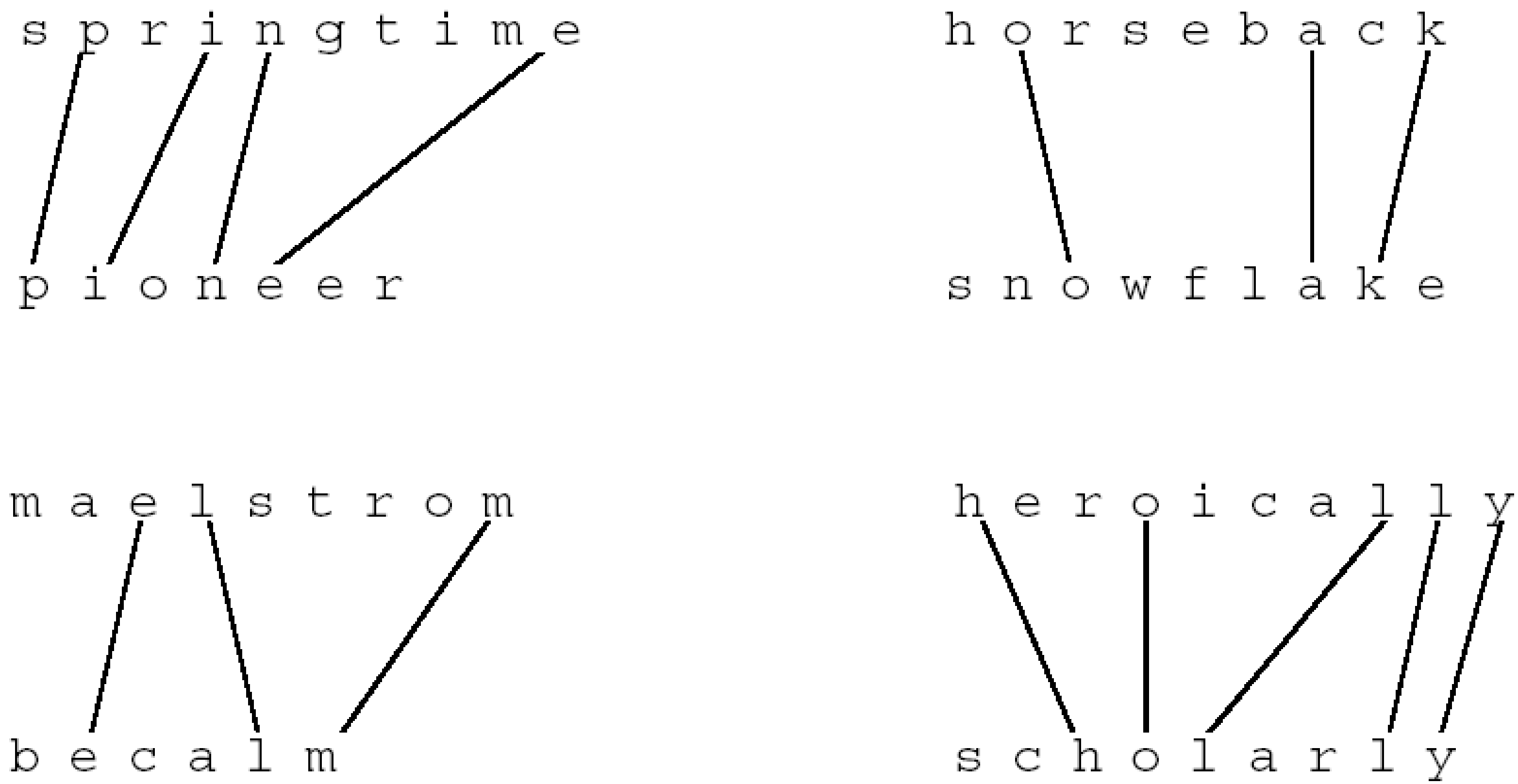
Dynamic Programming –LCS && LIS

Bruce Nan

Longest Common Subsequence

- *Input:* *2 sequences,* $X = x_1, \ldots, x_m$ *and* $Y = y_1, \ldots, y_n$.
- *Output:* *a subsequence common to both whose length is longest.*
- *Note:* *A subsequence doesn't have to be consecutive, but it has to be in order.*

Example 3. Longest Common Subsequence



Brute-force Algorithm

For every subsequence of X, check whether it's a subsequence of Y.

