

Universidade Paulista – UNIP

Ciência da Computação

**APS: Atividade Prática Supervisionada**

**Tema: Sistema para análise de performance de**

**algoritmos de ordenação de dados**

|  |  |
| --- | --- |
| *Daniela Ferreira* | *D394BC-7* |
| *Leonardo Henrique Machado* | *D41186-1* |
| *Lucas Merlin de Oliveira* | *D22398-4* |
| *Maycon Ravyer de A. Brasil* | *D2602E-0* |
| *Paulo Henrique Paiva da Silva* | *D4122I-8* |

**São Paulo**

**2018**

**ÍNDICE**

[**1.** **INTRODUÇÃO** 3](#_Toc528824752)

[**2.** **REFERENCIAL TEÓRICO** 5](#_Toc528824753)

[**2.1.** **Ordenação Bolha *(Bubble sort)*** 5](#_Toc528824754)

[**2.2.** **Busca Binária** 6](#_Toc528824755)

[**3.** **DESENVOLVIMENTO** 8](#_Toc528824756)

[**4.** **RESULTADOS E DISCUSSÃO** 9](#_Toc528824757)

[**5.** **CONSIDERAÇÕES FINAIS** 10](#_Toc528824758)

[**6.** **REFERÊNCIAS BIBLIOGRÁFICAS** 11](#_Toc528824759)

[**7.** **CÓDIGO FONTE** 12](#_Toc528824760)

[**8.** **FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS** 13](#_Toc528824761)

# 

# **INTRODUÇÃO**

É inegável o fato de que os métodos de programação, de alguma forma, estão se atualizando. Os modelos antigos estão sendo substituídos e/ou incrementados por novas técnicas e aplicações lógicas a fim de sanar as necessidades tecnológicas atuais.

Porém, ainda assim os métodos antigos ainda são utilizados em massa, seja por conta de sistemas, seja pela aplicabilidade ou até mesmo a “exatidão” em seus resultados.

Em diversas aplicações do dia a dia de um programador, sejam cientificas ou comerciais, iremos nos deparar com problemas de ordenação de elementos, como por exemplo, efetuar a ordenação de uma sequência de números em ordem crescente ou decrescente, ordenar nomes de uma lista de clientes em ordem alfabética, ordenar a valor do salário dos funcionários de um determinado departamento da empresa, entre outros eventos. Para ordenar tais elementos da maneira mais eficaz possível se faz necessário o uso de um algoritmo de ordenação.

No mercado atual existe um número alto de algoritmos de ordenações disponibilizados, obter o conhecimento deles e as suas aplicações é algo crucial para ser um programador respeitado no mercado de trabalho. Conhecendo estes algoritmos, o programador poderá efetuar a melhor escolha do algoritmo a ser utilizado de acordo com a necessidade de cliente e/ou projeto, melhorando o desempenho de sua aplicação.

Com essa ideia em mente, serão apresentadas algumas metodologias de ordenação de dados, que são classificados em:

• Ordenação Interna: onde todos os elementos a serem ordenados cabem na memória principal e qualquer registro pode ser imediatamente acessado.

• Ordenação Externa: onde os elementos a serem ordenados não cabem na memória principal e os registros são acessados sequencialmente ou em grandes blocos.

Essas ordenações se dão por meio de algoritmos, que por sua vez serão avaliadas e comparadas por meios gráficos. Dessa forma, será apresentado o método mais eficiente e nos mais variados testes. Mas antes, vale relembrar um conceito importante:

* ALGORITMO

***“Um algoritmo nada mais é do que uma instrução que mostra passo a passo os procedimentos necessários para a resolução de uma tarefa. Ele não responde a pergunta “o que fazer?”, mas sim “como fazer”. Em termos mais técnicos, um algoritmo é uma sequência lógica, finita e definida de instruções que devem ser seguidas para resolver um problema ou executar uma tarefa*.”**

Trecho disponível em https://www.tecmundo.com.br/

programacao/2082-o-que-e-algoritmo-.html <Acesso em 14/10/2018>

Explicando em termos simples, um algoritmo pode ser comparado á uma receita de bolo. A receita de bolo te mostra o passo-a-passo para fazer e como ele deverá ficar. Em programação, o algoritmo será responsável pelas instruções do que o programa deverá fazer.

