

F, 09/26/19

Fall'19 CSCE 629

Analysis of Algorithms

Fang Song

Texas A&M U

Lecture 12

- Longest common subsequence

Credit: based on slides by A.Smith and K.Wayne

Logistics

■ Monday 09/30

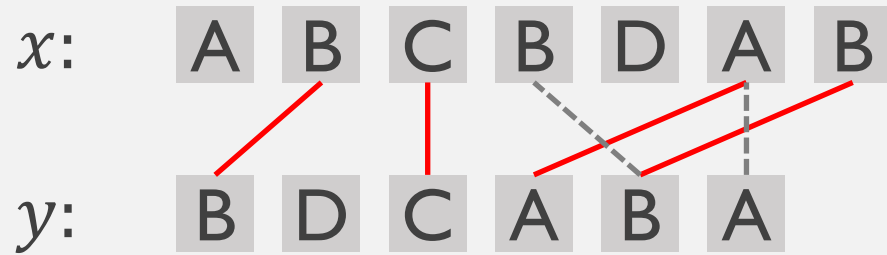
- Recitation by TA on HW problems
- Office hours rescheduled: Tuesday 10/01 1:30pm – 3:30pm

■ Mid-term

- Take home
- When: week 7 (release after 10/09 due M 10/14)
- Practice problems will be posted ahead of time
- Submission: scan of clear hand-writing accepted
- No collaboration of any form (e.g., Google) permitted
- More to be announced

Longest common subsequence (LCS)

- **Input.** Two subsequences $x[1, \dots, m]$ and $y[1, \dots, n]$
- **Output.** A **longest** subsequence common to both.

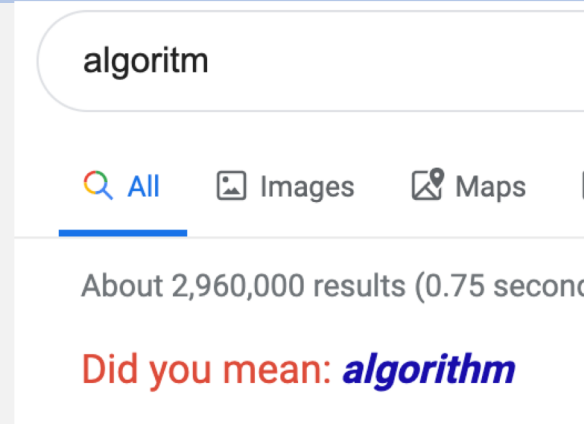


- **Other names you may heard of**
 - Sequence alignment
 - Edit distance: $n - \text{length}(LCS(x, y))$
 - ...

Motivation

■ String matching [Levenshtein 1965]

- Auto corrector
- Spell checker
- Speech recognition
- Machine translation



■ Computational biology [Needleman-Wunsch, 1970's]

- simple measure of genome similarity

ACGTACGTACGTACGTACGTACGTATCGTACGT

AACGTACGTACGTACGTACGTACGTACGTACGT

ACGTACGTACGTACGTACGTACGTACGTA T ATCGTACGT
AACGTACGTACGTACGTACGTACGTACGTA ATCGTACGT

DP1: develop a recursion

- **Input.** Two subsequences $x[1, \dots, m]$ and $y[1, \dots, n]$
- **Output.** A **longest** subsequence common to both.
 - (**Simplification**) Look at the **length** of a longest-common subsequence
 - Extend the algorithm to find the LCS itself

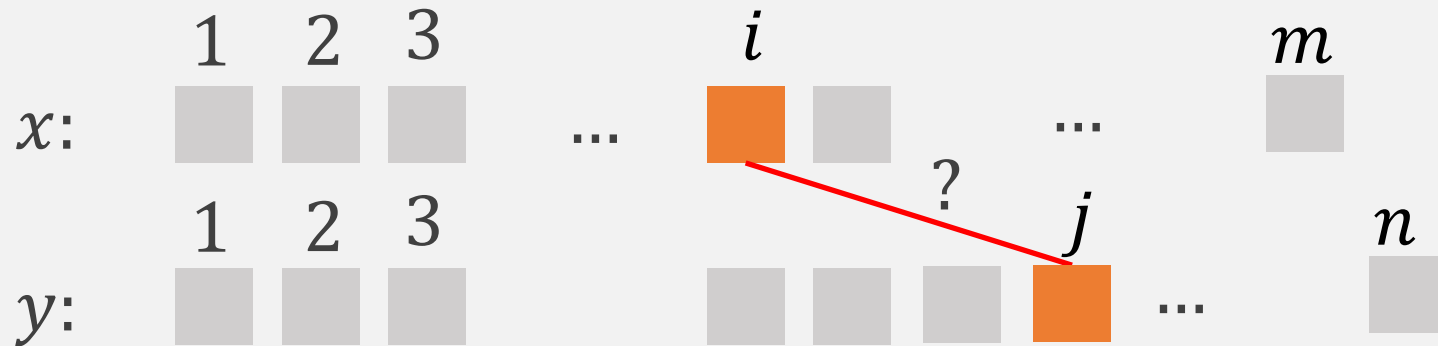
Notation. Denote the length of a sequence s by $|s|$.

