Exercícios de Implementação - Quicksort e Heapsort

Oficina de AEDs2

Exercício 1: Implementação do Quicksort com Pivô Central

Contexto

Você está desenvolvendo um sistema de gerenciamento de produtos para uma loja virtual. O sistema precisa ordenar rapidamente listas de preços de produtos para diversas funcionalidades como exibição ordenada, busca binária e análise de preços.

Objetivo

Implementar o algoritmo **Quicksort** utilizando a estratégia de seleção de pivô como o **elemento central** do array.

Requisitos Técnicos

1. Implemente a função `quickSort(int[] arr, int low, int high)`
2. Implemente a função `partition(int[] arr, int low, int high)` que:
   - Seleciona o pivô como o elemento do meio: `mid = low + (high - low) / 2`
   - Realiza a partição do array utilizando o pivô
   - Retorna o índice final do pivô
3. O algoritmo deve ordenar **in-place**

4. Deve lidar com arrays de qualquer tamanho

Código Base

```
public class QuickSort {
    public static void quickSort(int[] arr, int low, int high) {
        // SUA IMPLEMENTAÇÃO AQUI
    }

    public static int partition(int[] arr, int low, int high) {
        // SUA IMPLEMENTAÇÃO AQUI
        return -1;
    }
}
```

Teste Esperado

Para o array de entrada: `[89, 45, 120, 23, 67, 199, 34]`
A saída deve ser: `[23, 34, 45, 67, 89, 120, 199]`

Desafio Adicional (Opcional)

Adicione contadores para medir:

- Número total de comparações realizadas
- Número total de trocas (swaps) realizadas

Exercício 2: Implementação do Heapsort para Sistema de Pedidos

Contexto

Você está desenvolvendo um sistema de gerenciamento de pedidos para um e-commerce. O sistema precisa processar pedidos por ordem de prioridade, sendo que pedidos com maior prioridade devem ser atendidos primeiro.

Objetivo

Implementar o algoritmo **Heapsort** para ordenar um array de objetos `Pedido` baseado no campo de prioridade.

Requisitos Técnicos

1. Utilize a classe `Pedido` com os campos:
    - `int id`
    - `int prioridade`
    - `double valor`
2. Implemente a função `heapSort(Pedido[] arr)`
3. Implemente a função `heapify(Pedido[] arr, int n, int i)`
4. O algoritmo deve ordenar em ordem **decrescente** de prioridade

Código Base

```
class Pedido {
    public int id;
    public int prioridade;
    public double valor;

    public Pedido(int id, int prioridade, double valor) {
        this.id = id;
        this.prioridade = prioridade;
        this.valor = valor;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Prioridade: " + prioridade + ", Valor: " + valor;
    }
}

public class HeapSort {
    public static void heapSort(Pedido[] arr) {
        // SUA IMPLEMENTAÇÃO AQUI
    }

    public static void heapify(Pedido[] arr, int n, int i) {
        // n: número de elementos no array
        // i: número a ser ''heapficado'' (colocado na posição correta)
        // SUA IMPLEMENTAÇÃO AQUI
    }
}
```

Teste Esperado

Para os pedidos de entrada:

- Pedido(101, 3, 150.00)
- Pedido(102, 1, 75.50)
- Pedido(103, 5, 299.90)

- Pedido(104, 2, 45.00)
- Pedido(105, 4, 120.00)

A saída ordenada por prioridade deve ser:

- Pedido(103, 5, 299.90)
- Pedido(105, 4, 120.00)
- Pedido(101, 3, 150.00)
- Pedido(104, 2, 45.00)
- Pedido(102, 1, 75.50)

Desafio Adicional (Opcional)

Modifique o algoritmo para:

- Imprimir o número total de *swaps* realizados
- Implementar ordenação crescente e decrescente