CS 341: Algorithms
Lec 12: Dynamic Programming- Part 2

Armin Jamshidpey   Collin Roberts

Based on lecture notes by Éric Schost

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2025

The Longest Increasing Subsequence Problem

Input: An array $A[1..n]$ of integers

Output: A **longest increasing subsequence** of A (or just its length) (does **not** need to be contiguous)

Example: $A = [7, 1, 3, 10, 11, 5, 19]$ gives $[7, 1, 3, 10, 11, 5, 19]$

Remark: there are 2^n subsequences (including an empty one, which doesn't count)

Tentative subproblems

Attempt 1:

- **Subproblems:** the length $\ell[i]$ of a longest increasing subsequence of $A[1..i]$
- on the example, $\ell[6] = 4$
- so what? not enough to deduce $\ell[7]$

Tentative subproblems

Attempt 1:

- **Subproblems:** the length $\ell[i]$ of a longest increasing subsequence of $A[1..i]$
- on the example, $\ell[6] = 4$
- so what? not enough to deduce $\ell[7]$

Attempt 2:

- **Subproblems:** the length $\ell[i]$ of a longest increasing subsequence of $A[1..i]$, together with its last entry
- example: $\ell[6] = 4$, with last element 11
- OK if we can add $A[i+1]$, but what if not?

A more complicated recurrence

Attempt 3:

- let $L[i]$ be the length of a longest increasing subsequence of $A[1..i]$ **that ends with** $A[i]$, for $i = 1, \ldots, n$
- so $L[1] = 1$

Idea:

- a longest increasing subsequence S ending at $A[i]$ looks like

$$S = [\ldots, A[j], A[i]] = S' \textbf{ cat } [A[i]]$$

- S' is a longest increasing subsequence ending at $A[j]$ (or it is empty)
- don't know j, but we can try all $j < i$ for which $A[j] < A[i]$

Iterative algorithm

LongestIncreasingSubsequence$(A[1..n])$

$L[1] \leftarrow 1$
for $i = 2, \ldots, n$ **do**
  $L[i] \leftarrow 1$
  for $j = 1, \ldots, i-1$ **do**
    if $A[j] < A[i]$ **then**
      $L[i] = \max(L[i], L[j] + 1)$
return the maximum entry in L

Runtime: $\Theta(n^2)$

Remark:

- the algorithm does not return the sequence itself, but could be modified to do so

The Longest Common Subsequence Problem

Input: Arrays $A[1..n]$ and $B[1..m]$ of characters

Output: The maximum length k of a common subsequence to A and B
(subsequences do **not** need to be contiguous)

Example: $A =$**blurry**, $B =$**burger**, longest common subsequence is **burr**

Remark: there are 2^n subsequences in A, 2^m subsequences in B

A bivariate recurrence

Definition: let $M[i, j]$ be the longest subsequence between $\boldsymbol{A[1..i]}$ and $\boldsymbol{B[1..j]}$

- $M[0, j] = 0$ for all j
- $M[i, 0] = 0$ for all i
- $M[i, j]$ is the max of **up to three** values
  - $M[i, j-1]$ (don't use $B[j]$)
  - $M[i-1, j]$ (don't use $A[i]$)
  - **if** $A[i] = B[j]$, $\boldsymbol{1 + M[i-1, j-1]}$

The algorithm computes all $M[i, j]$, using two nested loops, so runtime $\boldsymbol{\Theta(mn)}$