Algorithm Design & Analysis (CSE222)

Lecture-10

Recap

- Matrix Chain multiplication
  - $A \in R^{5x3}$, $B \in R^{3x8}$ and $C \in R^{8x2}$.
  - $A \cdot (B \cdot C)$ is more efficient than $(A \cdot B) \cdot C$
- Table Filling (Bottom Up)
- Memoization (Top Down)

Outline

- Longest Common Subsequence
- Edit Distance
- Maximum Job

Computational Biology

DNA (consists of base) sequencing has played crucial role in computational biology.

- Gene identification, prediction
- Gene variations, uncovers evolution
- Personalized medications

Problem

Given two DNA sequences, compare the similarity between them.

- Check if two sequences are equal.
- Check if one sequence is a substring of another sequence.
- Find the longest sequence such that the bases in it are also present in the original sequence in the same order.

Example S_1 = {a**a**g**ct**c**g**} and S_2 = {g**act**a**g**} S_3 = {actg}

Note that S_3 is not a continuous sequence in either S_1 or S_2.

Longest Common Subsequence

Given a sequence X = <x_1, x_2, ..., x_m>, a sequence Z = <z_1, z_2, ..., z_k> is a *subsequence* of X if there is an increasing sequence <i1, i2, ..., ik> of X such that, $x_{ij} = z_j$ for every j = {1, 2, ..., k}.

x = <a b a c d>

z = <b c d>

z = <a a c>

z = <d c>

Longest Common Subsequence

Given a sequence $X = \langle x_1, x_2, \ldots, x_m \rangle$, a sequence $Z = \langle z_1, z_2, \ldots, z_k \rangle$ is a *subsequence* of X if there is an increasing sequence $\langle i1, i2, \ldots, ik \rangle$ of X such that, $x_{ij} = z_j$ for every $j = \{1, 2, \ldots, k\}$.

Given a sequence $X = \langle x_1, x_2, \ldots, x_m \rangle$, i^{th} prefix of X is $X_i = \langle x_1, x_2, \ldots, x_i \rangle$ for $i = \{0, 1, 2, \ldots, m\}$.

Given two sequences X and Y, Z is a *common subsequence* if it is a subsequence of both X and Y.

Longest such common subsequence is called the *longest common subsequence*.

Given two sequences X and Y, find its longest common subsequence.

Problem

Given two sequences X = $<x_1, x_2, \ldots, x_m>$ and Y = $<y_1, y_2, \ldots, y_n>$, find its longest common subsequence.

<u>Brute force</u>

1. If m < n, then go over every possible subset of X and return the longest subset that is a subsequence of Y.
2. Else, then go over every possible subset of Y and return the longest subset that is a subsequence of X.

Running time: $\min(2^m, 2^n)$

Claim

Given two sequences $X = <x_1, x_2, \ldots, x_m>$ and $Y = <y_1, y_2, \ldots, y_n>$, let $Z = <z_1, z_2, \ldots, z_k>$ be its longest common subsequence.

- If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is the LCS of X_{m-1} and Y_{n-1}.
- If $x_m \neq y_n$, then $z_k \neq x_m$ implies Z is the LCS of X_{m-1} and Y.
- If $x_m \neq y_n$, then $z_k \neq y_n$ implies Z is the LCS of X and Y_{n-1}.

Proof

Given two sequences $X = <x_1, x_2, \ldots, x_m>$ and $Y = <y_1, y_2, \ldots, y_n>$, let $Z = <z_1, z_2, \ldots, z_k>$ be its longest common subsequence.

- If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is the LCS of X_{m-1} and Y_{n-1}.

Proof by contradiction

- Let z_k of Z is not equal to x_m.
- Since, x_m and y_n are the equal (by assumption), so we can append x_m with Z and get LCS of X and Y of length k+1.
- This contradicts that $Z = <z_1, z_2, \ldots, z_k>$ is the LCS.
- Let W be LCS of X_{m-1} and Y_{n-1} such that it is greater than k-1, then we can append x_m with W and get LCS of X and Y of length k (Contradiction).
- Hence, Z_{k-1} is LCS of X_{m-1} and Y_{n-1}.

Proof

Given two sequences $X = <x_1, x_2, ..., x_m>$ and $Y = <y_1, y_2, ..., y_n>$, let $Z = <z_1, z_2, ..., z_k>$ be its longest common subsequence.

- If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is the LCS of X_{m-1} and Y_{n-1}.
- If $x_m \neq y_n$, then $z_k \neq x_m$ implies Z is the LCS of X_{m-1} and Y.

