Longest Common Subsequence (LCS)

Chapter 15

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
- Subsequences:
  - ACT
  - ATTC
  - ACTTGC
- TTA is not a subsequence!

Common Subsequence

- A common subsequence of two strings is a subsequence that appears in *both* strings
- Example:
  - ACTTGCG   and   AGTCTCG
- Common subsequences:
  - ATT
  - AGCG

Longest Common Subsequence

- A longest common subsequence is a common subsequence of maximal length
- Example:
  - ACTTGCG and AGTCTCG
- LCS:
  - ACTCG
  - ATTCG

Longest Common Subsequence

- Problem: Let $x_1x_2...x_m$ and $y_1y_2...y_n$ be two sequences over some alphabet.
  - We assume they are strings of characters

- Find a longest common subsequence (LCS) of $x_1x_2...x_m$ and $y_1y_2...y_n$

Example

- $x_1x_2x_3x_4x_5x_6x_7x_8$= b a c b f f c b
- $y_1y_2y_3y_4y_5y_6y_7y_8y_9$ = d a b e a b f b c

- Longest Common Subsequence is:

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

Naïve Algorithm

Dynamic Programming

- LCS can be solved using dynamic programming

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution bottom-up
4. Construct an optimal solution from the computed information

Step 1

- Characterizing a longest subsequence
- Optimal substructure: If $z = z_1 z_2 \ldots z_p$ is a LCS of $x_1 x_2 \ldots x_m$ and $y_1 y_2 \ldots y_n$, then at least one of these most hold
  - $x_m = y_n$, and $z_1 z_2 \ldots z_{p-1}$ is an LCS of $x_1 x_2 \ldots x_{m-1}$ and $y_1 y_2 \ldots y_{n-1}$,
  - $x_m \mathrel{!}= y_n$, and $z_1 z_2 \ldots z_p$ is an LCS of $x_1 x_2 \ldots x_{m-1}$ and $y_1 y_2 \ldots y_n$,
  - $x_m \mathrel{!}= y_n$, and $z_1 z_2 \ldots z_p$ is an LCS of $x_1 x_2 \ldots x_m$ and $y_1 y_2 \ldots y_{n-1}$.

Step 2: Recursive Solution

Let c_{ij} = length of LCS of $x_1x_2 \ldots x_i$ and $y = y_1y_2 \ldots y_j$.

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

$$b[i,j] = \begin{cases} \text{"}\nwarrow\text{"} & \text{if } x_i = y_j, \\ \text{"}\uparrow\text{"} & \text{if } x_i \neq y_j \text{ and } c[i-1, j] \geq c[i, j-1], \\ \text{"}\leftarrow\text{"} & \text{if } x_i \neq y_j \text{ and } c[i-1, j] < c[i, j-1]. \end{cases}$$

We compute the $c[i,j]$ and $b[i,j]$ in order of increasing $i+j$, or alternatively in order of increasing i, and for a fixed i, in order of increasing j.

Steps 3 & 4

```
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

- $x_1x_2x_3x_4x_5x_6x_7x_8$= b a c b f f c b
- $y_1y_2y_3y_4y_5y_6y_7y_8y_9$ = d a b e a b f b c

Example

	0	1 d	2 a	3 b	4 e	5 a	6 b	7 f	8 b	9 c
0										
1 b										
2 a										
3 c										
4 b										
5 f										
6 f										
7 c										
8 b										

Printing the LCS

```
PRINT-LCS(b, X, i, j)
if i == 0 or j == 0
    return
if b[i, j] == "↖"
    PRINT-LCS(b, X, i − 1, j − 1)
    print x_i
elseif b[i, j] == "↑"
    PRINT-LCS(b, X, i − 1, j)
else PRINT-LCS(b, X, i, j − 1)
```

Another Example

- What is the LCS in:
  - epidemiologist
  - refrigeration