Longest Common Subsequence

[Dynamic Programming]

Dr Dayal Kumar Behera

Sub-string vs Sub-sequence

Sub-string	Sub-sequence
A string that is part of a longer string	A sequence that is part of a longer sequence.
Ordered and symbols are consecutive	Ordered but symbols are not necessarily consecutive
X = “abcdef” Y = “bcd” Y is a sub-string of X	X = “abcdef” Y = “bde” Y is a sub-sequence of X

Longest Common Subsequence

DNA Strand or chain is expressed as a string over a finite set {A, C, G, T}

S1 = ACCGGTCGAGTCGCGCGGAAGCCGGCCGAA

S2 = GTCCGTTCGGAATGCCGTTGCTCTGTAAA

We want to determine how similar these DNA strands are:

- A common approach is to check if one strand is a subsequence of the other.
- In other words, we're looking for a third strand, S3, where the bases in S3 appear in both S1 and S2.
- These bases must appear in the same order, but not necessarily consecutively.
- The longer the strand S3, the more similar the DNA strands are.

Longest Common Subsequence

Formal Definition : Subsequence

Given a Sequence X = <x_1,x_2,x_3.......x_n>

another sequence Z = <z_1,z_2,z_3.......z_k> is a subsequence of X if there exist a strictly increasing sequence <i_1,i_2,i_3.......i_k> of indices of X such that for all j = 1, 2,...k we have $x_{i_j} = z_j$

Example

X = <A, B, C, B, D, A, B>

Z = <B, C, D, B> is a subsequence of X with index sequence (2, 3, 5, 7)

$x_{i_1} = z_1$ ➔ $x_2 = z_1$ for $i_1 = 2, Z\ is\ found\ in\ X$

Longest Common Subsequence

Formal Definition : Common Subsequence

Given two Sequences X and Y, we say that a sequence Z is a common subsequence of X and Y iff Z is a subsequence of both X and Y.

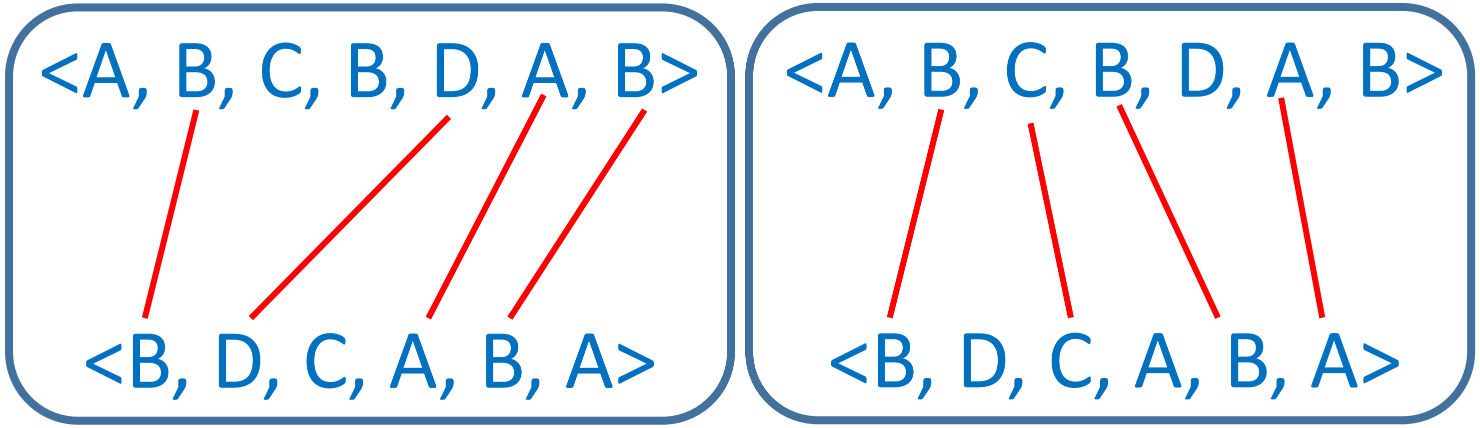
For Example: Given two sequences

X = <A, B, C, B, D, A, B>   Y =<B, D, C, A, B, A>

Common Subsequence

Z1= <B, C, B, A>

Z2= <B, D, A, B>



Longest Common Subsequence(LCS) : Maximum length common subsequence.

Longest Common Subsequence

LCS Problem: We are given two sequences $X = <x_1, x_2, x_3 \ldots\ldots x_m>$ and $Y = <y_1, y_2, y_3 \ldots\ldots y_n>$ and find the maximum length common subsequence of X and Y.

