Desafio de Programação: Cálculo da Matriz Inversa

Implementar métodos para o cálculo da matriz inversa utilizando Python, C ou Java, bem como mensurar e comparar os tempos de execução para diferentes tamanhos de matrizes. Posteriormente, realizar uma análise teórica relacionando os resultados experimentais com a complexidade computacional dos algoritmos.

---

1. Objetivos

- Implementar o método de **Gauss-Jordan** para o cálculo da matriz inversa.
- Implementar o método de **Decomposição LU** para o cálculo da matriz inversa.
- Mensurar o tempo de execução de ambos os métodos para matrizes de tamanhos variados.
- Relacionar os tempos de execução experimentais com as complexidades algorítmicas, discutindo as diferenças teóricas.

---

2. Introdução ao Problema

A matriz inversa é fundamental em diversas aplicações matemáticas e computacionais, incluindo:

- Solução de sistemas lineares.
- Transformações geométricas.
- Processamento de sinais e machine learning.

Dois métodos distintos podem ser empregados para calcular a matriz inversa:

2.1. Método de Gauss-Jordan

- **Descrição**: O método consiste em transformar a matriz original na matriz identidade através de operações elementares de linha, aplicadas simultaneamente a uma matriz identidade inicial.
- **Vantagem**: Simples de entender e direto na implementação.
- **Desvantagem**: Pode ser menos eficiente para matrizes grandes devido à grande quantidade de operações envolvidas.
- **Aplicabilidade**: Adequado para matrizes de pequeno e médio porte.

2.2. Método de Decomposição LU

- **Descrição**: A matriz é fatorada em duas matrizes triangulares (L e U), e a solução é obtida resolvendo sistemas lineares auxiliares.
- **Vantagem**: Maior eficiência para matrizes grandes e aplicações repetitivas.
- **Desvantagem**: Requer uma implementação mais elaborada e pode ser menos intuitivo.
- **Aplicabilidade**: Adequado para matrizes de médio e grande porte.

---

3. Implementação e Medição de Tempo de Execução

Implementar ambos os métodos utilizando a linguagem escolhida (Python, C ou Java) e mensurar o tempo de execução para os seguintes casos de teste:

- Matriz 3x3
- Matriz 5x5
- Matriz 8x8
- Matriz 20x20
- Matriz 100x100

Recomendações para a medição do tempo:

- **Python**: Utilizar funções como time.time() ou time.perf_counter().
- **C**: Utilizar a função clock() da biblioteca <time.h>.
- **Java**: Utilizar System.nanoTime().

Registrar os tempos de execução para cada método e cada tamanho de matriz.

---

4. Análise de Resultados e Complexidade Computacional

- Elaborar gráficos que relacionem o tempo de execução com o tamanho da matriz para ambos os métodos.
- Investigar e descrever os conceitos de complexidade algorítmica, especialmente as diferenças entre as complexidades $O(n^3)$ (Gauss-Jordan) e $O(n^3)$ (Decomposição LU).
- Redigir uma análise que explique a discrepância nos tempos de execução observados e correlacione esses resultados com a teoria de complexidade computacional.

---

5. Conclusão

- Avaliar a eficiência dos métodos utilizados para o cálculo da matriz inversa.
- Discutir possíveis otimizações para cálculos de matrizes inversas de grande dimensão.