# **REFERENCIAL TEÓRICO**

Após a breve explicação sobre o conceito de ordenação e algoritmos, abaixo será apresentado os métodos existentes. Posteriormente será mostrada sua execução na prática:

# **Ordenação Bolha *(Bubble sort)***

O método de ordenação bolha ou *bubble sort*, é um dos métodos de ordenação mais simples inventados até o momento, pode ser aplicado em vetores e listas dinâmicas. A ideia principal deste algoritmo é percorrer o vetor diversas vezes e a cada volta levar para o topo do vetor o maior elemento do mesmo. Essa movimentação dos elementos lembra a forma que as bolhas procuram o seu próprio nível, com base nisto gerou-se o nome deste algoritmo.

No melhor dos casos este método executa **n** operações relevantes, sendo n o número total de elementos do vetor. No pior dos casos serão feitas **n²** operações. Este algoritmo não é recomentado para programas e aplicações que necessitem de velocidade ou que operem com uma quantidade elevada de dados.

Este algoritmo percorre a lista de itens do vetor do início ao fim, analisando a ordem dos elementos dois a dois e os trocando de lugar caso seja necessário. Ele irá percorrer a lista até que nenhum elemento tenha sido alterado de lugar na última passagem pelo vetor.

Abaixo temos um exemplo em pseudocódigo deste método:

**var x,y,aux : INTEIRO**

**PARA x=0 ATE vet.tamanho PASSO 1  
    PARA y=x+1 ATE vet.tamanho PASSO 1  
     SE (vet[x] > vet[y]) ENTAO  
        var aux = vet[x]  
       vet[x] = vet[y]**

**vet[y] = aux  
FIMSE**

**FIMPARA**

**FIMPARA**

Exemplo prático:

Ordenar o vetor **{8, 9, 3, 5, 1} em ordem crescente:**

**{8, 9, 3, 5, 1}:** 8 > 3 = Sim, realizar troca. Reordenando o vetor:

# **Busca Binária** *(Binary Search)*

A busca binária consiste em realizar divisões sucessivas de espaço de busca comparando um elemento buscado, no caso a chave, com algum elemento no meio do vetor. Se o elemento do meio do vetor por coincidência for a chave, a busca termina com sucesso.

Caso o elemento do meio não seja o elemento que está sendo buscado, a busca continua na metade posterior do vetor, e se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Abaixo segue um pseudocódigo do método de busca binária:

**BUSCA-BINÁRIA (V[ ], início, fim, e)**

**i recebe o índice do meio entre início e fim**

**se (v[i] = e) entao**

**devolva o índice i # elemento e encontrado**

**fimse**

**se (inicio = fim) entao**

**não encontrou o elemento procurado**

**senão**

**se (V[i] vem antes de e) então**

**faça a BUSCA-BINÁRIA (V, i+1, fim, e)**

**senão**

**faça a BUSCA-BINÁRIA (V, inicio, i-1, e)**

**fimse**

**fimse**

# **Terceira Ordenação**

# **DESENVOLVIMENTO**

# **RESULTADOS E DISCUSSÃO**

# **CONSIDERAÇÕES FINAIS**

# **REFERÊNCIAS BIBLIOGRÁFICAS**

<https://www.embarcados.com.br/algoritmos-de-ordenacao-bubble-sort/> <acessado em 31/10/2018>

<https://pt.wikipedia.org/wiki/Bubble_sort> <data de acesso: 29/10/2018>

<https://www.devmedia.com.br/entendendo-o-algoritmo-bubble-sort-em-java/24812> <data de acesso: 01/11/2018>

<https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao/> <data de acesso: 29/10/2018>

[www.forum.imasters.com.br](http://www.forum.imasters.com.br) <acessado em 21/10/2018>

[www.larback.com.br](http://www.larback.com.br) <acessado em 21/10/2018>

[www.facom.ufms.br](http://www.facom.ufms.br) <acessado em 21/10/2018>

[www.ic.unicamp.br](http://www.ic.unicamp.br) <acessado em 21/10/2018>

[www.stackoverflow.com](http://www.stackoverflow.com) <acessado em 21/10/2018>

# **CÓDIGO FONTE**

# **FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS**