INSTITUTO FEDERAL
Sergipe

Otimização do Algoritmo Recursivo de Fibonacci – Aplicando Programação Dinâmica

(APP) Atividade Prática com Prazo de Entrega e Com Consulta – Peso 1

Objetivos Pedagógicos da Atividade

1. Explicar por que a versão recursiva ingênua de ***`Fibonacci`*** é exponencial (sobreposição de subproblemas, chamadas redundantes) e identificar riscos de ***`stack overflow;`***
2. Descrever princípios de Programação Dinâmica;
3. Implementar corretamente a otimização, garantindo que cada ***`F(n)`*** seja calculado no máximo uma vez;
4. Analisar complexidades: demonstrar a mudança de tempo para as versões ingênua e otimizada;
5. Instrumentar medições de desempenho (p.ex., ***`System.nanoTime`***) e conduzir experimentos comparando três abordagens: recursiva ingênua, recursiva memorizada e iterativa;
6. Comunicar decisões técnicas no código (nomes claros, comentários objetivos, testes manuais no projeto base) e empacotar a entrega conforme o padrão solicitado;
7. Gerenciar o próprio processo de trabalho (cumprir prazos, compreender bônus/penalidades) como parte de responsabilidade do discente.

Problema

- O **`método`** ***`fibonacciRecursivo`*** implementado na **`classe`** ***`Fibonacci`*** dentro do ***`package algoritmos.recursivos`*** no projeto **`algoritmos-estrutura-dados-java-lab`** (parte 02) é uma implementação válida e didática para entender o problema, mas impraticável em cenários reais, por isso é denominada de **`abordagem ingênua.`** A abordagem ingênua tem **`três problemas principais: 1) Redundância Computacional Exponencial`** - onde cada chamada recursiva recalcula os mesmos valores múltiplas vezes. **`2) Ineficiência em Memória –`** a recursão sem otimização cria uma pilha de chamadas *(call stack)* com profundidade

n, consumindo ***O(n)*** espaço e para *n* grandes, isso pode causar *stack overflow* e 3) **`Falta de Otimização`** - não aproveita resultados já computados (ao contrário de técnicas de programação dinâmica). Portanto, uma das **`técnicas de otimização`** propõe armazenar os resultados dos números de *Fibonacci* calculados anteriormente em uma tabela para evitar cálculos redundantes. Isso garante que cada número de *Fibonacci* seja calculado apenas uma vez, **`reduzindo a complexidade`** de tempo exponencial da abordagem ingênua ***O(2^n)*** para uma complexidade de tempo ***O(n)*** mais eficiente.

Otimização

1. Implemente um novo método estático ***`fibonacciRecursivoMemorizado`*** na classe ***`Fibonacci`***. Use a técnica mais adequada de programação dinâmica para otimização conforme o problema - valor (8,0) pontos.
2. Implemente as medições de tempo para a versão do método ***`fibonacciRecursivoMemorizado`*** e compare com as versões anteriores: ***`fibonacciRecursivo`*** (que implementa a versão ingênua) e ***`fibonacciIterativo`*** (que implementa a versão iterativa) - valor (2,0) pontos.

Critérios de Avaliação

- Implementação da otimização correta com a técnica mais adequada;
- Corretude do código (lógica);
- Entendimento conceitual sobre Recursividade e Programação Dinâmica.