Toward a better algorithm (1)

- Prefix
  - ABCB is a prefix of ABCBDAB
  - BCB is not a prefix of ABCBDAB
- x = ABCBDAB
  - $x[1..4]$: the first four symbols in x, a prefix of x
  - $x[1..4]$ = ABCB

Toward a better algorithm (2)

- Strategy
  - Consider the ***length*** of LCS first
  - Define $|x|$ the length of a sequence x
  - Define $c[i, j] = |\mathrm{LCS}(x[1 \,..\, i], y[1 \,..\, j])|$
  - Then, $c[m, n] = |\mathrm{LCS}(x, y)|$

- Theorem

$$c[i,j] = \begin{cases} c[i-1, j-1] + 1 & \text{if } x[i] = y[j], \\ \max\{c[i-1, j], c[i, j-1]\} & \text{otherwise.} \end{cases}$$

- Example:

		A	B	C	B	D	A	B	m+1
	0	0	0	0	0	0	0	0	
B	0	↑0	↖1	↑1	↖1	↑1	↑1	↖1	length of LCS(B, ABCB)
D	0	↑0	↑1	←1	↑1	↖2	←2	←2	
C	0	↑0	↑1	↖2	←2	↑ 2	↑ 2	↑ 2	
A	0	↖1	↑1	↑ 2	↑ 2	↑ 2	↖3	←3	length of LCS(BDCA, ABCBDA)
B	0	↑1	↖2	↑ 2	↖3	←3	↑ 3	↖4	
A	0	↖1	↑ 2	↑ 2	↑ 3	↑ 3	↖4	↑4	length of LCS(x, y)
n+1									

```
LCS-LENGTH(X, Y)
let b[1..m , 1..n], c[0..m, 0..n] be new arrays
for i = 1 to m
    c[i, 0] = 0                          // initialize c
for i = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_i
            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
```

Time? $O(mn)$

Space? $O(mn)$

see “↖”, print!

		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	0
B	0	↑0	↖1	↑0	↖1	↑0	↑0	↖1
D	0	↑0	↑1	←1	↑1	↖2	←2	←2
C	0	↑0	↑1	↖2	←2	↑ 2	↑ 2	↑ 2
A	0	↖1	↑1	↑ 2	↑ 2	↑ 2	↖3	←3
B	0	↑1	↖2	↑ 2	↖3	←3	↑ 3	↖4
A	0	↖1	↑ 2	↑ 2	↑ 3	↑ 3	↖4	↑ 4

Output: B D A B

PRINT-LCS(*b*, *X*, *i*, *j*)
1.if *i* == 0 or *j* == 0
2. **return**
3.if *b*[*i*, *j*] == ↖
4. PRINT-LCS(*b*, *X*, *i*-1, *j*-1)
5. print x_i
6.elseif *b*[*i*, *j*] == ↑
7. PRINT-LCS(*b*, *X*, *i*-1, *j*)
8.else
9. PRINT-LCS(*b*, *X*, *i*, *j*-1)

Time? $O(m+n)$

Back to the Theorem (1)

$$c[i,j] = \begin{cases} c[i-1,j-1]+1 & \text{if } x[i]=y[j], \\ \max\{c[i-1,j],\ c[i,j-1]\} & \text{otherwise.} \end{cases}$$

- Theorem (Optimal substructure of an LCS)
  - Let $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$
  - $Z = \langle z_1, z_2, \ldots, z_k \rangle$ be a 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}

Improving the code

- Can we eliminate the *b* table?
  - Yes! Determine in $O(1)$ time which of the three values was used to compute $c[i, j]$.
- What's the time and space complexity if we need to compute only the length?
  - Time: $O(mn)$
  - Space: $O(\min(m, n))$