FIAP – FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA

AUGUSTO DOUGLAS NOGUEIRA DE MENDONÇA - RM558371
GABRIEL VASQUEZ QUEIROZ DA SILVA – RM557056
GUILHERME ARAUJO DE CARVALHO – RM558926
GUSTAVO OLIVEIRA RIBEIRO – RM559163

CHALLENGE – DYNAMIC PROGRAMMING

Sprint 4

SÃO PAULO
2025

SUMÁRIO

INTRODUÇÃO

A crescente digitalização dos processos laboratoriais tem impulsionado o uso de técnicas computacionais para aprimorar a precisão e a eficiência das análises patológicas. No contexto da DASA, a análise macroscópica de amostras representa uma etapa essencial no diagnóstico, mas que ainda pode envolver fluxos pouco otimizados, impactando o tempo de execução e a produtividade dos profissionais responsáveis.

Com o objetivo de aprimorar esse processo, este projeto propõe a aplicação de Programação Dinâmica (PD) para otimizar o agrupamento das etapas de análise, reduzindo o custo total de processamento e interligação entre módulos de visão computacional. A formulação segue a estrutura clássica da PD, com definição de estados, decisões, função de transição e função objetivo, além da implementação de três versões do algoritmo: recursiva simples, recursiva com memorização e iterativa (bottom-up).

A solução foi desenvolvida em Python, priorizando clareza e reprodutibilidade. Foram implementados testes automatizados para validar a equivalência entre as abordagens e disponibilizado um repositório completo no GitHub, que documenta todo o processo de desenvolvimento e permite futuras expansões, como integração com dados reais e análise visual dos resultados de otimização.

1 FORMULAÇÃO DO PROBLEMA

1.1 Contextualização

O processo de análise macroscópica de amostras laboratoriais envolve uma sequência de etapas que transformam dados brutos em informações úteis para o diagnóstico. Cada etapa — como captação de imagem, filtragem, segmentação, análise de textura e extração de padrões — possui um custo computacional distinto, representado por sua complexidade de entrada e saída (I/O). A forma como essas etapas são encadeadas impacta diretamente o tempo total de execução do exame. Assim, o desafio consiste em encontrar a ordem ótima de processamento que minimize o custo total, garantindo maior eficiência e menor sobrecarga aos sistemas e profissionais.

Esse problema pode ser modelado como uma variação do problema de multiplicação ótima de cadeias de matrizes, resolvido classicamente por Programação Dinâmica (PD). A seguir, descrevem-se formalmente os componentes do modelo:

- Estado: Representa um subproblema que considera a execução ótima entre as etapas i e j da pipeline, denotado como C(i, j).
- Decisão: Escolher o ponto de divisão k entre i e j, indicando onde a sequência de operações será separada.
- Função de transição:

$$C(i,j)=i\leq k<j\min\ [C(i,k)+C(k+1,j)+Pi-1\times Pk\times Pj\ ]$$

  onde PPP representa as complexidades de entrada e saída de cada módulo.
- Função objetivo: Minimizar o custo total de execução da pipeline, isto é, encontrar C(1, N).

1.2 Solução

Três versões foram desenvolvidas para resolver o problema:

- Versão Recursiva Simples: Implementação direta da equação de recorrência, com complexidade exponencial, usada como referência conceitual.
- Versão Recursiva com Memorização (Top-Down): Introduz o uso de cache para armazenar subproblemas já resolvidos, reduzindo o tempo de execução para O(n3)O(n^3)O(n3).

- Versão Iterativa (Bottom-Up): Baseia-se em tabulação, preenchendo uma matriz de custos de forma incremental, também com complexidade O(n3)O(n^3)O(n3), mas com melhor controle de memória e desempenho previsível.

2 RELATÓRIO TÉCNICO

O código foi inteiramente desenvolvido em Python 3, com o objetivo de demonstrar, de forma prática, a aplicação de Programação Dinâmica (PD) na otimização de pipelines de análise macroscópica. O projeto encontra-se documentado e versionado em um repositório no GitHub, contendo tanto o algoritmo principal quanto os testes de verificação de coerência entre as versões recursiva e iterativa.

A estrutura do programa é organizada de forma modular, contemplando funções independentes para entrada e validação de dados, execução das versões de PD e interface interativa via terminal.
A função get_pipeline_complexities() é responsável pela coleta e validação das complexidades de I/O de cada etapa da pipeline, representando a dimensão dos dados entre as operações de análise.
Esses valores são armazenados na lista P, que constitui o parâmetro fundamental das demais funções de cálculo.

As três implementações da PD — custo_recursivo_simples, custo_recursivo_memoizado e custo_iterativo_bottom_up — representam diferentes abordagens para resolver o mesmo problema de otimização.
A versão recursiva simples é usada apenas como referência teórica, enquanto a versão com memorização (Top-Down) introduz o uso de um dicionário (memo) para armazenar subproblemas resolvidos, eliminando recomputações desnecessárias.
Por fim, a versão iterativa (Bottom-Up) constrói uma matriz dp[i][j] que armazena os custos mínimos entre as etapas, permitindo que a solução seja obtida de forma incremental e eficiente.

