

[Hora do Desafio]

[Desafios de Programação]

[Atividade avaliativa em times, vale 1.0 ponto]



[DESAFIO]

Análise comparativa
dos métodos de
ordenação.



Algoritmos:

Bubble Sort,
Insertion Sort,
Quick Sort,
Merge Sort,
Shell Sort,
Selection Sort,
Radix Sort

e mais um escolhido pelo
time.

Objetivo:

Classificar quanto ao
BigO, realizar
experimentos, analisar os
dados obtidos, comparar
com base em indicadores
e concluir sobre os
resultados apresentados
pelos métodos.

Trios:



Criar **nomes** para os times que tenham relação com os **métodos de ordenação**.

Desafio em trios.

Usar a prática do Pair Programming (adaptada).

[Explicação]

Prática
Pair Programming

do
eXtreme
Programming (XP)

Programação em pares (*Pair Programming*)

Duas pessoas implementam juntas o código diante do mesmo computador, revezando-se no teclado.



Uma pessoa é o "controlador", escreve o código, enquanto o outro, chamado de "observador" ("navegador"), analisa cada linha do código. Os papéis são revezados e, geralmente, tem-se um iniciante e um desenvolvedor mais experiente.



<https://youtu.be/a8WaP3Fwqa0>

Compartilhe o
ambiente de
programação!



Método e níveis de complexidade (big- e tiny-):



(1) **Classificar** os algoritmos quanto ao **Big-O**. [**Todos**]

(2) **Implementar** os algoritmos. [ : 8 algoritmos -  : 6 algoritmos]

(3) **Testar** cada algoritmo em **8 cenários** (aleatória pequena, crescente, decrescente, repetido, vazia, um item, muitos repetidos, longa). [ : automatizar -  : manual]

(4) **Realizar a análise comparativa**. Para a análise comparativa, utilizar os seguintes **indicadores**: número de **comparações**, **trocac**s e **tempos** realizadas pelos algoritmos. [**Todos**]

(4.1) As condições de entrada correspondem à geração de um vetor aleatório, que será submetido ao algoritmo para avaliação das seguintes **massas de dados**: **pequena** (1.000), **média** (10.000), **grande** (50.000) e **super grande** (100.000). É necessário que todos os algoritmos tenham a **mesma situação inicial**, ou seja, a mesma organização de dados (mesmo vetor). [ : até super grande -  : até grande]

(5) **Gerar gráficos ou tabela** contendo o número médio de comparações, trocas e tempos realizados para cada massa de dados. No **mínimo, três casos de testes para cada cenário**. [ : gráfico/tabela -  : tabela]

Requisitos:



Além dos gráficos/tabelas, defina os seguintes aspectos:

- Como foram **gerados** os vetores aleatórios?
- O que foi feito para que cada algoritmo ordenasse a **mesma condição** de vetor (mesma situação inicial)?
- Como foram adaptados os algoritmos para realizar a contagem do número de **comparações**, **trocas** (quantidade de vezes que uma dupla de valores trocou de posição no vetor) e **tempo** (informar como foi considerado o *clock* do processador e configurações da máquina onde a análise ocorreu)?
- Explicar os resultados dos experimentos com base na literatura, pautado em **referências**.
- Explicar **tabelas** e/ou **gráficos**.
- Incluir a classificação **Big-O**.

Observações:

Realizar **vários**
cenários.

Cenários de análise:

Para documentar os cenários de análise utilize a tabela comparativa para cada algoritmo (note que é necessário registrar três indicadores: comparações, trocas e tempo):



		Tempo											
MÉTODO	ALGORITMO	PEQUENO (1.000)			MÉDIO (10.000)			GRANDE (50.000)			SUPER GRANDE (100.000)		
MÉTODO	ALGORITMO 1	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3
		00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000	00:00:00:000
		Média dos Cenários 00:00:00:000			Média dos Cenários 00:00:00:000			Média dos Cenários 00:00:00:000			Média dos Cenários 00:00:00:000		
		Comparações											
MÉTODO	ALGORITMO	PEQUENO (1.000)			MÉDIO (10.000)			GRANDE (50.000)			SUPER GRANDE (100.000)		
MÉTODO	ALGORITMO 1	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3
		0	0	0	0	0	0	0	0	0	0	0	0
		Média dos Cenários 0			Média dos Cenários 0			Média dos Cenários 0			Média dos Cenários 0		
		Troca											
MÉTODO	ALGORITMO	PEQUENO (1.000)			MÉDIO (10.000)			GRANDE (50.000)			SUPER GRANDE (100.000)		
MÉTODO	ALGORITMO 1	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3	Cenário 1	Cenário 2	Cenário 3
		0	0	0	0	0	0	0	0	0	0	0	0
		Média dos Cenários 0			Média dos Cenários 0			Média dos Cenários 0			Média dos Cenários 0		

*realizar comparações para os algoritmos de ordenação.

*realizar comparações para os algoritmos de ordenação.

1. **Planejarás a solução...** esboçarás e discutirás com o par!
2. **Consultarás o par e os livros** (referências na sala virtual).
3. **Realizarás vários casos de testes.**
4. **Adicionarás lindos comentários** ao código-fonte.
5. **Utilizarás boas-práticas** de programação!
6. **Serás original e criativo** desenvolvendo a própria solução.

Regras gerais:



Observações:

Siga as regras... Elas foram definidas para que você e seu time desenvolvam um **processo construtivo e evolutivo** de sucesso!

Regra de ouro:

Crie funções pautado em referências!



O que temos que entregar?

Solução pela sala virtual + vídeo de **apresentação** + **discussão dos resultados** em aula.



Na Sala virtual:

Postar a **solução até a data informada na sala virtual às 19hs.**

Em aula pelo time (8 minutos):

- **Discussão dos resultados** (gráficos e/ou tabelas).
- **Demonstração da execução**, apresente um caso de teste.
- **Explicação de um método** utilizado na análise comparativa, algoritmo e adaptações necessárias para avaliar as comparações, trocas e o tempo.

Critérios de avaliação:

(0,2) **Pensamento computacional** (atende o que pede a questão, representa e manipula os algoritmos de forma adequada, apresenta domínio do processo de construção da solução. **Requisito: atender todas às regras e métodos apresentados.**

(0,2) **Correta execução do programa** (execução do programa sem erros, confiabilidade da resposta na execução de casos de teste e cenários) + **clareza e confiança dos resultados.**

(0,2) **Documentação do código-fonte** (identificação das funções e comentários nas principais linhas de comando - variáveis, regras e lógicas -, organização do código) + **documentação do método.**

(0,2) **Originalidade** (autoria do código-fonte pautado em referências, capacidade de pensar na solução do problema).

(0,2) **Discussão em aula + vídeo de apresentação** (clareza e profundidade - detalhes - na apresentação, cumprimento dos prazos).



Bom desafio!

[Não execute, pense antes!]



Principais referências para apoiar o desafio:

Sobre ordenação:

1. Oliveira, A. B.; Prada, A.; Silva, R. R. Métodos de Ordenação Interna. Florianópolis: Visual Books, 2002.
2. Ascencio, A. F. G; Araújo, G. S. Estrutura de dados: algoritmos, análise da complexidade e implementações em Java, C/C++. São Paulo: Pearson Prentice Hall, 2010 - Capítulo 2.
3. Simuladores:
VisuAlgo (Halim, Steven). Data Structure Visualizations (Galles, David). AlgoVis.io