Brute-force Approach: We will find all subsequence of X and check each subsequence to see whether it is also a subsequence of Y. (Keeping track of the longest sequence)

Note: As there are 2^m possible sub sequences of X, It is going to take exponential time.

Theorem

Optima Sub-Structure of LCS

Let $X = <x_1,x_2,x_3.......x_m>$ and $Y = <y_1,y_2,y_3.......y_n>$ be sequences. And let $Z = < z_1,z_2,3_3.......z_k >$ be any LCS of X and Y

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

Proof

$X = \boxed{x_1, x_2, x_{3....}x_{m-1}}\boxed{x_m}$ $Y = \boxed{y_1, y_2, y_{3....}y_{n-1}}\boxed{y_n}$

$Z = \boxed{z_1, z_2, z_{3....}z_{k-1}}\boxed{z_k}$

$x_m = y_n$

X $\boxed{B\ A\ C}$

Y $\boxed{D\ B\ C}$

Z $\boxed{B\ C}$

$x_m \neq y_n$

$x_m \neq y_n$ and y_n not in LCS

X	**A B C**
Y	**B C A**
Z	**B C**

***Z* is an LCS of *X* and Y_{n-1}**

Z is the LCS of X[1, 2, 3] and Y[1, 2]

$x_m \neq y_n$ and x_m not in LCS

X	**B C A**
Y	**A B C**
Z	**B C**

Z* is an LCS of X_{m-1} and *Y

Z is the LCS of X[1, 2] and Y[1, 2, 3]

X_i and Y_j end with x_i=y_j

Let X_i denote the *ith prefix* x[1..*i*] of x[1..*m*], and X_0 denotes an empty prefix

X_i	$x_1\ x_2\ \ldots\ x_{i-1}$	x_i
Y_j	$y_1\ y_2\ \ldots\ y_{j-1}$	$y_j=x_i$
Z_k	$z_1\ z_2 \ldots z_{k-1}$	$z_k=y_j=x_i$

$x_m = y_n$

X	B A C
Y	D B C
Z	B C

Z_k is Z_{k-1} followed by $z_k = y_j = x_i$ where Z_{k-1} is an LCS of X_{i-1} and Y_{j-1} and
LenLCS(i, j)=LenLCS(i-1, j-1)+1

X_i and Y_j end with $x_i \neq y_j$

X_i $\boxed{x_1\ x_2\ \dots\ x_{i-1}\ \ x_i}$

Y_j $\boxed{y_1\ y_2\ \dots\ y_{j-1}}\boxed{y_j}$

Z_k $\boxed{z_1\ z_2 \dots z_{k-1}}\boxed{z_k \neq y_j}$

Z_k is an LCS of X_i and Y_{j-1}

X_i $\boxed{x_1\ x_2\ \dots\ x_{i-1}}\boxed{x_i}$

Y_j $\boxed{y_j\ y_1\ y_2\ \dots y_{j-1}\ \ y_j}$

Z_k $\boxed{z_1\ z_2 \dots z_{k-1}}\boxed{z_k \neq x_i}$

Z_k is an LCS of X_{i-1} and Y_j

$$LenLCS(i, j) = \max\{LenLCS(i, j-1), LenLCS(i-1, j)\}$$

Recursive Approach

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

$$\begin{array}{cccc} & & & \mathbf{y_j} \\ & & j-1 & j \\ & i-1 & i-1,\ j-1 & i-1,\ j \\ \mathbf{x_i} & i & i,\ j-1 & i,\ j \end{array}$$

Writing the recurrence equation

- **Let X_i denote the *ith prefix* x[1..i] of x[1..m], and**
- **X_0 denotes an empty prefix**
- **We will first compute the *length of an LCS of X_m and Y_n*, *LenLCS*(m, n), and then use information saved during the computation for finding the actual subsequence**
- **We need a recursive formula for computing *LenLCS*(i, j).**

X = ABCBDAB
Y = BDCABA

LCS Example

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0						
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

LCS Example

i	j	0	1	2	3	4	5	6
		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

LCS Example

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

LCS Example

i	j	0	1	2	3	4	5	6
		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

LCS Example

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0						
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

LCS Example

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	← 0	← 0	← 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	← 1	↖ 2	← 2
3	C	0	↑ 1	← 1	↖ 2	← 2	← 2	← 2
4	B	0	↖ 1	← 1	↑ 2	← 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	← 2	← 2	↑ 3	← 3
6	A	0	↑ 1	↑ 2	← 2	↖ 3	← 3	↖ 4
7	B	0	↑ 1	↑ 2	← 2	↑ 3	↖ 4	← 4