Time: $(n2^m)$.

2^m subsequences of X to check.

Each subsequence takes $\Theta(n)$ time to check:
scan Y for first letter, from there scan for second, and so on.

Step 1. Define Data Structure

- Input: 2 sequences, $X = x_1, \dots, x_m$ and $Y = y_1, \dots, y_n$.

Notation:

$$X_i \;=\; \text{prefix } \langle x_1, \dots, x_i \rangle$$
$$Y_i \;=\; \text{prefix } \langle y_1, \dots, y_i \rangle$$

Let $c(i, j) =$ length of LCS for X_i and Y_j

Ultimately, we are interested in $c(m,n)$.

Step 2. Define Recurrence

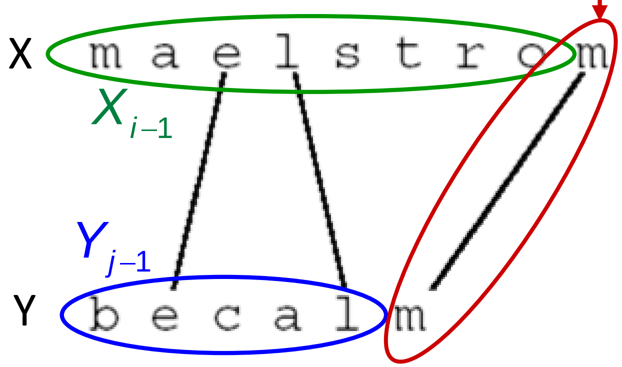
Case 1. Input sequence is empty

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

Recurrence

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

Case 2. Last elements match: must be part of an LCS

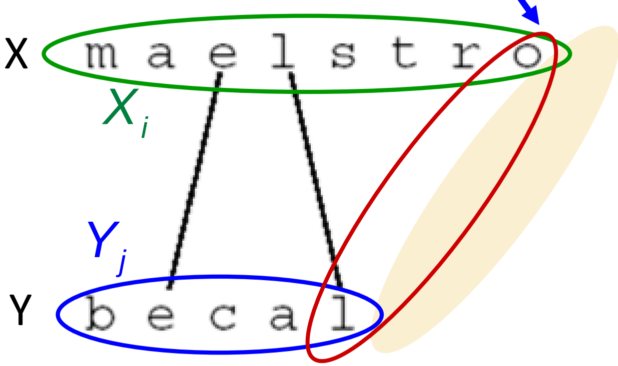


Z ___ ___ ··· ___ ___m

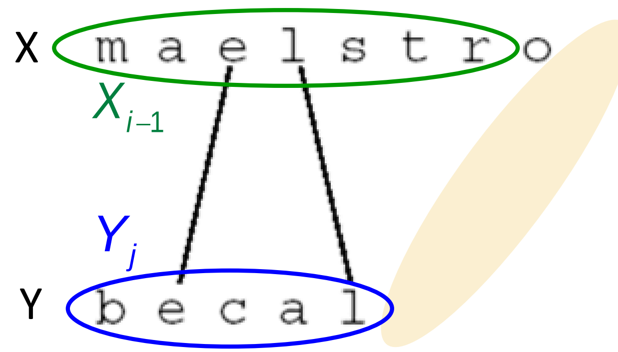
Recurrence

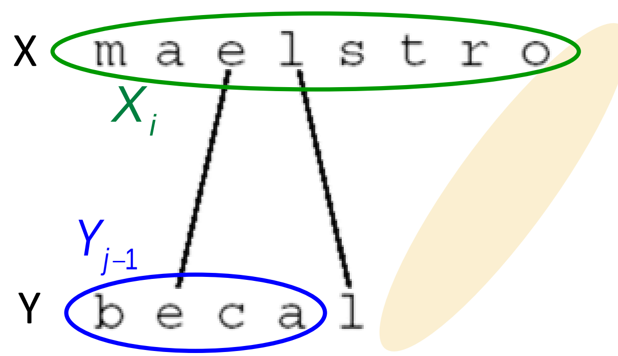
$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0\,, \\ c[i-1,j-1]+1 & \text{if } i,j > 0 \text{ and } x_i = y_j\,, \\ \max(c[i-1,j], c[i,j-1]) & \text{if } i,j > 0 \text{ and } x_i \neq y_j\,. \end{cases}$$

Case 3. Last elements don't match: at most one of them is part of LCS



Choose!





