UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

INSTITUTO METRÓPOLE DIGITAL

PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA DA INFORMAÇÃO

DISCIPLINA: Estrutura de Dados e Algoritmos **PROFESSOR:** Itamir Morais

NOME COMPLETO: Maria Eduarda Fernandes Lago **MATRÍCULA:** 20241023191

TÍTULO: Fichamento dos Capítulos 1, 2 e 3 do livro *Introdução a Algoritmos* (Thomas H. Cormen et al.)

1. Capítulo 1: O Papel do Algoritmo na computação

O capítulo inicial do livro trata da definição e da importância dos algoritmos no contexto da computação. Um algoritmo é descrito como uma sequência finita de instruções bem definidas, utilizadas para transformar uma entrada de dados em uma saída esperada. A obra demonstra que algoritmos não se limitam ao universo computacional, trazendo exemplos de aplicação em áreas como biotecnologia e segurança digital.

Um dos exemplos apresentados é o Projeto Genoma Humano, que envolveu a determinação da sequência de três bilhões de pares de bases do DNA humano. O tratamento, o armazenamento e a análise desses dados exigem algoritmos sofisticados, demonstrando a interdisciplinaridade entre biologia e ciência da computação. Outro exemplo é o comércio eletrônico, que requer mecanismos de segurança, como criptografia e assinaturas digitais, baseados em algoritmos numéricos e na teoria dos números.

O texto destaca que, mesmo com o avanço dos recursos de hardware, a eficiência algorítmica continua sendo um fator crítico. Algoritmos ineficientes podem comprometer seriamente o desempenho de sistemas, especialmente diante de grandes volumes de dados. Por isso, algoritmos são considerados uma tecnologia tão relevante quanto os próprios equipamentos computacionais.

O autor enfatiza que algoritmos estão no núcleo da maioria das tecnologias computacionais modernas. Dominar algoritmos não é apenas saber programar, mas compreender as limitações de tempo e espaço e ser capaz de selecionar a solução mais adequada para cada problema. Nesse sentido, uma sólida formação em algoritmos é vista como um diferencial que separa programadores qualificados de iniciantes.

Esses conceitos fundamentais servem de base para os capítulos seguintes, onde os algoritmos começam a ser analisados em termos de desempenho e aplicabilidade.

2. Capítulo 2: Estatística de Classificação e Ordem

O capítulo inicia-se com a diferenciação entre pseudocódigo e código real. O pseudocódigo é apresentado como uma forma de descrever algoritmos utilizando qualquer método expressivo que torne sua lógica mais clara e concisa para a leitura humana, sem a rigidez sintática de uma linguagem de programação formal. Já o código real corresponde a um conjunto de instruções escritas em uma linguagem compilável ou interpretável por computadores.

Após essa distinção conceitual, o capítulo introduz o problema de ordenação por inserção, utilizando como analogia a organização de cartas em um jogo. Nesse processo, os elementos são retirados um a um e inseridos na posição correta dentro da parte já ordenada, simulando a forma como uma pessoa organizaria cartas em mãos. O algoritmo se mostra uma opção eficiente para conjuntos de dados pequenos, sendo valorizado por sua simplicidade, facilidade de implementação e bom desempenho em entradas reduzidas — embora não seja recomendado para grandes volumes de dados, dada sua complexidade quadrática.

O capítulo também introduz os fundamentos da análise de algoritmos, apresentando o modelo de máquina RAM como base para estimar o tempo de execução dos algoritmos. A análise considera que cada instrução do pseudocódigo tem um custo constante, e o tempo total é obtido a partir do número de execuções de cada linha. A eficiência do algoritmo é então descrita como uma função do tamanho da entrada, que pode variar conforme o tipo de problema e a estrutura dos dados fornecidos.

