Visual Sort - Aplicação educacional para aprendizado de Algoritmos de Ordenação

Dagson G. C. Araújo[1]

[1]Instituto Metrópole Digital – Universidade Federal do Rio Grande do Norte (UFRN)

dagson.cassiano.144@ufrn.edu.br

Resumo. *Neste relatório técnico, é descrito o desenvolvimento de um sistema em Java para a visualização de algoritmos de ordenação em funcionamento. Este sistema apresenta uma abordagem interativa que combina conceitos de orientação a objetos, estruturas de dados, e algoritmos fundamentais de ordenação. Este documento aborda as escolhas técnicas e arquiteturais do projeto, assim como uma análise das estruturas e algoritmos empregados.*

1. Introdução

A ordenação de dados é uma operação fundamental em Ciência da Computação, sendo amplamente utilizada em aplicações que variam de sistemas de bancos de dados a interfaces gráficas de usuário. A compreensão prática do funcionamento dos algoritmos de ordenação é essencial para estudantes e profissionais da área. Este trabalho propõe a construção de um sistema interativo em Java que permite visualizar, em tempo real, o funcionamento de diferentes algoritmos de ordenação.

No relatório, serão apresentados detalhes sobre o design do sistema, incluindo diagramas de classes e explicação das decisões de projeto. Também serão descritas as estruturas de dados e algoritmos utilizados, seguidas de uma análise dos resultados alcançados.

2. Justificativa

O ensino de algoritmos de ordenação é um componente essencial nos currículos de Ciência da Computação e áreas relacionadas. No entanto, muitos alunos encontram dificuldades em compreender o funcionamento interno desses algoritmos apenas a partir de explicações teóricas ou pseudocódigos.

Este projeto visa preencher a lacuna entre teoria e prática, utilizando tecnologia para melhorar o aprendizado e a experiência educacional.

3. Objetivo Geral

Desenvolver um sistema interativo em Java que permita a visualização em tempo real do funcionamento de algoritmos de ordenação, com o objetivo de facilitar o aprendizado e a compreensão desses algoritmos por parte de estudantes e entusiastas da Ciência da Computação.

4. Objetivos Específicos

- Implementar os algoritmos de ordenação Bubble Sort, Quick Sort, Merge Sort, Heap Sort, Bogo Sort, Radix Sort, Shell Sort, Insertion Sort e Selection Sort de forma modular e extensível.
- Criar uma interface gráfica intuitiva que demonstre visualmente o processo de ordenação, evidenciando as trocas.
- Oferecer suporte para a inclusão de novos algoritmos e modos de visualização sem alterar significativamente o núcleo do sistema.
- Documentar e disponibilizar o código como recurso didático para estudantes e educadores.

5. Metodologia

Para o desenvolvimento do sistema, foram seguidas as etapas abaixo:

- Planejamento:
    - Identificação do tema e dos requisitos funcionais e não funcionais.
    - Seleção dos algoritmos de ordenação a serem implementados.
    - Definição da arquitetura do sistema baseada em Orientação a Objetos.

- Desenvolvimento:
    - Implementação dos algoritmos de ordenação.
    - Construção da interface gráfica com a biblioteca Swing.
    - Aplicação de padrões de projeto para garantir flexibilidade e clareza.

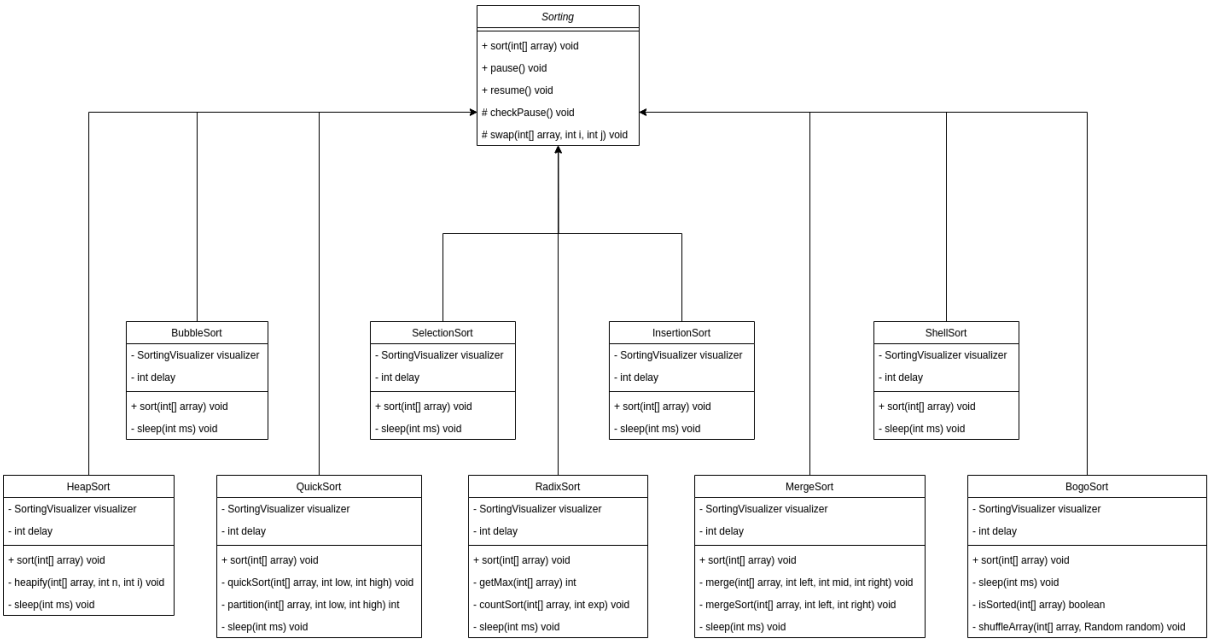
- Documentação:
    - Geração de documentação técnica do código.

