Relatório do Projeto

Longest Common Subsequence

Pedro Cunial

May 9, 2017

Contents

Abstract

Um problema muito comum na biotecnologia é definir a semelhança entre dois genomas, ou seja, o quão parecidos são os *DNAs* de dois seres vivos, seja para um teste de paternidade, ou até mesmo para definir o quão distante genéticamente o homem está do macaco. Computacionalmente este cálculo torna-se muito extenso devido a grande quantidade de redundâncias, neste artigo, pretende-se discutir a possível otimização na solução do problema pelo uso da programação dinâmica.

Palavras Chave: Programação Dinâmica, Biotecnologia, Algoritmos, Recursão, Memoização

1 Introdução

Muito comum em problemas computacionais relacionados à biotecnologia, conhecido como *Longest Common Subsequence* busca encontrar a maior subsequência de caracteres comuns em duas sequências (sejam elas palavras ou apenas sequências de caracteres). Por definição, uma subsequência difere de uma substring (ou subtexto) no sentido de que uma subsequência não precisa representar caracteres contínuos em uma sequência, da forma que $ADFGE$ é uma subsequência de $AEDGFGE$ com $DGADEFEGAE$, por mais que não seja uma substring.

2 Aplicações

Como o próprio exemplo citado já sugere, o problema pode ser utilizado para cruzar genomas humanos com o intuito de descobrir o quão semelhantes são, estudo utilizado em exames de paternidade, por exemplo. Além disso, expandindo o conceito do problema, se substitui-se a busca de uma subsequência em duas sequências pela busca de uma subsequência de palavras em dois livros ou textos, podendo utilizar o mesmo algoritmo como indicador de plágio em obras.

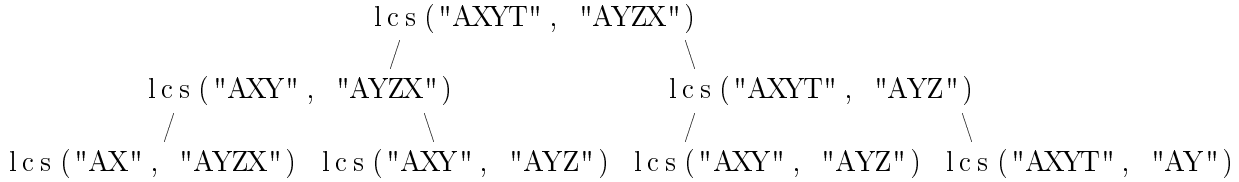
Este segundo exemplo é o utilizado pela maior parte das ferramentas populares de controle de versão (como o Git) para a reconciliação de mudanças (`git merge`, `git pull` etc).

3 Implementação

3.1 Definição

Para encontrar a maior subsequência em duas sequências, pode-se análisar os caracteres destas sequências par a par até o seu fim, de forma que quando o resultado for igual encontramos mais um caractere da subsequência.

Desta forma, uma solução ingênua para o problema seria implementar uma recursão caractere a caractere cruzando-os, gerando uma árvore como a seguinte:



fonte: http://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/

3.2 Melhorias Possíveis

Pela própria representação da árvore de chamadas da função, percebe-se que o cálculo redundante da subsequência de "AXY" com "AYZ" dos dois lados da árvore. Dado que o cálculo de cada subsequência já é uma operação muito custosa por si, o cálculo redundante da mesma torna-se um enorme empecilho para a viabilidade do algoritmo.

Felizmente, este problema pode ser resolvido pela técnica de memoização, salvando os cálculos já feitos, de forma que cálculos redundantes sejam substituidos por apenas uma consulta (de tempo constante) em uma estrutura de dados salvos.

4 Didática

4.1 Principais Pontos

Em aspéctos didáticos, os pontos mais relevantes a serem discutidos no algoritmo são as partes em que a memoização pode ser utilizada, assim como qual estrutura de dados utilizar para a mesma.

Dada a simplicidade do algoritmo em si, a abordagem de como transcrever o código de *top-down* para *bottom-up*, além de explorar casos solucionáveis pelo problema, desenvolvendo uma solução em conjunto com a sala em Python – a sugestão da solução ser desenvolvida em Python vem da possível dificuldade e falta de familiaridade do público com a sintaxe do C ou C++, além das simplificações didáticas da linguagem.

4.2 Principais Dificuldades

Dado o público alvo da aula, um uso excessivo de termos técnicos, principalmente associados à recursão pode ser problemático. Além disso, como citado anteriormente a sintaxe do C ou C++ provavelmente será um aspecto problemático, o desenvolvimento de um código em uma aula de algoritmo seria de grande serviço para a sua compreensão.

Por fim, o conceito de complexidade pode ser problemático, no entanto, algumas simplificações podem ser feitas, como por exemplo simplesmente aceitar que uma consulta na estrutura utilizada para memoização não tem

custo computacional algum, enquanto mais uma iteração do código teria um custo consideravelmente alto.

5 Conclusão

O problema *Longest Common Subsequence* é de grande relevância não só para o estudo de temas da biotecnologia mas também para o estudo e aprendizado de algoritmos, principalmente de programação dinâmica.

Apesar de não ser tão completo quanto outros casos estudados pela programação dinâmica, o *Longest Common Subsequence* é um bastante completo, no sentido que sua implementação ingênua é bastante simples, de forma que transformá-lo em um algoritmo de programação dinâmica é um processo bastante linear, permitindo um terceiro passo que seria o transformar do algoritmo top-down em sua implementação bottom-up.