Análise e Complexidade de Algoritmo

Ellen Cristyne Melo RibeiroVitorino
Centro Universitário Euroamericano (UNIEURO)
Brasília, Brasil
ellen.vitorino@hortmail.com

Gustavo Barboza
Centro Universitário Euroamericano (UNIEURO)
Brasília, Brasil
gustabarboza37@gmail.com

Felipe Silva Santiago
Centro Universitário Euroamericano (UNIEURO)
Brasília, Brasil
felipesantiagobastos@gmail.com

Letícia de Oliveira Barros
Centro Universitário Euroamericano (UNIEURO)
Brasília, Brasil
leticiadeoliveirabarros12@gmail.com

Resumo — Este artigo apresenta uma introdução à análise de complexidade de algoritmos, destacando sua importância na avaliação de desempenho computacional. São abordados conceitos fundamentais como notação assintótica, variações de algoritmos e aplicações práticas. O estudo reforça a relevância da escolha adequada de algoritmos para garantir soluções eficientes.

Palavras-chave: Análise de algoritmos; Complexidade computacional; Notação Big O; Eficiência de código; Estrutura de repetição; Desempenho de algoritmos.

I. INTRODUÇÃO

Este artigo tem como objetivo abordar a Análise de Complexidade de Algoritmos, área fundamental da ciência da computação que avalia o desempenho de algoritmos em termos de tempo de execução e uso de memória, de acordo com o tamanho da entrada.

Entender essa análise é essencial porque, embora um problema possa ser teoricamente resolvido por um computador, isso não é considerado tratável quando existe um algoritmo que o resolve de forma eficiente mesmo no pior caso [SZW][1].

A escolha de algoritmos eficientes impacta diretamente o desempenho de sistemas reais. Em aplicações como bancos de dados, sistemas operacionais e plataformas web, algoritmos de busca e ordenação são constantemente utilizados, e uma escolha inadequada pode comprometer seriamente a performance, especialmente diante de grandes volumes de dados.

Dessa forma, compreender e aplicar a análise de complexidade é crucial não apenas para obter resultados corretos, mas para garantir soluções otimizadas e escaláveis.

II. OBJETIVOS

O objetivo deste artigo é aprofundar o estudo sobre análise de complexidade de algoritmos, dando continuidade ao projeto desenvolvido na disciplina. A proposta é analisar o desempenho de diferentes métodos de ordenação já estudados, levando em conta o tempo de execução e o uso de memória em diferentes situações: caso melhor, médio e pior.

Também se busca compreender como a notação assintótica, especialmente a Big O, permite avaliar e comparar a eficiência dos algoritmos, independentemente do ambiente em que são executados. O trabalho pretende destacar em quais contextos cada método se mostra mais eficiente, considerando o uso de recursos e o comportamento diante de grandes volumes de dados. Dessa forma, reforça-se a importância da escolha adequada de algoritmos para garantir soluções mais práticas e escaláveis.

III. CONCEITOS

A análise de complexidade de algoritmos busca avaliar o desempenho de uma solução computacional com base no número de operações realizadas e na quantidade de memória utilizada, considerando o crescimento do tamanho da entrada (n). Como destaca Silva[2] (2020), simplesmente medir o tempo de execução de um algoritmo em um computador específico não é suficiente, pois diferentes máquinas podem apresentar variações. Por isso, utiliza-se a análise assintótica, que abstrai fatores externos e foca no comportamento do algoritmo em função de *n*.

Para calcular a complexidade de um algoritmo, analisamos quantas vezes cada instrução é executada conforme o tamanho da entrada (n) [3]. Por exemplo, se um algoritmo possui um único comando que é executado uma vez, sua complexidade é **O(1)**. Se ele percorre todos os elementos de uma lista com n itens, o número de operações cresce proporcionalmente a n, resultando em **O(n)**. Já se houver dois loops aninhados, cada um percorrendo n elementos, o total de operações será aproximadamente n × n, ou seja, **O(n²)**. Essa contagem nos permite criar uma função T(n), que representa o tempo de execução em função de n, e a notação Big O nos ajuda a expressar o comportamento dessa função de forma simplificada, ignorando constantes e termos menos relevantes.

