Izabela Cecilia Silva Barbosa

Exercício 1

1. **Responda de maneira breve, mas completamente, às questões abaixo. Dê um exemplo em cada resposta:**
    a. É aquele que produz o resultado correto para um determinado problema/tarefa, de forma precisa e confiável, sem se preocupar com eficiência máxima e desempenho.
    Ex: Algoritmo de busca linear que encontra um elemento x em uma lista.
2. **O que é um algoritmo eficiente?**
    a. É aquele que utiliza recursos de forma otimizada para resolver um problema, minimizando custos.
    Ex: Algoritmo QuickSort que tem tempo médio de O(n log n)
3. **O que é o custo de um algoritmo?**
    a. Refere-se à quantidade de recursos computacionais necessários para executar um algoritmo.
    Ex: Algoritmo de ordenação que requer grandes números de comparações e movimentações de elementos terá um alto custo computacional.

Exercício 2

2) Escreva um estudo simples acerca da complexidade dos métodos de ordenação de dados em memória
principal listados abaixo. Para cada item da questão, escolha um dos algoritmos propostos. Sua resposta deve
conter (i) um resumo/explicação de como o método resolve o problema (não envolvendo linhas de código) e
(ii) considerações sobre a complexidade dos métodos e seus casos de execução.

Bubble Sort

O algoritmo Bubble Sort é uma técnica de ordenação simples de se entender, imagine que você tem uma lista de números desordenados, o bubble sort funciona comparando os números dois a dois, começando do início da lista. Se o número atual for maior do que o próximo, os dois números são trocados de posição. Esse processo é repetido até que todos os números estejam em ordem.
Quanto a sua complexidade, ele possui uma complexidade de tempo quadrática O(n^2). Isso significa que, conforme o tamanho da lista aumenta, o tempo necessário para ordená-la aumenta quadril lateralmente. Essa complexidade ocorre porque o Bubble Sort percorre repetidamente toda a lista, comparando e trocando elementos até que a lista esteja completamente ordenada. Mesmo em listas parcialmente ordenadas, o Bubble Sort ainda precisará percorrer todos os elementos para garantir que a lista esteja completamente ordenada. Quando aplicado em uma lista ordenada, sua complexidade será de O(n), pois

irá apenas percorrer a lista verificando os valores, sem fazer troca, sendo seu melhor caso. O seu pior caso acontece quando a lista está em ordem reversa, sendo O(n^2).

Merge Sort

O algoritmo Merge Sort é uma técnica de ordenação de divisão e conquista. Ele divide uma lista de elementos em sub-listas menores, ordena cada sub-lista individualmente e, em seguida, combina as sub-listas ordenadas para obter uma lista final ordenada. Esse processo de divisão e mesclagem é repetido recursivamente até que a lista esteja completamente ordenada.
Quanto a sua complexidade, ele possui uma complexidade de tempo O(n log n). Isso é porque ele divide a lista em duas metades e, sem seguida, mescla essas metades ordenadas. Como a operação de mesclagem leva um tempo proporcional ao tamanho total das das listas, a complexidade do mergesort é determinada pela recursão em log n divisões da lista e a operação de mesclagem linear. O seu melhor, médio e pior caso mantém a complexidade de O(n log n), tornando ele uma boa opção para lidar com dados grandes.

Fontes:

Sorting Algorithms: The Difference Between Bubble Sort and Merge Sort | by AnnMargaret Tutu | Medium
Quicksort vs. Mergesort | Baeldung on Computer Science
python - Calculo de complexidade BubbleSort - Stack Overflow em Português
Merge Sort (blogcyberini.com)