PUC Minas

PONTIFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Trabalho Prático

Fundamentos de Projeto e Análise de Algoritmos

Em ambientes com muitos dados (como sistemas, sensores ou logs de servidores), diferentes equipes podem registrar os mesmos eventos de formas diferentes. Analisar esses dados em conjunto exige que se encontrem padrões comuns, mesmo com variações nas sequências.

PUC Minas

Equipe

ANA CLARA BERTOLDO

CAROLINA BAIÃO

EVALDO ARAUJO

MATHEUS YURI

RYAN SAMUEL

Índice

Introdução

Programação Dinâmica

É uma técnica de resolução de problemas complexos dividindo-os em subproblemas menores. Os resultados dos subproblemas são armazenados (memorizados) para evitar repetições. Ideal para problemas com repetição de cálculos, como sequências, caminhos mínimos e alinhamento de strings.

Programação Dinâmica + BackTracking

Usamos programação dinâmica para encontrar o tamanho da maior subsequência comum (LCS). Depois, usamos backtracking para reconstruir todas as subsequências possíveis com esse tamanho, seguindo os caminhos válidos no vetor de memória.

Objetivo

Programação Dinâmica

Implementar uma solução eficiente que calcule o tamanho da maior subsequência comum entre duas sequências de eventos. A programação dinâmica permite evitar cálculos repetitivos, otimizando o tempo de execução do programa ao armazenar os resultados de subproblemas anteriores.
O foco é garantir que, mesmo com entradas maiores, o programa consiga fornecer a resposta de forma rápida e precisa.

Programação Dinâmica + BackTracking

Após calcular o tamanho da subsequência comum com programação dinâmica, o objetivo com o backtracking é descobrir todas as subsequências possíveis que possuem esse tamanho.
O backtracking percorre os caminhos válidos no vetor de memória gerado pela programação dinâmica, explorando todas as combinações possíveis de eventos em comum, sempre respeitando a ordem original. Com isso, conseguimos não só saber o "quanto" as sequências têm em comum, mas também "quais são" essas subsequências.

METODOLOGIA

Leitura e Validação
O programa começa lendo a quantidade de conjuntos e as duas sequências de caracteres (Helena e Marcus). Depois, valida se cada sequência tem até 80 letras e usa só letras minúsculas de ‘a’ a ‘z’.

Construção da Matriz (DP)
Criamos uma matriz que compara os caracteres das duas sequências. Cada célula armazena o tamanho da maior subsequência comum até aquele ponto. Isso permite calcular a LCS de forma eficiente.

Backtracking e Resultados
Com a matriz pronta, usamos recursão para voltar pelas posições e encontrar todas as LCS possíveis.
O resultado é exibido em ordem alfabética, sem repetições, como solicitado no enunciado.

Considerações Finais

O projeto permitiu aplicar na prática conceitos de programação dinâmica e backtracking para resolver o problema da LCS.

Foram implementadas duas versões: uma que calcula apenas o comprimento da subsequência comum, e outra que exibe todas as soluções possíveis.

A linguagem C# foi utilizada por oferecer estrutura clara para manipulação de listas, vetores e funções recursivas.

Validamos as entradas conforme os critérios do enunciado e testamos diferentes combinações de dados.

Concluímos que a combinação de DP e backtracking é eficiente para esse tipo de problema e reforça a importância da modelagem de algoritmos.

Prática

Referências

Microsoft Docs. System.Collections.Generic Namespace. Disponível em: https://learn.microsoft.com/pt-br/dotnet/api/system.collections.generic?view=net-9.0.
Acesso em: 20 jun. 2025.

Microsoft Docs. System.Linq Namespace. Disponível em: https://learn.microsoft.com/pt-br/dotnet/api/system.linq?view=net-9.0.
Acesso em: 20 jun. 2025.

Obrigado!