Essa análise é feita por meio de funções de complexidade, que descrevem o número de operações necessárias para resolver um problema. Conforme ilustrado na **Tabela 1**, essas funções variam de acordo com o tipo de

[1] SZWARCFITER, J. L. Gráfos e Algoritmos computacionais. Rio de Janeiro: Campus, 1984.

[2] SILVA, Walmir. *Análise de Complexidade de Algoritmos*. GrowthCode, 8 ago. 2020.

algoritmo e o crescimento esperado de seu tempo de execução ou uso de memória, de acordo com o tamanho da entrada.

Notação	Tipo de Complexidade	Descrição
O (1)	Constante	O tempo de execução é fixo, independentemente do tamanho da entrada.
O (n)	Linear	O tempo cresce proporcionalmente à entrada.
O (n^2)	Quadrática	O tempo cresce rapidamente; típico em algoritmos de força bruta

Tabela 1 – Classificação simplificada de complexidades assintóticas.

Conforme ilustrado na **Tabela 1**, as funções de complexidade apresentam comportamentos distintos de crescimento, que podem ser visualmente representados no gráfico a seguir. Esse tipo de visualização permite observar com clareza a diferença entre algoritmos altamente eficientes e aqueles que se tornam inviáveis com grandes volumes de dados.

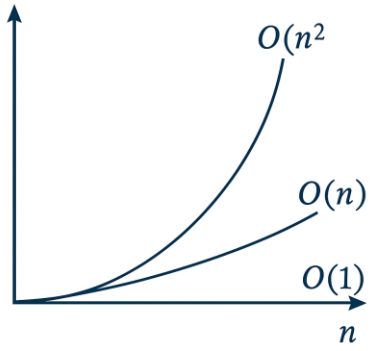

Figura 1 – Crescimento das funções.

Na Figura 1, é possível notar que O (1) permanece constante independentemente da entrada, O(n) cresce de forma linear, e O(n^2) apresenta crescimento acelerado, típico de algoritmos com estruturas de repetição aninhadas.

IV. Variações / Algoritmos

Como sabemos, existem milhares de variações de algoritmos. Como um mesmo problema computacional pode ser resolvido de múltiplas formas. Cada algoritmo é uma "receita" de passos, e as variações são apenas diferentes versões dessa receita, cada uma com suas vantagens e desvantagens.

- Eficiência (Tempo e Espaço): Algoritmos podem ser mais rápidos (levando menos tempo para executar) ou mais "leves" (usando menos memória). Um exemplo clássico é na ordenação de dados: alguns são rápidos, mas usam mais memória, outros são mais lentos, mas conservam recursos.
- **Simplicidade**: Às vezes, a prioridade é um algoritmo que seja fácil de entender e implementar, mesmo que não seja o mais veloz, pois a clareza do código reduz a probabilidade de erros e facilita a manutenção futura[3].
- Adaptação a Casos Específicos: Há variações criadas para lidar melhor com certas condições. Por exemplo, na busca por um item, se a lista já estiver organizada, há algoritmos que encontram o item muito mais rápido do que se ela estivesse desorganizada.

V. Aplicações Práticas

A análise e complexidade de algoritmos constitui uma abordagem fundamental para estimar os recursos computacionais (como tempo de execução e uso de memória) necessários à resolução de um problema. Essa estimativa permite não apenas a escolha do algoritmo mais eficiente, mas também a otimização de sistemas computacionais em diversas áreas da ciência e da tecnologia.