A função verificar_e_executar_dp() realiza a execução das duas abordagens dinâmicas, medindo o tempo de processamento e comparando os resultados obtidos.
Quando ambos os métodos retornam o mesmo valor mínimo de custo, é exibida a mensagem de coerência verificada, comprovando a consistência da formulação teórica e da implementação prática.

O projeto ainda inclui uma função de menu (main_menu()) que oferece uma interface textual amigável, permitindo que o usuário defina os parâmetros da pipeline, execute a otimização e visualize os resultados diretamente no terminal.
Essa escolha reforça a acessibilidade e clareza pedagógica do sistema, tornando-o ideal tanto para fins acadêmicos quanto para prototipagem de soluções reais no ambiente da DASA.

CONCLUSÃO

O projeto apresentado demonstrou, de forma estruturada e prática, como a Programação Dinâmica pode ser aplicada para otimizar pipelines de análise macroscópica de amostras laboratoriais, cenário típico de exames realizados por patologistas da DASA. A abordagem adotada permitiu modelar o problema com definição clara de estados, decisões, função de transição e função objetivo, traduzindo o fluxo de trabalho real em uma estrutura computacional capaz de minimizar o custo total de processamento.

A implementação em Python, organizada de maneira modular e interativa, permitiu comparar três abordagens distintas: a versão recursiva simples, a versão recursiva com memorização (Top-Down) e a versão iterativa (Bottom-Up). O desenvolvimento demonstrou que ambas as soluções dinâmicas produzem resultados consistentes, enquanto a versão iterativa oferece melhor eficiência e previsibilidade de tempo de execução, especialmente para pipelines com múltiplas etapas. Além disso, a inclusão de testes unitários automatizados garantiu a confiabilidade do código e a robustez da solução frente a diferentes cenários de complexidade.

O projeto evidencia, ainda, que técnicas clássicas de otimização, quando aplicadas de maneira cuidadosa ao contexto da visão computacional e patologia digital, podem gerar benefícios tangíveis em termos de eficiência operacional, redução de redundâncias e uso otimizado de recursos computacionais. O repositório completo no GitHub, com código e testes, garante transparência, reprodutibilidade e possibilidade de expansão futura, permitindo integração com bancos de dados reais, visualização gráfica ou sistemas mais complexos de automação laboratoriais.

Em síntese, este trabalho não apenas cumpriu os objetivos propostos — modelar, implementar e validar uma solução de Programação Dinâmica para otimização de pipelines de análise — como também forneceu uma base sólida para aplicações futuras em laboratórios digitais, mostrando que a combinação entre modelagem matemática e implementação prática é uma ferramenta poderosa para melhorar processos clínicos e apoiar decisões médicas de forma eficiente e confiável.

REFERÊNCIAS BIBLIOGRÁFICAS

Artigo: SILVA, João; SOUZA, Maria. *Valores de referência para exames laboratoriais de colesterol, hemoglobina glicada e creatinina da população adulta brasileira*. Revista Brasileira de Epidemiologia, v. 22, Suppl. 02, 2019. Disponível em: SciELO. Acesso em: 15 out. 2025.

Site especializado: PINHEIRO, Pedro. *Valores de referência dos principais exames laboratoriais*. MD.Saúde, 2025. Disponível em: MD.Saúde. Acesso em: 15 out. 2025.

Documento acadêmico: PUC-RIO. *Referências bibliográficas sobre exames de sangue e hemoterapia*. Rio de Janeiro: PUC-Rio, 2025. Disponível em: PUC-Rio. Acesso em: 15 out. 2025.

Livro: CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms.* 3. ed. Cambridge: MIT Press, 2009. Disponível em: cs.cmu.edu. Acesso em: 31 out. 2025.

Artigo: HUANG, *Q. et al. Computational pathology: A comprehensive review of ...* **ScienceDirect**, 2025. Disponível em: ScienceDirect. Acesso em: 31 out. 2025.

Artigo: SMITH, B. et al. *Developing image analysis pipelines of whole-slide images.* **PMC/NCBI**, 2020. Disponível em: PMC/NCBI. Acesso em: 31 out. 2025.

Documento Acadêmico: GEUBBELMANS, M. *Optimization of whole slide imaging scan settings for computer vision using human lung cancer tissue*. **ResearchGate**, 2024. Disponível em: ResearchGate. Acesso em: 31 out. 2025.

Artigo: KOMURA, D.; ISHIKAWA, S. *Machine learning methods for histopathological image analysis.* **ScienceDirect**, 2024. Disponível em: ScienceDirect. Acesso em: 31 out. 2025.