Longest Common Subsequence

A **subsequence** of a string S, is a set of characters that appear in left-to-right order, but not necessarily consecutively.

Example

$$ACTTGCG$$

- ACT , $ATTC$, T , $ACTTGC$ are all subsequences.
- TTA is not a subequence

A **common subequence** of two strings is a subsequence that appears in both strings. A **longest common subequence** is a common subsequence of maximal length.

Example

$$\begin{aligned} S_1 &= AAACCGTGAGTTATTCGTTCTAGAA \\ S_2 &= CACCCCTAAGGTACCTTTGGTTC \end{aligned}$$

Longest Common Subsequence

A subsequence of a string S, is a set of characters that appear in left-to-right order, but not necessarily consecutively.

Example

$$ACTTGCG$$

- ACT , $ATTC$, T , $ACTTGC$ **are all subsequences.**
- TTA **is not a subequence**

A common subequence of two strings is a subsequence that appears in both strings. A longest common subequence is a common subsequence of maximal length.

Example

$$\begin{aligned} S_1 &= AAACCGTGAGTTATTCGTTCTAGAA \\ S_2 &= CACCCCTAAGGTACCTTTGGTTC \end{aligned}$$

Longest Common Subsequence

A **subsequence** of a string S, is a set of characters that appear in left-to-right order, but not necessarily consecutively.

Example

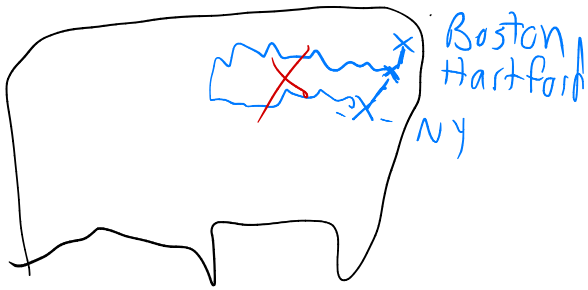
$$ACTTGCG$$

- ACT , $ATTC$, T , $ACTTGC$ are all subsequences.
- TTA is not a subequence

A **common subequence** of two strings is a subsequence that appears in both strings. A **longest common subequence** is a common subsequence of maximal length.

Example

$$\begin{aligned} S_1 &= AAACCGTGAGTTATTCGTTCTAGAA \\ S_2 &= CACCCCTAAGGTACCTTTGGTTC \end{aligned}$$



Example

$$S_1 = AAACCGTGAGTTATTCGTTCTAGAA$$
$$S_2 = CACCCCTAAGGTACCTTTGGTTC$$

LCS is

$$ACCTAGTACTTTG$$

Has applications in many areas including biology.

Algorithm 1

Enumerate all subsequences of S_1, and check if they are subsequences of S_2.

Questions:

- **How do we implement this?**
- **How long does it take?**

Optimal Substructure

Theorem Let $X = < x_1, x_2, \ldots, x_m >$ and $Y = < y_1, y_2, \ldots, y_n >$ be sequences, and let $Z = < z_1, z_2, \ldots, z_k >$ 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 an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}.

A A A T
A C A T

I say LCS does not use the T

Proof

Let $X =< x_1, x_2, \ldots, x_m >$ and $Y =< y_1, y_2, \ldots, y_n >$ be sequences, and let $Z =< z_1, z_2, \ldots, z_k >$ 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 an LCS of X_{m-1} and Y_{n-1}.**
2. **If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.**
3. **If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}.**

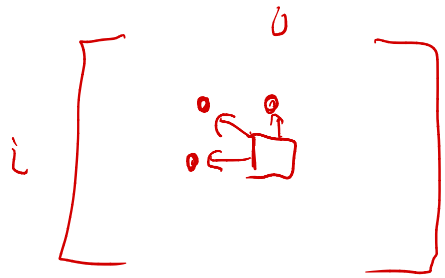
Proof

1. **If $z_k \neq x_m$, then we could append $x_m = y_n$ to Z to obtain a common subsequence of X and Y of length $k+1$, contradicting the supposition that Z is a *longest* common subsequence of X and Y. Thus, we must have $z_k = x_m = y_n$. Now, the prefix Z_{k-1} is a length-$(k-1)$ common subsequence of X_{m-1} and Y_{n-1}. We wish to show that it is an LCS. Suppose for the purpose of contradiction that there is a common subsequence W of X_{m-1} and Y_{n-1} with length greater than $k-1$. Then, appending $x_m = y_n$ to W produces a common subsequence of X and Y whose length is greater than k, which is a contradiction.**
2. **If $z_k \neq x_m$, then Z is a common subsequence of X_{m-1} and Y. If there were a common subsequence W of X_{m-1} and Y with length greater than k, then W would also be a common subsequence of X_m and Y, contradicting the assumption that Z is an LCS of X and Y.**
3. **The proof is symmetric to the previous case.**

Recursion for length

c[i,j] is the longest subsequence of X[1..i] & Y[1..j]

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



Code

```
LCS − Length(X, Y)
    m ← length[X]
    n ← length[Y]
    for i ← 1 to m
            do c[i, 0] ← 0
    for j ← 0 to n
            do c[0, j] ← 0
    for i ← 1 to m
            do for j ← 1 to n
                    do 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] ← "←"
    return c and b
```

i A T T C

j T C A

AT AT

TCA

$O(nm)$