UNIVERSIDADE/INSTITUTO: Campus Ulbra Torres-RS/Curso de Análise e Desenvolvimento de Sistemas
DISCIPLINA: Estruturas de Dados e Algoritmos
ALUNO(S): Guilherme Hugentobler Kross Pinho e Ruhan da Silva Bolzan
PROFESSOR: Juliano Ramos Matos

Relatório de Desenvolvimento – Manipulação de Matrizes Bidimensionais e Algoritmos de Ordenação

1. Introdução

O presente trabalho teve como objetivo desenvolver um sistema em linguagem Java capaz de manipular matrizes bidimensionais de inteiros e aplicar algoritmos de ordenação. O estudo permitiu a aplicação de conceitos de vetores, matrizes, loops e algoritmos de ordenação, reforçando a compreensão sobre estruturas de dados estáticas, recursão e complexidade algorítmica.

2. Descrição da Classe Matriz

2.1 Atributos

- **int[][] dados** → armazena os valores da matriz.
- **int linhas** → número de linhas da matriz.
- **int colunas** → número de colunas da matriz.

2.2 Métodos e responsabilidades

- **preencherManual()**
  Solicita que o usuário digite cada valor da matriz.
  Permite controlar exatamente os elementos inseridos.
- **preencherAutomatico()**
  Preenche a matriz com valores aleatórios entre 1 e 20.
  Facilita testes sem necessidade de digitar todos os elementos.
- **removerElemento(int linha, int coluna)**
  Substitui o valor da posição especificada por 0.
  Permite testar a manipulação de dados da matriz.
- **exibir()**
  Mostra a matriz no console em formato tabular, permitindo fácil visualização dos elementos.

- **Getters (getLinhas(), getColunas(), getDados())**
  Permitem acessar informações da matriz para uso externo, como nos métodos de ordenação.

3. Lógica das funcionalidades

- **Preenchimento manual:** utiliza dois loops aninhados (for) para percorrer todas as linhas e colunas da matriz, solicitando um valor para cada célula.

- **Preenchimento automático:** similar ao manual, mas gera números aleatórios usando a classe Random.

- **Remoção de elementos:** verifica se a posição indicada é válida e substitui o valor por 0.

- **Exibição:** percorre a matriz e imprime cada valor separado por tabulação (\t), linha por linha.

- **Ordenação:** aplicada de três formas:

    - **Por linhas:** cada linha da matriz é tratada como um vetor e ordenada com Bubble Sort ou Merge Sort.

    - **Por colunas:** cada coluna é percorrida manualmente e ordenada usando Bubble Sort ou Merge Sort.

    - **Matriz completa:** todos os elementos são colocados em um vetor, ordenados e reinseridos na matriz.

4. Algoritmos de Ordenação

4.1 Bubble Sort

- **Abordagem:** iterativa.

- **Complexidade:** $O(n^2)$ no pior caso.

- **Funcionamento:** compara elementos adjacentes e realiza trocas até que a sequência esteja ordenada.

- **Quando é eficiente:** para matrizes pequenas ou quase ordenadas.

4.2 Merge Sort

- **Abordagem:** recursiva.
- **Complexidade:** O(n log n) no pior caso, mais eficiente para grandes conjuntos de dados.
- **Funcionamento:** divide o problema em subvetores, ordena recursivamente e intercala os resultados para formar o vetor ordenado.
- **Quando é eficiente:** para matrizes grandes ou totalmente desordenadas.

4.3 Comparação teórica

- **Bubble Sort:** simples, fácil de implementar, mas lento para matrizes grandes.
- **Merge Sort:** mais rápido em geral, porém requer manipulação de vetores auxiliares e recursão.

O trabalho permitiu aplicar os dois algoritmos, comparando diretamente suas abordagens e compreendendo situações em que cada um é mais adequado.

5. Conclusões

Durante a realização do trabalho, foram observados os seguintes pontos:

- O desenvolvimento reforçou o entendimento de loops, vetores e matrizes, bem como a aplicação prática de algoritmos de ordenação.
- A implementação das diferentes formas de ordenação (linha, coluna, matriz completa) permitiu compreender a importância de organizar os dados antes de aplicar algoritmos.
- O uso de Bubble Sort e Merge Sort possibilitou observar as diferenças entre algoritmos iterativos e recursivos, além de compreender a relação entre complexidade e tamanho dos dados.
- O trabalho contribuiu para o aprendizado sobre modularização, boas práticas de programação e manipulação de estruturas estáticas em Java.