Desafio de Programação: Cálculo do Determinante e Análise de Complexidade

Implementar dois métodos para o cálculo do determinante de uma matriz quadrada 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 **Expansão de Laplace (Recursivo)** para o cálculo do determinante.
- Implementar o método de **Eliminação de Gauss** para o cálculo do determinante.
- 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

O determinante de uma matriz é um valor numérico associado a uma matriz quadrada, com aplicações que incluem:

- Verificação da invertibilidade de uma matriz (determinante diferente de zero).
- Solução de sistemas lineares.
- Cálculo de áreas e volumes em transformações lineares.

Dois métodos distintos podem ser empregados para calcular o determinante:

2.1. Expansão de Laplace (Recursão)

- **Descrição:** Consiste em expandir o determinante utilizando os menores complementares. Para uma matriz n×nn \times nn×n, realiza-se a expansão pela primeira linha e calcula-se recursivamente os determinantes das submatrizes obtidas.
- **Vantagem:** Facilidade de compreensão e implementação.
- **Desvantagem:** Ineficiência para matrizes de maior ordem, devido à complexidade de tempo **O(n!)**.

- **Aplicabilidade:** Indicada para matrizes pequenas (por exemplo, 3x3 ou 4x4).

2.2. Eliminação de Gauss

- **Descrição:** Consiste em transformar a matriz em forma triangular superior por meio de operações de linha. O determinante é então obtido pelo produto dos elementos da diagonal principal.
- **Vantagem:** Eficiência no cálculo para matrizes de ordem elevada, com complexidade de tempo **$O(n^3)$**.
- **Desvantagem:** Implementação relativamente mais complexa.
- **Aplicabilidade:** Adequada para matrizes de ordem mediana a elevada (por exemplo, 5x5 ou superiores).

---

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 **$O(n!)$** (Expansão de Laplace) e **$O(n^3)$** (Eliminação de Gauss).

- Redigir uma análise que explique a discrepância nos tempos de execução observados e correlacione esses resultados com a teoria de complexidade computacional.