Problem Set 2 - Exercise 5

Fabio Pernisi

January 2024

Problem Description

Given two binary sequences of lengths n and m, respectively, find the length of their longest common subsequence.

Running time $O(n^2 + m)$.

Solution

The problem at hand is a modification of the classical Longest Common Subseuqence (LCS). Instead of storing the LCS length directly, our dynamic programming table $dp[i][\ell]$ represents the *smallest index j in sequence B for which the LCS of $A[1..i]$ and $B[1..j]$ has a length of at least ℓ.*

This redefinition allows us to use the properties of binary sequences to update the table entries more efficiently.

To understand the optimization, consider the nature of binary sequences: each element is either '0' or '1'. This allows us to preprocess B to quickly determine the next occurrence of a matching chsracter. For each position in B, we can store the next index where '0' and '1' appear. With this information, when we wish to extend an LCS by one element (say $A[i] = 1$), we can directly jump to the next occurrence of '1' in B beyond the current LCS boundary.

The table is filled by iterating over the lengths of possible subsequences in A and for each, determining the minimum extension needed in B. The update rule for $dp[i][\ell]$ leverages the precomputed positions of '0's and '1's in B and uses the previous subsequence information to find the smallest index j efficiently.

By only considering extensions of the LCS when a match is found and quickly skipping non-matching characters, we avoid the $O(m)$ per-entry cost typical of standard LCS algorithms, thus achieving the $O(n^2 + m)$ running time.

Pseudocode

```
Input: Binary sequences A of length n, B of length m
Output: Length of the longest common subsequence
function BinaryLCS(A, B)
    Preprocess B to record the first appearance of each binary character after each position
    Initialize a 2D array dp with dimensions n + 1 by n + 1, filled with ∞
    for i ← 1 to n do
        for ℓ ← 1 to i do
            dp[i][ℓ] ← min(dp[i − 1][ℓ], index of first appearance of A[i] in B after dp[i − 1][ℓ − 1])
        end for
    end for
    return maximum ℓ such that dp[n][ℓ] ≠ ∞
end function
```