Exemplo 1

Estrutura da Classe

A classe Matriz é uma implementação de uma matriz bidimensional de inteiros. Ela inclui várias funções para manipular dados dentro dessa matriz, como inserção, preenchimento, remoção e exibição de valores. A classe é composta pelos seguintes elementos:

Atributo matriz: Um array bidimensional int [][] que armazena os valores da matriz.

Construtor Matriz: Esse construtor inicia a matriz com um número especificado de linhas e colunas.

Métodos de Manipulação e Operação:

preencherManual(): Preenche a matriz com valores fornecidos pelo usuário através do console.

preencherAleatorio(): Preenche a matriz com valores aleatórios entre 0 e 99.

inserirElemento(int linha, int coluna, int valor): Insere um elemento em uma posição específica da matriz.

removerElemento(int linha, int coluna): Remove um elemento de uma posição específica da matriz (define o valor como 0).

exibirMatriz(): Exibe a matriz no console.

Lógica utilizada nos métodos

Construtor Matriz

O construtor é utilizado para inicializar a matriz de inteiros (int[][]) com um número de linhas e colunas especificado pelo usuário, dentro dele criamos uma classe matriz para iniciar com o número de linhas e colunas

Método preencherManual()

Utiliza a classe Scanner para capturar entradas do usuário.

Os dois laços for são utilizados para iterar sobre cada elemento da matriz (primeiro iterando sobre as linhas e depois sobre as colunas).

O valor inserido pelo usuário é lido e atribuído à posição correspondente na matriz.

Método preencherAleatorio()

Ele utiliza a classe Random para gerar números aleatórios.

Usa dois laços for aninhados para iterar sobre cada posição da matriz.

Cada posição é preenchida com um número aleatório gerado entre 0 e 99.

Método inserirElemento(int linha, int coluna, int valor)

Primeiro ele verifica se os índices fornecidos (linha e coluna) estão dentro dos limites válidos da matriz, se a posição for válida, o valor fornecido é inserido na posição especificada, caso contrário, uma mensagem de erro é exibida.

Método exibirMatriz()
Utiliza um laço for-each para percorrer cada linha da matriz, dentro desse laço, outro laço for-each percorre cada elemento da linha, cada elemento é impresso seguido de um espaço ( ) para manter a formatação. Ao final de cada linha, System.out.println() é chamado para mover para a próxima linha.

Exemplo 2

Estrutura da Classe

A classe Matriz é uma representação de uma matriz bidimensional de números inteiros, com funcionalidades para preencher a matriz com valores aleatórios, exibir a matriz, ordenar linhas e colunas utilizando diferentes algoritmos de ordenação (Bubble Sort e Merge Sort), e ordenar a matriz inteira.

Atributo matriz: Um array bidimensional int [][] que armazena os valores da matriz.

Construtor Matriz: Construtor que inicializa a matriz com o número de linhas e colunas especificado pelo usuário. A matriz é criada utilizando um array bidimensional de inteiros.

Métodos:

preencherAleatorio(): Preenche a matriz com valores aleatórios entre 0 e 99.

exibirMatriz(): Exibe a matriz no console, imprimindo os elementos linha por linha.

ordenarLinhaBubble(int linha): Ordena uma linha específica da matriz utilizando o algoritmo Bubble Sort.

ordenarColunaBubble(int coluna): Ordena uma coluna específica da matriz utilizando o algoritmo Bubble Sort.

ordenarLinhaMerge(int linha): Ordena uma linha específica da matriz utilizando o algoritmo Merge Sort.

ordenarColunaMerge(int coluna): Ordena uma coluna específica da matriz utilizando o algoritmo Merge Sort.

ordenarMatrizCompleta(): Ordena toda a matriz como um único array, utilizando o algoritmo Merge Sort.

mergeSort(int[] array, int inicio, int fim): Implementação do algoritmo Merge Sort para ordenar um array de inteiros.

merge(int[] array, int inicio, int meio, int fim): Função auxiliar do Merge Sort que faz a intercalação de dois subarrays ordenados.

Lógica utilizada nos métodos

preencherAleatorio(): Percorre cada elemento da matriz utilizando dois loops aninhados. Assim para cada posição, ele atribui um valor aleatório entre 0 e 99, gerado por (int) (Math.random() * 100).

exibirMatriz(): Utiliza um loop for-each para percorrer cada linha e outro para percorrer cada elemento dentro da linha. Imprime os valores da matriz no console, separando cada elemento por um espaço.

ordenarLinhaBubble(int linha): Implementa o Bubble Sort para ordenar uma linha específica da matriz. Compara elementos e os troca de posição se estiverem na ordem incorreta, repetindo o processo até que a linha esteja ordenada.

ordenarColunaBubble(int coluna): Utiliza Bubble Sort para comparar elementos da coluna e os troca se necessário.

ordenarLinhaMerge(int linha): Aplica o algoritmo Merge Sort para ordenar uma linha específica. Esse método utiliza o método auxiliar mergeSort para realizar a ordenação.

ordenarColunaMerge(int coluna): Copia os elementos da coluna especificada para um array temporário. Aplica o Merge Sort no array temporário e, em seguida, copia os elementos ordenados de volta para a coluna original da matriz.

ordenarMatrizCompleta():

- Converte a matriz inteira em um vetor unidimensional, aplica o Merge Sort para ordená-lo, e depois distribui os valores ordenados de volta na estrutura bidimensional da matriz.

mergeSort(int[] array, int inicio, int fim):

- Implementação clássica do Merge Sort. O array é dividido recursivamente até que cada parte tenha um único elemento ou esteja vazia. Depois, é feita a combinação dos arrays menores de forma ordenada.

merge(int[] array, int inicio, int meio, int fim):

- Função de intercalação que combina dois subarrays ordenados em um único array ordenado. Utiliza arrays temporários L (esquerda) e R (direita) para manter os elementos e então os insere de volta no array original de forma ordenada.

Diferenças e para quais casos são eficientes

O Bubble Sort é considerado complexo assim sendo ineficiente em listas grandes enquanto o Merge Sort muito mais eficiente para listas grandes.

Ambos os algoritmos são estáveis. Isso significa que eles mantêm a ordem relativa dos elementos iguais na matriz

Bubble Sort é um dos algoritmos de ordenação mais simples de implementar e entender, enquanto o Merge Sort é muito mais complexo devido o uso de recursão e intercalação.

Bubble sort: É mais eficiente para listas pequenas ou quase ordenadas. Também pode ser vantajoso quando há necessidade de um algoritmo in-place com uma implementação simples.

Merge sort: É mais eficiente para listas grandes devido à sua complexidade de tempo. É ideal para listas que não cabem na memória (ordenando dados externos) por conta da sua abordagem de divisão e intercalação.

Quando o custo de complexidade de tempo é um fator importante, o Merge Sort se sobressai em relação ao Bubble Sort, apesar do custo adicional de espaço.