CS 341: Algorithms

Lecture 8: Dynamic programming, continued

Éric Schost

based on lecture notes by many other CS341 instructors

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

Fall 2024

Longest increasing subsequence

The 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 $[7, \color{red}{1}, \color{red}{3}, \color{red}{10}, \color{red}{11}, 5, \color{red}{19}]$

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 $\ell[7]$

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

Bonus: a faster algorithm

As before, $\boldsymbol{\ell[i]}$=of a longest increasing subsequence of $A[1..i]$

Idea: we consider the "best" increasing sequences in $A[1..i]$

- can have several increasing sequence of length $\boldsymbol{j}$ for each $\boldsymbol{j = 1, \ldots, \ell[i]}$
- for any j, **best** increasing sequence of length j: one whose **last entry** is the **smallest**

Example: $A = [\mathbf{2}, \mathbf{8}, \mathbf{10}, \mathbf{11}, 1, 3, \mathbf{5}]$, $\boldsymbol{\ell[6] = 4}$, done $\boldsymbol{i = 6}$

- $j = 1$, best increasing sequence $[1]$ can add $\mathbf{5}$
- $j = 2$, best increasing sequence $[1, 3]$ can add $\mathbf{5}$
- $j = 3$, best increasing sequence $[2, 8, 10]$ can't add $\mathbf{5}$
- $j = 4$, best increasing sequence $[2, 8, 10, 11]$ can't add $\mathbf{5}$

Bonus: a faster algorithm

As before, $\boldsymbol{\ell[i]}$=of a longest increasing subsequence of $A[1..i]$

Idea: we consider the "best" increasing sequences in $A[1..i]$

- can have several increasing sequence of length $\boldsymbol{j}$ for each $\boldsymbol{j = 1, \ldots, \ell[i]}$
- for any j, **best** increasing sequence of length j: one whose **last entry** is the **smallest**

Example: $A = [\mathbf{2}, \mathbf{8}, \mathbf{10}, \mathbf{11}, 1, 3, \mathbf{5}]$, $\boldsymbol{\ell[6] = 4}$, doing $\boldsymbol{i = 7}$

- $j = 1$, best increasing sequence $[\mathbf{1}]$ can add $\mathbf{5}$
- $j = 2$, best increasing sequence $[1, \mathbf{3}]$ can add $\mathbf{5}$
- $j = 3$, best increasing sequence $[2, 8, \mathbf{10}]$ can't add $\mathbf{5}$
- $j = 4$, best increasing sequence $[2, 8, 10, \mathbf{11}]$ can't add $\mathbf{5}$

$\mathbf{1 < 3 < 5 < 10 < 11}$ so $\boldsymbol{\ell[7] = 4}$ and we update the $j = 3$ sequence to $[1, 3, 5]$

Bonus: a faster algorithm

As before, $\boldsymbol{\ell[i]}$=of a longest increasing subsequence of $A[1..i]$

Idea: we consider the "best" increasing sequences in $A[1..i]$

- can have several increasing sequence of length $\boldsymbol{j}$ for each $\boldsymbol{j = 1, \ldots, \ell[i]}$
- for any j, **best** increasing sequence of length j: one whose **last entry** is the **smallest**

Example: $A = [\mathbf{2}, \mathbf{8}, \mathbf{10}, \mathbf{11}, 1, 3, \mathbf{15}]$, $\boldsymbol{\ell[6] = 4}$, done $\boldsymbol{i = 6}$

- $j = 1$, best increasing sequence $[1]$ can add **15**
- $j = 2$, best increasing sequence $[1, 3]$ can add **15**
- $j = 3$, best increasing sequence $[2, 8, 10]$ can add **15**
- $j = 4$, best increasing sequence $[2, 8, 10, 11]$ can add **15**

Bonus: a faster algorithm

As before, $\boldsymbol{\ell}[\boldsymbol{i}]$=of a longest increasing subsequence of $A[1..i]$

Idea: we consider the "best" increasing sequences in $A[1..i]$

- can have several increasing sequence of length $\boldsymbol{j}$ for each $\boldsymbol{j = 1, \dots, \ell[i]}$
- for any j, **best** increasing sequence of length j: one whose **last entry** is the **smallest**

Example: $A = [\mathbf{2}, \mathbf{8}, \mathbf{10}, \mathbf{11}, 1, 3, \mathbf{15}]$, $\boldsymbol{\ell[6] = 4}$, doing $\boldsymbol{i = 7}$

- $j = 1$, best increasing sequence $[\mathbf{1}]$ can add **15**
- $j = 2$, best increasing sequence $[1, \mathbf{3}]$ can add **15**
- $j = 3$, best increasing sequence $[2, 8, \mathbf{10}]$ can add **15**
- $j = 4$, best increasing sequence $[2, 8, 10, \mathbf{11}]$ can add **15**