Proof by contradiction

- Since, $z_k \neq x_m$ so at most z_k is equal to x_{m-1} and y_n.
- Let W be LCS of X_{m-1} and Y such that it is greater than k, W is also LCS of X and Y of length k (Contradiction).
- Hence, Z is LCS of X_{m-1} and Y.

Proof

Given two sequences $X = <x_1, x_2, \dots, x_m>$ and $Y = <y_1, y_2, \dots, y_n>$, let $Z = <z_1, z_2, \dots, z_k>$ be its longest common subsequence.

- If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is the LCS of X_{m-1} and Y_{n-1}.
- If $x_m \neq y_n$, then $z_k \neq x_m$ implies Z is the LCS of X_{m-1} and Y.
- If $x_m \neq y_n$, then $z_k \neq y_n$ implies Z is the LCS of X and Y_{n-1}.

Proof by contradiction – same as previous case.

Recursive Solution

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

Running time:

Table filling or Memoization?

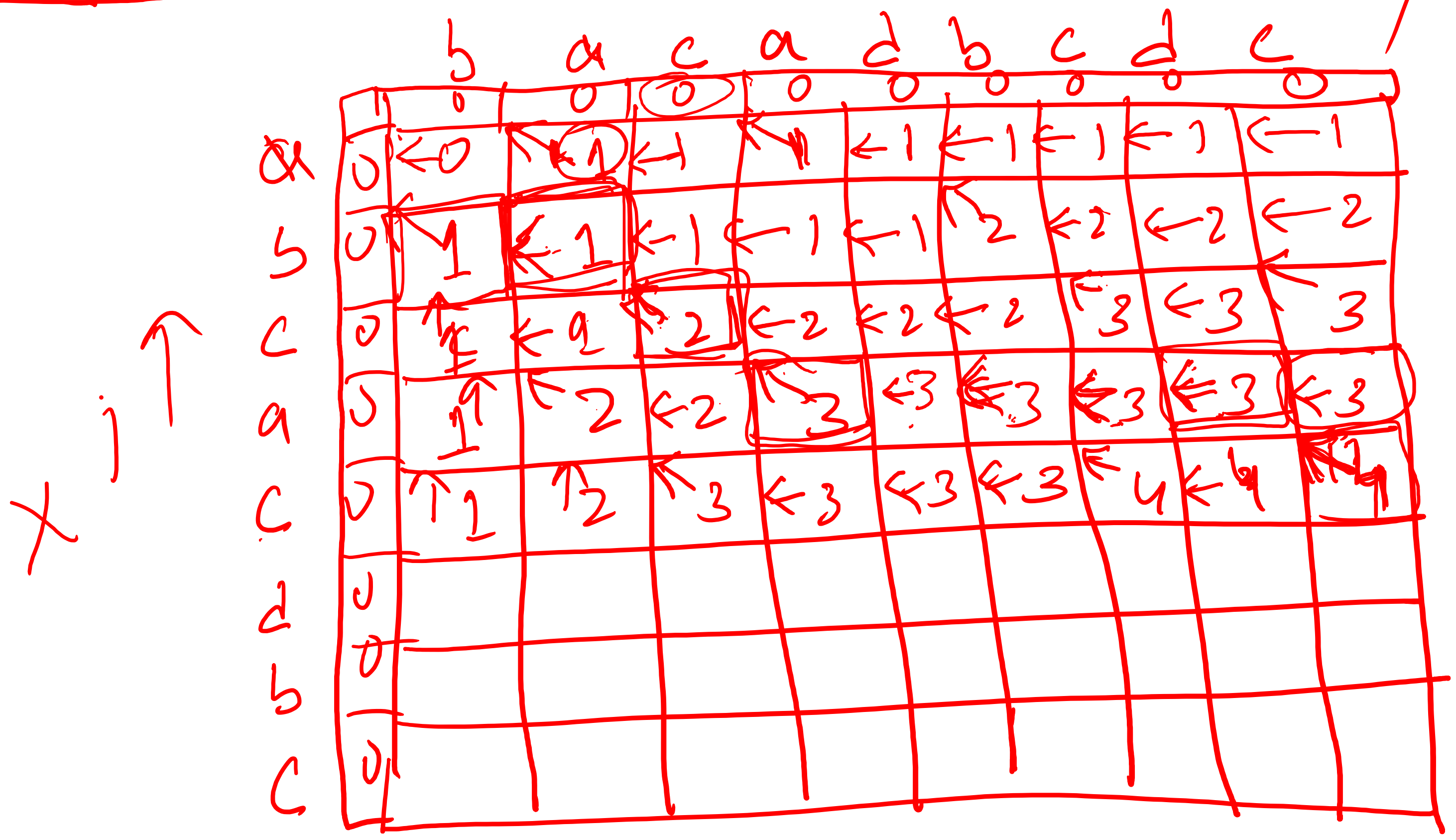
LCS-LENGTH(X, Y)

```
m = X.length
n = Y.length
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i-1, j-1] + 1
            b[i, j] = "↖"
        elseif c[i-1, j] ≥ c[i, j-1]
            c[i, j] = c[i-1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j-1]
            b[i, j] = "←"
return c and b
```