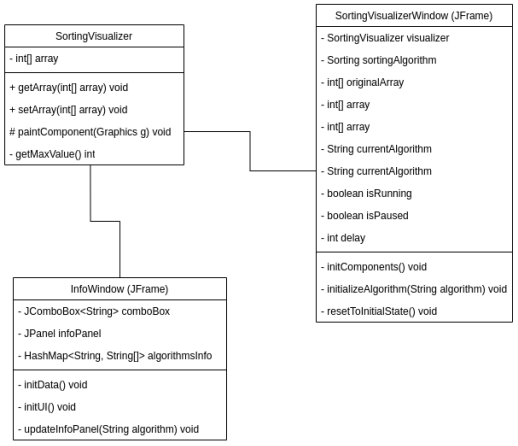
6. Descrição da abordagem de solução do problema

O sistema foi desenvolvido com base em um design modular, utilizando os princípios de orientação a objetos. A arquitetura principal inclui as seguintes classes:

- Main: Classe principal para inicialização do programa e interface com o usuário.
- SortingVisualizer: Gerencia a visualização gráfica e coordena a execução dos algoritmos de ordenação.
- InfoWindow: Utilizada para mostrar informações educacionais acerca dos algoritmos.
- Sorting: Classe abstrata que define a interface para algoritmos de ordenação.
- BubbleSort, MergeSort, etc.: Implementações específicas de algoritmos de ordenação.

6.1. Diagrama de Classes





6.1. Decisões de Projeto

- Uso do padrão Model-View-Controller para permitir a intercambialidade dos algoritmos de ordenação.
- Padrão Observer implementado para atualizar a interface gráfica à medida que os dados são ordenados.
- Modularidade no design para facilitar a adição de novos algoritmos e componentes visuais.

7. Descrição geral das estruturas de dados e algoritmos utilizados

7.1. Estruturas de Dados

- Array: Representa os elementos a serem ordenados.
- Thread: São utilizadas para executar os algoritmos de ordenação de forma assíncrona.
- Booleanos: Variáveis como isRunning e métodos como checkPause() em Sorting ajudam a gerenciar o fluxo de execução.
- Heap: Não é usado diretamente, mas sim conceitualmente no HeapSort.

7.2. Algoritmos Utilizados

- Bubble Sort
- Quick Sort
- Merge Sort
- Heap Sort
- Bogo Sort
- Radix Sort
- Shell Sort
- Insertion Sort
- Selection Sort

8. Conclusão

O sistema desenvolvido oferece uma plataforma prática para a compreensão de algoritmos de ordenação, permitindo a visualização de cada etapa do processo. A abordagem modular facilita sua expansão, possibilitando a inclusão de novos algoritmos ou

funcionalidades. Este projeto reforça a importância de integrar conceitos teóricos e práticos para melhorar a compreensão dos fundamentos de algoritmos.

9. Referências

Hackerearth. Visualizador de Algoritmos de Ordenação. Disponível em: https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/visualize/.

Sociedade Brasileira de Computação (SBC). Modelos para Publicação de Artigos. Disponível em: http://www.sbc.org.br/documentos-da-sbc/summary/.

Wikipédia. Algoritmos de Ordenação. Disponível em: https://pt.wikipedia.org/wiki/Algoritmo_de_ordena%C3%A7%C3%A3o.