Def. $c(i, j) := |\text{LCS}(x[1, \dots, i], y[1, \dots, j])|$

- **Goal.** Find $c(m, n)$
- **Basis:** $c(i, j) = 0$ if $i = 0$ or $j = 0$
- **Recursion:** how to define $c(i, j)$ recursively?

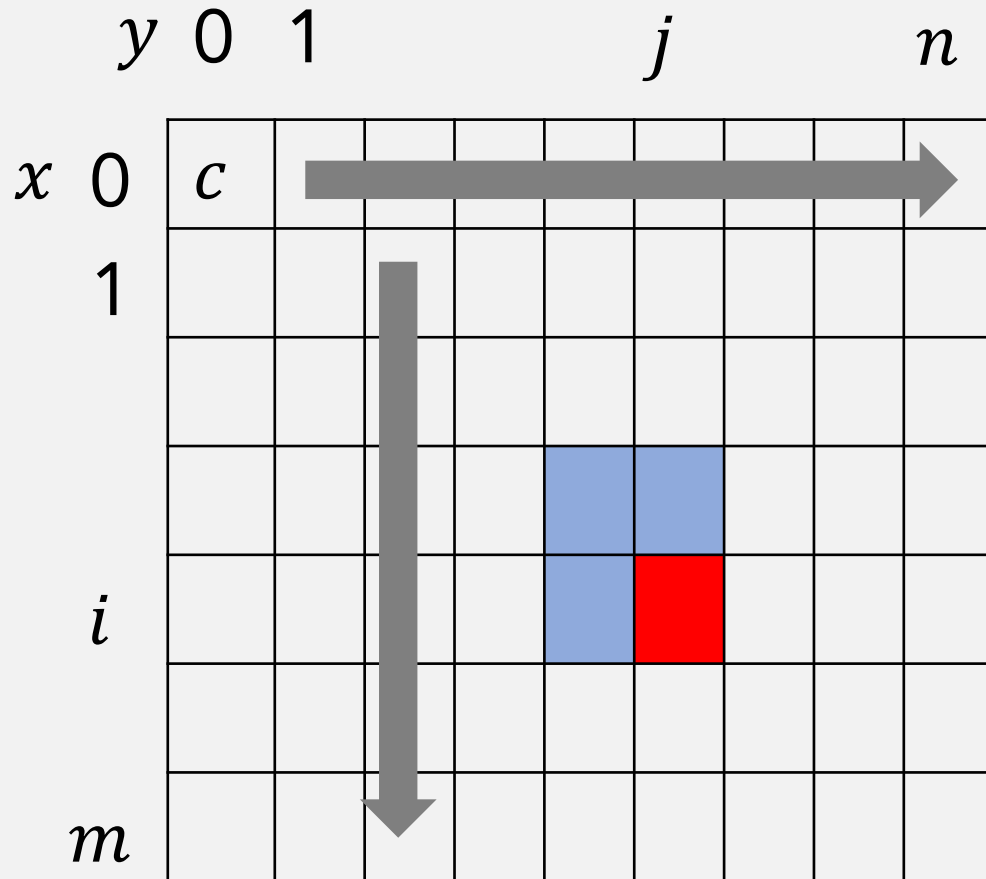
DP1: develop a recursion

- **Case1:** $x[i] = y[j]$ $c(i, j) = c(i - 1, j - 1) + 1$
- **Case2:** $x[i] \neq y[j]$ $c(i, j) = \max\{c[i - 1, j], c[i, j - 1]\}$



