ordenação por inserção?
2. Implemente um método que ordene um vetor utilizando insertionsort





InsertionSort

```
public static void insertionSort(int arr[]){  
    int n = arr.length;  
    for (int i = 1; i < n; i++) {  
        int elemento = arr[i];  
        int j = i - 1;  
  
        while (j >= 0 && arr[j] > elemento) {  
            arr[j + 1] = arr[j];  
            j = j - 1;  
        }  
        arr[j + 1] = elemento;  
    }  
}
```