Step 3. Provide an Algorithm

```
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
```

Running time? $O(mn)$

Step 4. Compute Optimal Solution

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

Running time? O(m+n)

- Initial call is PRINT-LCS(b, X, m, n).

Example

- $b[i, j]$ points to table entry whose subproblem we used in solving LCS of X_i and Y_j.

- When $b[i, j] = \nwarrow$, we have extended LCS by one character. So longest common subsequence = entries with $\nwarrow$ in them.

	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	$\uparrow$ 0	$\uparrow$ 0	$\uparrow$ 0	$\nwarrow$ 1	$\leftarrow$ 1	$\nwarrow$ 1
2	B	0	$\nwarrow$ 1	$\leftarrow$ 1	$\leftarrow$ 1	$\uparrow$ 1	$\nwarrow$ 2	$\leftarrow$ 2
3	C	0	$\uparrow$ 1	$\uparrow$ 1	$\nwarrow$ 2	$\leftarrow$ 2	$\uparrow$ 2	$\uparrow$ 2
4	B	0	$\nwarrow$ 1	$\uparrow$ 1	$\uparrow$ 2	$\uparrow$ 2	$\nwarrow$ 3	$\leftarrow$ 3
5	D	0	$\uparrow$ 1	$\nwarrow$ 2	$\uparrow$ 2	$\uparrow$ 2	$\uparrow$ 3	$\uparrow$ 3
6	A	0	$\uparrow$ 1	$\uparrow$ 2	$\uparrow$ 2	$\nwarrow$ 3	$\uparrow$ 3	$\nwarrow$ 4
7	B	0	$\nwarrow$ 1	$\uparrow$ 2	$\uparrow$ 2	$\uparrow$ 3	$\nwarrow$ 4	$\uparrow$ 4

Example 6: Longest Increasing Subsequence

- *Input:* 1 sequence, $X = x_1, \ldots, x_n$.
- *Output:* the longest *increasing* subsequence of X.
- *Note:* A subsequence doesn't have to be consecutive, but it has to be in order.

Step 1. Define an array of values to compute

$\forall i \in \{0, \ldots, n\}$, $\mathrm{A}(i) =$ length of LIS of X ending in x_i.

Ultimately, we are interested in $\max\{A(i) \mid 1 \le i \le n\}$

Step 2. Provide a Recurrent Solution

$$\text{for } 1 \le i \le n,$$

$$A(i) = 1 + \max\{A(j) \mid 1 \le j < i \text{ and } x_j < x_i\}$$

Step 3. Provide an Algorithm

```
function A=LIS(X)
for i=1:length(X)
    m=0;
    for j=1:i-1
        if X(j) < X(i) & A(j) > m
            m=A(j);
        end
    end
    A(i)=m+1;
end
```

Running time? $O(n^2)$

Step 4. Compute Optimal Solution

```
function lis=printLIS(X, A)
[m,mi]=max(A);
lis=printLISm(X,A,mi,'LIS: ');
lis=[lis,sprintf('%d', X(mi))];
```

Running time? O(n)

```
function lis=printLISm(X, A, mi, lis)
if A(mi) > 1
    i=mi-1;
    while ~(X(i) < X(mi) & A(i) == A(mi)-1)
        i=i-1;
    end
    lis=printLISm(X, A, i, lis);
    lis=[lis, sprintf('%d ', X(i))];
end
```

LIS Example

X = 96 24 61 49 90 77 46 2 83 45

A = 1 1 2 2 3 3 2 1 4 2

X =	96	24	61	49	90	77	46	2	83	45
A =	1	1	2	2	3	3	2	1	4	2

> printLIS(X,A)

> LIS: 24 49 77 83