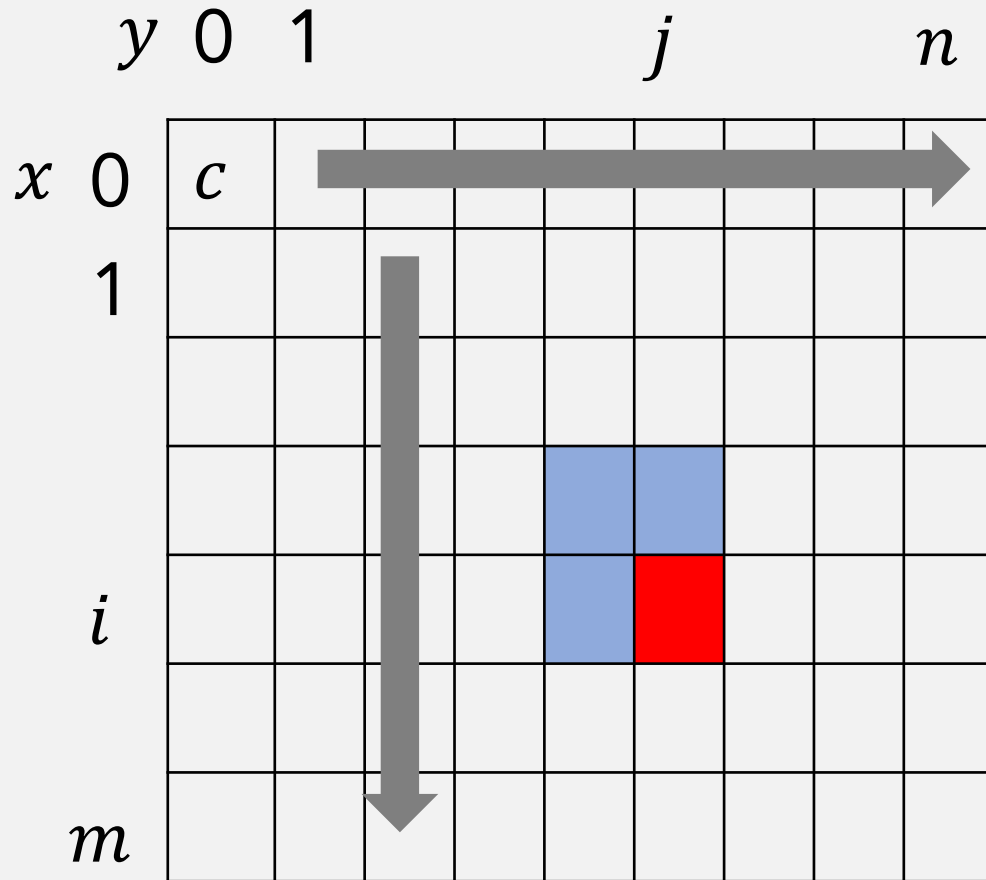
$$c(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c(i - 1, j - 1) + 1 & \text{if } x[i] = x[j] \\ \max \{c(i - 1, j), c(i, j - 1)\} & \text{if } x[i] \neq x[j] \end{cases}$$

DP2: build up solutions



- Subproblems: $O(mn)$
- Memoization data structure
 - 2-D array $c[0, \dots, m, 0, \dots, n]$
- Dependencies
 - Each $c(i, j)$ depends on its three neighbors $c(i-1, j-1)$, $c(i, j-1)$, $c(i-1, j)$
- Evaluation order
 - Left-to-right, row by row

DP2: build up solutions



- Running time: $O(mn)$

```

LCSLen( $x[1, \dots, m], y[1, \dots, n]$ )
//  $c(i, j)$  memoize subproblem values
For  $j = 0, \dots, n$ 
     $c[0, j] \leftarrow 0$ 
For  $i = 1, \dots, m$  // row by row
     $c[i, 0] \leftarrow 0$ 
    For  $j = 1, \dots, n$  // left to right
        If  $x[i] = y[j]$ 
             $c(i, j) = c(i - 1, j - 1) + 1$ 
        Else
             $c(i, j) = \min\{c(i, j - 1), c(i - 1, j)\}$ 
    
```


Example

	y	A	B	C	B	D	A	B
x	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0							
A	0							
B	0							
A	0							4

DP3: constructing an optimal solution

- Reconstruct LCS by tracing backwards

$$LCS(x, y) = \text{BCBA}$$

NB. Multiple solutions are possible.

- Space: $O(mn)$

- Can you do it in $\min\{m, n\}$? (Hint: divide and conquer)

	<i>y</i>	A	B	C	B	D	A	B
<i>x</i>	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
B	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4

Further development

[MasekPaterson']CSSI980] $O(n^2/\log n)$

A Faster Algorithm Computing String Edit Distances*

How about $O(n^{1.9999})$?

Quadratic Barrier

[BI'STOC2015]

Edit Distance Cannot Be Computed
in Strongly Subquadratic Time
(unless SETH is false)

[BEG'SODA2018]

Approximating Edit Distance in Truly Subquadratic Time: Quantum
and MapReduce*†

[CDGKS'FOCS2018]

2018 IEEE 59th Annual Symposium on Foundations of Computer Science

Approximating Edit Distance Within Constant Factor in Truly Sub-Quadratic Time

[HRS'STOC2019]

Near-Linear Time Insertion-Deletion Codes and
(1+ ϵ)-Approximating Edit Distance via Indexing

[BGHS'STOC2019]

$1 + \epsilon$ Approximation of Tree Edit Distance in Quadratic Time*†