$\mathbf{1 < 3 < 10 < 11 < 15}$ so $\boldsymbol{\ell[7] = 5}$ and we have the $j = 5$ sequence $[2, 8, 10, 11, 15]$

Bonus: a faster algorithm

As before, $\boldsymbol{\ell}[\boldsymbol{i}]$=of a longest increasing subsequence of $A[1..i]$

Idea: we consider the "best" increasing sequences in $A[1..i]$

- can have several increasing sequence of length $\boldsymbol{j}$ for each $\boldsymbol{j = 1, \ldots, \ell[i]}$
- for any j, **best** increasing sequence of length j: one whose **last entry** is the **smallest**

Example: $A = [\mathbf{2}, \mathbf{8}, \mathbf{10}, \mathbf{11}, 1, 3, \mathbf{0}]$, $\boldsymbol{\ell[6] = 4}$, done $\boldsymbol{i = 6}$

- $j = 1$, best increasing sequence $[1]$ can't add **0**
- $j = 2$, best increasing sequence $[1, 3]$ can't add **0**
- $j = 3$, best increasing sequence $[2, 8, 10]$ can't add **0**
- $j = 4$, best increasing sequence $[2, 8, 10, 11]$ can't add **0**

Bonus: a faster algorithm

As before, $\boldsymbol{\ell}[\boldsymbol{i}]$=of a longest increasing subsequence of $A[1..i]$

Idea: we consider the "best" increasing sequences in $A[1..i]$

- can have several increasing sequence of length $\boldsymbol{j}$ for each $\boldsymbol{j = 1, \ldots, \ell[i]}$
- for any j, **best** increasing sequence of length j: one whose **last entry** is the **smallest**

Example: $A = [\mathbf{2}, \mathbf{8}, \mathbf{10}, \mathbf{11}, 1, 3, \mathbf{0}]$, $\boldsymbol{\ell[6] = 4}$, doing $\boldsymbol{i = 7}$

- $j = 1$, best increasing sequence $[\mathbf{1}]$ can't add $\mathbf{0}$
- $j = 2$, best increasing sequence $[1, \mathbf{3}]$ can't add $\mathbf{0}$
- $j = 3$, best increasing sequence $[2, 8, \mathbf{10}]$ can't add $\mathbf{0}$
- $j = 4$, best increasing sequence $[2, 8, 10, \mathbf{11}]$ can't add $\mathbf{0}$

$\mathbf{0 < 1 < 3 < 10 < 11}$ so $\boldsymbol{\ell[7] = 4}$ and we update the $j = 1$ sequence to $[0]$

Iterative algorithm

Remarks

- sufficient to store **the last entry in each best increasing sequence**
- these last entries are increasing $(\mathbf{1} < \mathbf{3} < \mathbf{10} < \mathbf{11})$
- so we can use binary search to find where the new $A[i]$ fits

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

1. $b \leftarrow [-\infty, \infty, \dots, \infty]$, $\ell \leftarrow 0$ indexed starting from 0
2. **for** $i = 1, \dots, n$ **do**
3. find $k \in \{0, \dots, \ell\}$ such that $b[k] < A[i] \leq b[k+1]$
4. $b[k+1] \leftarrow A[i]$
5. **if** $k = \ell$ **then** ℓ++
6. **return** ℓ

Runtime: $O(n \log(n))$

Longest common subsequence

The problem

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

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

Exercise

an algorithm for longest **common** subsequence can be used for longest **increasing** subsequence

A bivariate recurrence

Definition: let $M[i, j]$ be the length of a 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 $\boldsymbol{\Theta(mn)}$

Bonus: if $A[i] = B[j]$, no need to consider $M[i, j-1]$ and $M[i-1, j]$

Edit distance

The problem

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

Output: minimum number of {add, delete, change} operations that turn A into B

Example: $A =$**snowy**, $B =$**sunny**

s	n	o	w	y
s	u	n	n	y

3C

s	_	n	o	w	y
s	u	n	n	y	_

1A, 2C, 1D

_	s	n	o	w	y	_
s	u	n	_	_	n	y

2A, 2C, 2D

Examples: DNA sequences made of **a**, **c**, **g**, **t**

The recurrence

Definition: let $\boldsymbol{D[i,j]}$ be the edit distance between $\boldsymbol{A[1..i]}$ and $\boldsymbol{B[1..j]}$

- $D[0,j] = j$ for all j (add j characters to empty A)
- $D[i,0] = i$ for all i (delete i characters from A)
- $D[i,j]$ is the min of **three** values
  - $D[i-1,j-1]$ (if $A[i] = B[j]$) or $D[i-1,j-1]+1$ (otherwise)
  - $D[i-1,j]+1$ (delete $A[i]$ and match $A[1..i-1]$ with $B[1..j]$)
  - $D[i,j-1]+1$ (add $B[j]$ and match $A[1..i]$ with $B[1..j-1]$)

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