Abaixo, alguns exemplos práticos relevantes do uso de algoritmos e análise de complexidade:

- Plataformas de redes sociais: fazem uso intensivo de algoritmos sofisticados em mecanismos de busca, personalização de conteúdo e sistemas de recomendação, com o objetivo de otimizar a experiência do usuário e aumentar o engajamento.
- Treinamento de modelos de *machine learning*: exige o processamento de grandes volumes de dados para identificar padrões e realizar inferências. Nesse contexto, a notação Big O é amplamente utilizada para descrever o comportamento assintótico dos algoritmos, permitindo a avaliação de sua escalabilidade e desempenho conforme o aumento do tamanho da entrada.
- Algoritmos de criptografia: são empregados para proteger informações sensíveis, como senhas e dados bancários. A análise de complexidade é essencial para garantir que esses algoritmos sejam robustos o suficiente para resistir a ataques computacionais, assegurando a confidencialidade e a segurança dos dados.
- **Inteligência artificial (IA)**: técnicas como busca heurística, árvores de decisão e redes neurais dependem de algoritmos com boa performance para garantir respostas rápidas e eficazes. A análise de complexidade ajuda a selecionar estruturas e estratégias adequadas para tarefas como reconhecimento de fala, tomada de decisão e jogos.

[3] [6] MARTIN, Robert C. Código Limpo: Habilidades Práticas do Agile Software. Alta Books, 2009.

Assim, a análise de complexidade se mostra essencial na escolha e otimização de algoritmos, contribuindo diretamente para o desenvolvimento de sistemas mais eficientes, seguros e escaláveis em diversas áreas da computação.

VI. Avanços / Otimizações

Com o avanço da computação, várias otimizações foram aplicadas aos métodos de ordenação abordados no projeto para melhorar o desempenho prático, mesmo quando a complexidade teórica se mantém a mesma. [4]

Entre essas melhorias, estão estratégias que envolvem a escolha mais adequada de elementos para dividir os dados e o uso combinado de diferentes abordagens em partes específicas da entrada. Essas mudanças ajudam a reduzir o tempo de execução e tornam os algoritmos mais eficientes na prática. Além disso, ajustes específicos foram pensados para situações em que se utiliza memória limitada ou estruturas como listas encadeadas, diminuindo o número de operações desnecessárias.

Essas otimizações mostram que entender a complexidade teórica é importante, mas também é essencial saber adaptar os algoritmos para o dia a dia. Isso contribui para desenvolver sistemas mais eficientes e preparados para lidar com os desafios da computação atual.

VII. Conclusão

A análise de complexidade de algoritmos é fundamental para avaliar o desempenho de soluções computacionais em termos de tempo de execução e uso de memória, conforme o tamanho da entrada. Essa avaliação é essencial para garantir a eficiência de sistemas reais, especialmente diante de grandes volumes de dados. O estudo destaca a importância da análise assintótica, que utiliza notações como a Big O para descrever o comportamento do algoritmo independentemente de fatores externos, focando no crescimento do número de operações. Compreender e aplicar a análise de complexidade é crucial para desenvolver soluções otimizadas e escaláveis em diversas áreas da computação.

Referências

[1] SZWARCFITER, J. L. Gráfos e Algoritmos computacionais. Rio de Janeiro: Campus, 1984

[2] SILVA, Walmir. *Análise de Complexidade de Algoritmos*. GrowthCode, 8 ago. 2020.

[3] MARTIN, Robert C. Código Limpo: Habilidades Práticas do Agile Software. Alta Books, 2009.

[4] Inspirado em TOSCANI, Laira V.; VELOSO, Paulo A. S. *Uma metodologia para cálculo da complexidade de algoritmos*. Anais do IV SBES, Águas de São Pedro, 1990.

[4] Inspirado em TOSCANI, Laira V.; VELOSO, Paulo A. S. *Uma metodologia para cálculo da complexidade de algoritmos*. Anais do IV SBES, Águas de São Pedro, 1990.