Revisão Prática de Estrutura de Dados

Os exercícios devem ser implementados em C++.

Exercício 1: Implementação de Busca em Largura (BFS) em Grafo

Objetivo: Implementar o algoritmo de Busca em Largura (BFS) para encontrar o caminho mais curto entre dois vértices em um grafo não direcionado.

Instruções:

1. Crie um grafo não direcionado utilizando a representação de **Lista de Adjacência**.
2. O grafo deve ter no mínimo 6 vértices e 8 arestas.
3. Implemente a função `bfs(Grafo* grafo, int verticeInicial, int verticeDestino)` que retorne o caminho mais curto (sequência de vértices) do `verticeInicial` ao `verticeDestino`.
4. Se não houver caminho, a função deve indicar isso.
5. No `main`, crie o grafo, chame a função `bfs` para dois pares de vértices diferentes e imprima os resultados.

Exemplo de Saída Esperada (para caminho de 0 a 3):

Caminho mais curto de 0 para 3: 0 -> 1 -> 3

Dica para prova: Tenha a implementação base dos algoritmos de busca em profundidade, busca em largura e dijkstra.

Exercício 2: Implementação e Comparação de Algoritmos de Ordenação

Objetivo: Implementar e comparar o desempenho de dois algoritmos de ordenação: Selection Sort e Merge Sort.

Instruções:

1. Implemente as funções `selectionSort(int arr[], int n)` e `mergeSort(int arr[], int l, int r)`.
2. Crie um array de inteiros com 1000 elementos aleatórios (entre 1 e 10000).
3. Meça o tempo de execução de cada algoritmo para ordenar o array. Para isso, você pode usar a biblioteca `<chrono>`.
4. Imprima o array original (apenas os primeiros 20 elementos para não poluir a saída), o array ordenado por cada método (também os primeiros 20 elementos) e o tempo de execução de cada um.
5. Comente no código sobre a complexidade teórica de cada algoritmo e o que você observou na prática.

Exemplo de Saída Esperada:

Array Original (primeiros 20): [..., ..., ...]

Selection Sort:
Array Ordenado (primeiros 20): [..., ..., ...]
Tempo de execução: X ms

Merge Sort:
Array Ordenado (primeiros 20): [..., ..., ...]
Tempo de execução: Y ms

Dica para a prova: Tenha a implementação de todos os algoritmos de ordenação vistos em aula.

Exercício 3: Implementação de Tabela Hash (Sei que parece difícil, mas não é)

Objetivo: Implementar uma Tabela Hash simples utilizando qualquer função hash, pode pesquisar para fazer essa, mas entenda o código.

Pode ser qualquer implementação, serve como base para a prova.