Example

X = <a, b, c, a, c, d, b, c> and Y = <b, a, c, a, d, b, c, d, c>

c|b



bca

Z= bcac

Outline

- Longest Common Subsequence
- Edit Distance
- Maximum Job

Levenshtein (Edit) Distance

Given two strings, <u>edit distance</u> measures the number of operations needed to transform one string into another.

<u>Operations</u>

- Insert
- Replace
- Delete

Input: ‘horse’ and ‘rose’
Replacing ‘h’ with ‘r’ results to ‘rorse’
Deleting ‘r’ results to ‘rose’.

Edit Distance

Input: A = horse and B = rose

Case-1: Do nothing
If A[5] equal to B[4]
then transform A[0-4] → B[0-3].

Case-2: Insert
If A[3] not equal to B[3]
then transform A[0-3] → B[0-2].

Case-3: Replace
If A[3] not equal to B[3]
then transform A[0-2] → B[0-2].

Case-4: Delete
If A[3] not equal to B[3]
then transform A[0-2] → B[0-3].

Example

Input: A = horse and B = rose

Maintain the following table

Case 1

case 4: delete 'r'

Cas 3: replace 'h' with 'r'

edit distance

	' '	h	o	r	s	e
' '	$\boxed{0}$	← 1	← 2	← 3	← 4	← 5
r	↑ 1	$\boxed{\nwarrow 1}$	← 2	↖ 2	← 3	← 4
o	↑ 2	↑ 2	$\boxed{\nwarrow 1}$	$\boxed{\leftarrow 2}$	← 3	← 4
s	↑ 3	↑ 3	↑ 2	↖ 2	$\boxed{\nwarrow 2}$	← 3
e	↑ 4	↑ 4	↑ 3	↑ 3	↑ 3	$\boxed{\nwarrow 2}$

Outline

- Longest Common Subsequence
- Edit Distance
- Maximum Job

Maximum Job

There are two machines A and B. At any given minute we can run job in either machine A or machine B. At ith minute we can run ai steps in machine A or bi steps in machine B. We can move jobs from one machine to another but it will cost us one minute, i.e., no processing for one minute.

Goal: Design an algorithm that determines the maximum number of steps that can be executed in ‘n’ minutes.

Subproblem

Cost(k, A) = The maximum number of steps that can be executed in minutes {1, 2, ..., k} such that machine A executes instruction at minute k.

Cost(k, B) = The maximum number of steps that can be executed in minutes {1, 2, ..., k} such that machine B executes instruction at minute k.

$$\text{cost}(k, A) = \max \begin{cases} a_k + \text{cost}(k-1, A) \\ a_k + \text{cost}(k-2, B) \end{cases}$$

$$\text{cost}(k, B) = \max \begin{cases} b_k + \text{cost}(k-1, B) \\ b_k + \text{cost}(k-2, A) \end{cases}$$

Base Cases

Cost(1, A) = a_1

Cost(1, B) = b_1.

Cost(2, A) = $a_1 + a_2$

Cost(2, B) = $b_1 + b_2$.

Solving Final Problem

$$\max \begin{cases} \text{cost}(n, A), \\ \text{cost}(n, B) \end{cases}$$

Algorithm

$\textsc{Maximum Job}(A, B, n)$

$\text{cost}[1,1] = a_1;\ \text{cost}[1,2] = b_1$

$\text{cost}[2,1] = a_1 + a_2;\ \text{cost}[2,2] = b_1 + b_2$

for $i = 3 \cdots n$

$$\text{cost}(i, A) = \max \begin{cases} a_i + \text{cost}(i-1, A), \\ a_i + \text{cost}(i-2, B) \end{cases}$$

$$\text{cost}(i, B) = \max \begin{cases} b_i + \text{cost}(i-1, B), \\ b_i + \text{cost}(i-2, A) \end{cases}$$

return $\max \begin{cases} \text{cost}(n, A), \\ \text{cost}(n, B) \end{cases}$

Reference

Slides

Introduction to Algorithms by CLRS - Chp-15.4