O algoritmo de ordenação por inserção, por exemplo, apresenta desempenho linear no melhor caso — quando a entrada está ordenada — e desempenho quadrático no pior caso — quando a entrada está em ordem inversa. Essas análises reforçam a importância de escolher algoritmos adequados ao contexto, levando em conta não apenas o tamanho, mas também as características da entrada.

Por fim, o autor destaca que a análise algorítmica pode se tornar mais complexa em cenários futuros, como nos algoritmos aleatorizados, onde o comportamento pode variar para uma mesma entrada.

O capítulo 2 também apresenta o paradigma de **Divisão e Conquista**, uma estratégia amplamente utilizada no projeto de algoritmos eficientes. Ao contrário da abordagem incremental adotada na ordenação por inserção, esse paradigma propõe dividir o problema original em subproblemas menores, resolvê-los recursivamente e, por fim, combinar as soluções parciais para obter a resposta final.

Esse processo segue três etapas principais:

1. **Divisão** do problema em subproblemas menores (geralmente de mesmo tipo);
2. **Conquista**, resolvendo os subproblemas recursivamente (ou diretamente, caso sejam suficientemente pequenos);
3. **Combinação** das soluções para formar a resposta final ao problema original.

A Ordenação por Intercalação é apresentado como um exemplo clássico dessa abordagem. O algoritmo divide a entrada em duas partes iguais, ordena cada uma recursivamente, e as mescla com eficiência por meio do procedimento auxiliar Merge. Esse procedimento funciona como uma simulação de duas pilhas de cartas ordenadas, nas quais escolhe-se a menor das cartas superiores para formar uma nova pilha ordenada. Para evitar verificações repetitivas, são utilizadas as sentinelas, o que simplifica a implementação sem comprometer a

eficiência. A complexidade da intercalação é Θ(n), onde *n* é o número de elementos sendo mesclados. Já o algoritmo de Ordenação por Intercalação completo é analisado por meio de uma recorrência, um modelo matemático que descreve o tempo de execução de algoritmos recursivos.
Além disso, o autor destaca que esse tipo de análise é essencial para prever o desempenho de algoritmos recursivos e será aprofundado nos capítulos seguintes. Essa formalização permite comparar algoritmos de forma objetiva e embasada.

3. **Capítulo 3: Caracterizando os Tempos de Execução**

O Capítulo 3 aprofunda a análise da eficiência dos algoritmos considerando o comportamento de seu tempo de execução à medida que o tamanho da entrada aumenta. O objetivo é entender quais algoritmos continuam eficazes mesmo quando aplicados a volumes de dados muito grandes. Para isso, o capítulo introduz o conceito de eficiência assintótica, que permite abstrair detalhes como constantes e termos de menor ordem e concentrar-se apenas no ritmo de crescimento da função que representa o tempo de execução.

Embora utilize notações matemáticas específicas, como Θ, O e Ω, o mais importante é compreender que elas são maneiras de classificar algoritmos conforme a velocidade com que seu tempo de execução cresce. Essas notações permitem comparar algoritmos de forma objetiva, identificando quais são mais eficientes para grandes volumes de dados. Por exemplo, um algoritmo que cresce proporcionalmente a n log n tende a ser mais eficiente do que outro que cresce como n^2, mesmo que para entradas pequenas o segundo pareça mais rápido.

Além disso, o capítulo mostra que as diferenças entre melhores, piores e médios casos de execução devem ser analisadas com clareza, e que o uso correto dessas notações evita generalizações enganosas. Também são discutidas algumas propriedades das comparações entre funções que ajudam a manter consistência nas análises.

Portanto, mais do que decorar fórmulas ou símbolos, este capítulo convida o leitor a refletir sobre o comportamento dos algoritmos em diferentes cenários, e fornece uma base conceitual para decidir, de forma fundamentada, qual abordagem será mais adequada dependendo da situação prática. Dessa forma, a análise assintótica se torna uma ferramenta indispensável para qualquer programador que deseje tomar decisões eficientes em projetos reais.