CENTRO UNIVERSITÁRIO HERMINIO OMETTO
UNIARARAS

GUILHERME CAVENAGHI

Avaliação Comparativa de Algoritmos em Diferentes Linguagens de Programação: Impacto no Desempenho e Eficiência Computacional

Projeto de Trabalho de Conclusão de Curso apresentado ao curso de Engenharia da Computação da UniAraras, como parte dos requisitos para obtenção do grau de Bacharel.

Orientador: Prof. Renato Luciano Cagnin

ARARAS
2025

Resumo

Este trabalho tem como objetivo realizar uma análise comparativa de desempenho de algoritmos clássicos implementados em diferentes linguagens de programação. Com base nas linguagens mais utilizadas na indústria e na comunidade acadêmica, serão avaliadas métricas como tempo de execução, uso de memória e complexidade de implementação. A proposta visa identificar como a escolha da linguagem pode influenciar na eficiência computacional de uma aplicação.

Palavras-chave: algoritmos, linguagens de programação, desempenho, eficiência computacional.

1 Introdução

O desenvolvimento de software eficiente depende de diversos fatores, e entre eles destaca-se a linguagem de programação utilizada. Embora a lógica dos algoritmos possa ser universal, a forma como eles são implementados e executados varia significativamente entre as linguagens. Neste contexto, torna-se relevante compreender de que forma essa escolha impacta o desempenho e a eficiência computacional.

O conceito central deste trabalho é a análise comparativa entre diferentes linguagens, com foco em algoritmos clássicos. Parte-se do princípio de que, embora o algoritmo em si permaneça constante, o ambiente de execução, o modelo de compilação e a forma de gerenciamento de memória diferem entre linguagens, afetando diretamente os resultados. O problema (GAP) reside na carência de estudos empíricos que abordem essa comparação de maneira prática, considerando dados objetivos como tempo e memória.

A justificativa para esta pesquisa se baseia na crescente demanda por aplicações de alto desempenho e na necessidade de tomar decisões mais embasadas na escolha da linguagem, tanto na academia quanto na indústria. Entender esses impactos pode auxiliar desenvolvedores a optarem por ferramentas mais adequadas ao seu contexto de projeto.

A motivação deste trabalho surge da curiosidade em saber se determinadas linguagens, vistas como mais modernas ou mais populares, realmente oferecem vantagens práticas em termos de desempenho, ou se essa percepção está mais ligada a fatores culturais e de mercado do que técnicos.

Os objetivos desta pesquisa incluem: avaliar o desempenho de algoritmos clássicos implementados em diferentes linguagens; identificar os fatores técnicos que afetam esse desempenho; e oferecer uma síntese clara de quais linguagens se destacam em cada aspecto analisado.

Por fim, será apresentado um resumo dos resultados obtidos com os testes comparativos, destacando os pontos fortes e fracos de cada linguagem no contexto dos algoritmos utilizados.

2 Objetivos

- Avaliar o desempenho de algoritmos clássicos (como ordenação, busca e recursão) em diferentes linguagens de programação.
- Identificar fatores que influenciam a performance nas linguagens analisadas.
- Discutir o impacto da linguagem no consumo de recursos computacionais.

3 Fundamentação Teórica

3.1 Algoritmos Clássicos

Definição de algoritmos como ordenação (QuickSort, MergeSort), busca (Binária) e estruturas recursivas.

3.2 Complexidade Computacional

A análise dos algoritmos considerados neste trabalho será baseada nas classes de complexidade da Teoria da Computação. Estas classes descrevem o comportamento dos algoritmos em termos de tempo e espaço exigidos conforme o tamanho da entrada cresce.

A classe **P** representa os problemas que podem ser resolvidos em tempo polinomial por uma máquina determinística. Já a classe **NP** representa os problemas cujas soluções podem ser verificadas em tempo polinomial, mesmo que não saibamos resolvê-los de forma eficiente.

Problemas **NP-completos** são os mais difíceis dentro da classe NP, e um algoritmo eficiente para qualquer um deles implicaria em uma solução eficiente para todos os problemas NP. Já os **NP-difíceis** não pertencem necessariamente à classe NP, mas são ao menos tão difíceis quanto os problemas NP-completos.

As definições e análises dos algoritmos implementados neste trabalho serão fundamentadas nessas categorias de complexidade, permitindo compreender não apenas a eficiência empírica, mas também os limites teóricos esperados para cada abordagem em diferentes linguagens de programação.

3.3 Linguagens de Programação

Serão consideradas as linguagens mais populares de acordo com rankings do TIOBE tiobe [2024], GitHub Octoverse octoverse [2022] e Stack Overflow Developer Survey

stackoverflow [2024]:

- Python
- C
- C++
- Java
- JavaScript
- Go
- Rust
- TypeScript
- C#
- Kotlin

4 Justificativa

Com a crescente demanda por sistemas eficientes e responsivos, a escolha da linguagem de programação tornou-se um fator crítico no desenvolvimento de software. Embora existam inúmeros estudos sobre algoritmos, poucos se concentram em comparações práticas entre linguagens populares no contexto da eficiência computacional. Este trabalho se justifica pela necessidade de fornecer uma base empírica que auxilie desenvolvedores, estudantes e pesquisadores na tomada de decisões mais conscientes, considerando o impacto direto da linguagem no desempenho final das aplicações.

5 Delimitação do Tema

Este estudo foca na avaliação de algoritmos clássicos, como ordenação, busca e estruturas recursivas, implementados em linguagens de programação populares. A análise será limitada ao desempenho desses algoritmos em ambiente controlado, utilizando dados sintéticos. O escopo não abrange algoritmos paralelos ou distribuídos, nem aspectos específicos de bibliotecas ou frameworks externos. O foco é a linguagem em si e sua eficiência na execução de algoritmos representativos.

6 Metodologia

Será desenvolvida uma suíte de testes para implementar os mesmos algoritmos em cada linguagem selecionada. Os testes medirão:

- Tempo de execução
- Uso de memória
- Facilidade de implementação (complexidade de código)

As execuções serão repetidas várias vezes em uma mesma máquina para garantir consistência. Serão usados dados sintéticos com diferentes tamanhos para simular cenários variados.

7 Resultados Esperados

Espera-se identificar quais linguagens oferecem melhor desempenho para tarefas específicas e como características como paradigma, tipagem, compilação influenciam o comportamento dos algoritmos. Esses dados poderão auxiliar desenvolvedores e pesquisadores na escolha mais adequada de linguagem para projetos com requisitos de desempenho.

Referências Bibliográficas

octoverse. GitHub Octoverse 2022. https://octoverse.github.com/2022/top-programming-languages, 2022. Acesso em: abril 2025.

stackoverflow. Stack Overflow Developer Survey 2024. https://survey.stackoverflow.co/2024/technology, 2024. Acesso em: abril 2025.

tiobe. TIOBE Index for April 2025. https://www.tiobe.com/tiobe-index/, 2024. Acesso em: abril 2025.