LCS Example

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	← 0	← 0	← 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	← 1	↖ 2	← 2
3	C	0	↑ 1	← 1	↖ 2	← 2	← 2	← 2
4	B	0	↖ 1	← 1	↑ 2	← 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	← 2	← 2	↑ 3	← 3
6	A	0	↑ 1	↑ 2	← 2	↖ 3	← 3	↖ 4
7	B	0	↑ 1	↑ 2	← 2	↑ 3	↖ 4	← 4

LCS Example

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	← 0	← 0	← 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	← 1	↖ 2	← 2
3	C	0	↑ 1	← 1	↖ 2	← 2	← 2	← 2
4	B	0	↖ 1	← 1	↑ 2	← 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	← 2	← 2	↑ 3	← 3
6	A	0	↑ 1	↑ 2	← 2	↖ 3	← 3	↖ 4
7	B	0	↑ 1	↑ 2	← 2	↑ 3	↖ 4	← 4

LCS Length Algorithm

Algo LCS_Length

1. m← X.length
2. n← Y.length
3. Let B[1..m, 1..n] and C[1..m, 1..n] be two tables
4. for i= 0 to m
5. C[i, 0] =0
6. for j= 0 to n
7. C[0, j] =0

```
for i= 1 to m
    for j= 1 to n
        if ( x_i == y_j)
                then  C[i, j] = C[i-1, j-1]+1
                      B[i, j] = "↖"
        else if (C[i-1, j] >= C[i, j-1])
                then  C[i, j] = C[i-1, j]
                      B[i, j] = "↑"
        else   C[i, j] = C[i, j-1]
               B[i, j] = "←"
```

PRINT-LCS(B,X,i,j)

Algo PRINT-LCS(B, X, i, j)

```
if i==0 OR j==0 then return
if B[i, j] == "↖"
        PRINT-LCS(B, X, i-1, j-1)
        print x_i
else if B[i, j] == "↑"
        PRINT-LCS(B, X, i-1, j)
else PRINT-LCS(B, X, i, j-1)
```

LCS Example (Another Solution)

	j	0	1	2	3	4	5	6
i		Y_j	B	D	C	A	B	A
0	X_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	← 1
2	B	0	↖ 1	← 1	← 1	← 1	↖ 2	← 2
3	C	0	↑ 1	← 1	↖ 2	← 2	← 2	← 2
4	B	0	↖ 1	← 1	↑ 2	← 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	← 2	← 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	← 2	↖ 3	← 3	↖ 4
7	B	0	↑ 1	↑ 2	← 2	↑ 3	↖ 4	← 4

LCS Example (Another Solution)

```
for i= 1 to m
    for j= 1 to n
        if ( x_i == y_j)
            then  C[i, j] = C[i-1, j-1]+1
                  B[i, j] = "↖"
        else if (C[i, j-1] >= C[i-1, j])
            then  C[i, j] = C[i, j-1]
                  B[i, j] = "←"
        else   C[i, j] = C[i-1, j]
               B[i, j] = "↑"
```

Another Example

	y_j	b	a	b	a	a	b	a	a	b
x_i	0	0	0	0	0	0	0	0	0	0
a	0	0 ↑	↖ 1	1 ←	↖ 1	↖ 1	1 ←	↖ 1	↖ 1	1 ←
b	0	↖ 1	1 ↑	↖ 2	2 ←	2 ←	↖ 2	2 ←	2 ←	↖ 2
b	0	↖ 1	1 ↑	↖ 2	2 ↑	2 ↑	↖ 3	3 ←	3 ←	↖ 3
a	0	1 ↑	↖ 2	2 ↑	↖ 3	↖ 3	3 ←	↖ 4	↖ 4	4 ←
b	0	↖ 1	2 ↑	↖ 3	3 ←	3 ←	↖ 4	4 ↑	4 ↑	↖ 5
a	0	1 ↑	↖ 2	3 ↑	4	↖ 4	4 ↑	↖ 5	↖ 5	5 ↑
b	0	↖ 1	2 ↑	↖ 3	4 ↑	4 ↑	↖ 5	5 ↑	5 ↑	↖ 6
a	0	1 ↑	↖ 2	3 ↑	↖ 4	↖ 5	5 ↑	↖ 6	↖ 6	6 ↑

Possible Solutions:

abbaab

ababaa

bababa

Thank You