Longest Common Subsequence (LCS)

- **Problem: Given sequences x[1..m] and y[1..n], find a longest common subsequence of both.**
- **Example: x=ABCBDAB and y=BDCABA,**
  - **BCA is a common subsequence and**
  - **BCBA and BDAB are two LCSs**

LCS

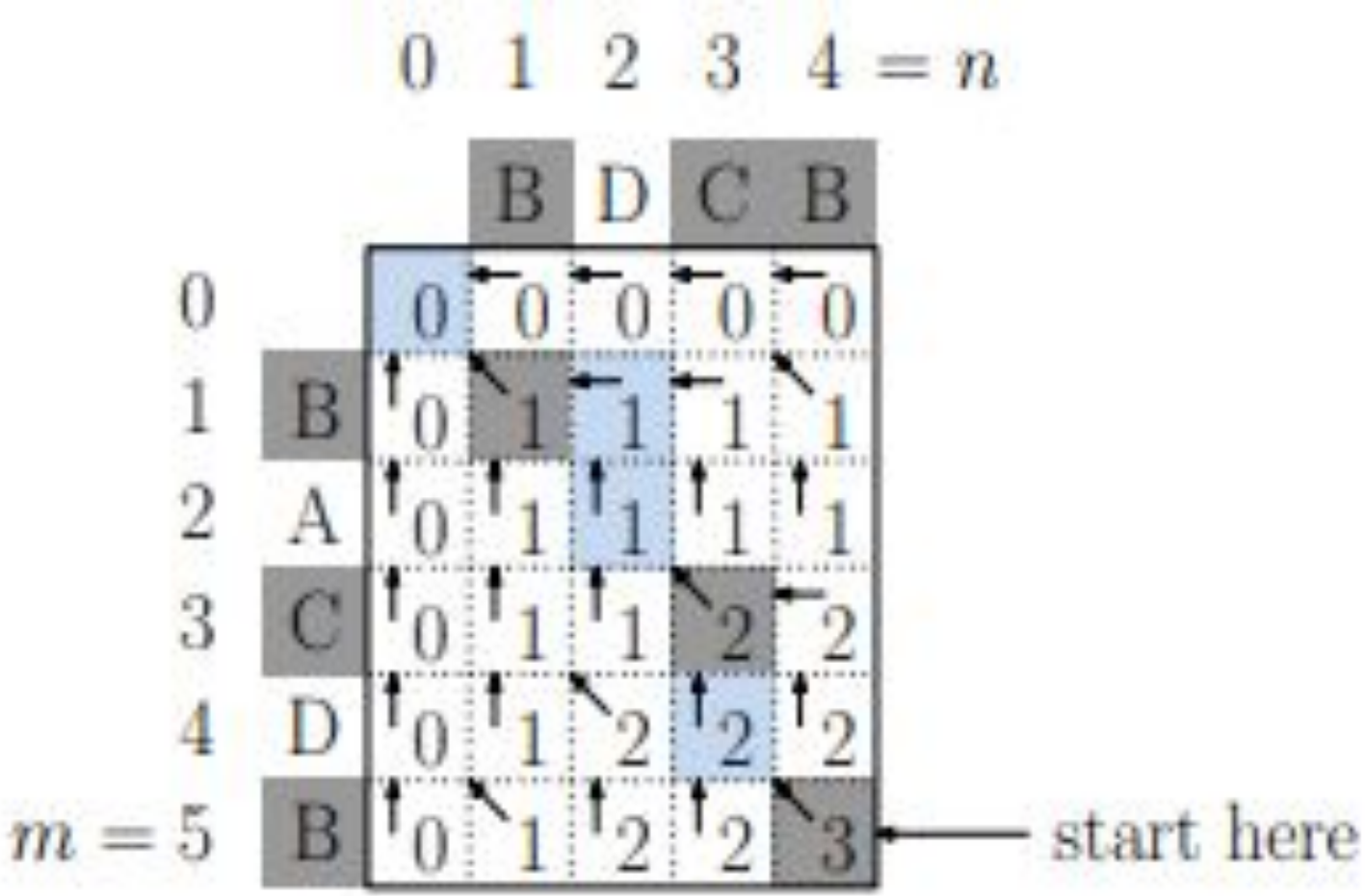
- Brute force solution
- Writing a recurrence equation
- The dynamic programming solution
- Application of algorithm

Brute force solution

- **Solution: For every subsequence of x, check if it is a subsequence of y.**
- **Analysis :**
  - **There are 2^m subsequences of x.**
  - **Each check takes $O(n)$ time, since we scan y for first element, and then scan for second element, etc.**
  - **The worst case running time is $O(n2^m)$.**

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*).


0 1 2 3 4 = n
B D C B
0
1 B
2 A
3 C
4 D
m = 5 B
start here

The recurrence equation

$$lenLCS(i,j) = \begin{cases} 0 & \text{if } i = 0, \text{ or } j = 0 \\ lenLCS(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max\{lenLCS(i-1, j), lenLCS(i, j-1)\} & \text{otherwise} \end{cases}$$

The dynamic programming solution

- **Initialize the first row and the first column of the matrix *LenLCS* to 0**
- **Calculate *LenLCS* (1, *j*) for *j* = 1,…, *n***
- **Then the *LenLCS* (2, *j*) for *j* = 1,…, *n*, etc.**
- **Store also in a table an *arrow* pointing to the array element that was used in the computation.**
- **It is easy to see that the computation is O(*mn*)**

Example

	y_j	B	D	C	A
x_j	0	0	0	0	0
A	0	0	0	0	1
B	0	1	1	1	1
C	0	1	1	2	2
B	0	1	1	2	2

To find an LCS follow the arrows, for each diagonal arrow there is a member of the LCS

LCS-Length(X, Y)

```
m ← length[X}
n ← length[Y]
for i ← 1 to m do
     c[i, 0] ← 0
for j ← 1 to n do
     c[0, j] ← 0
```

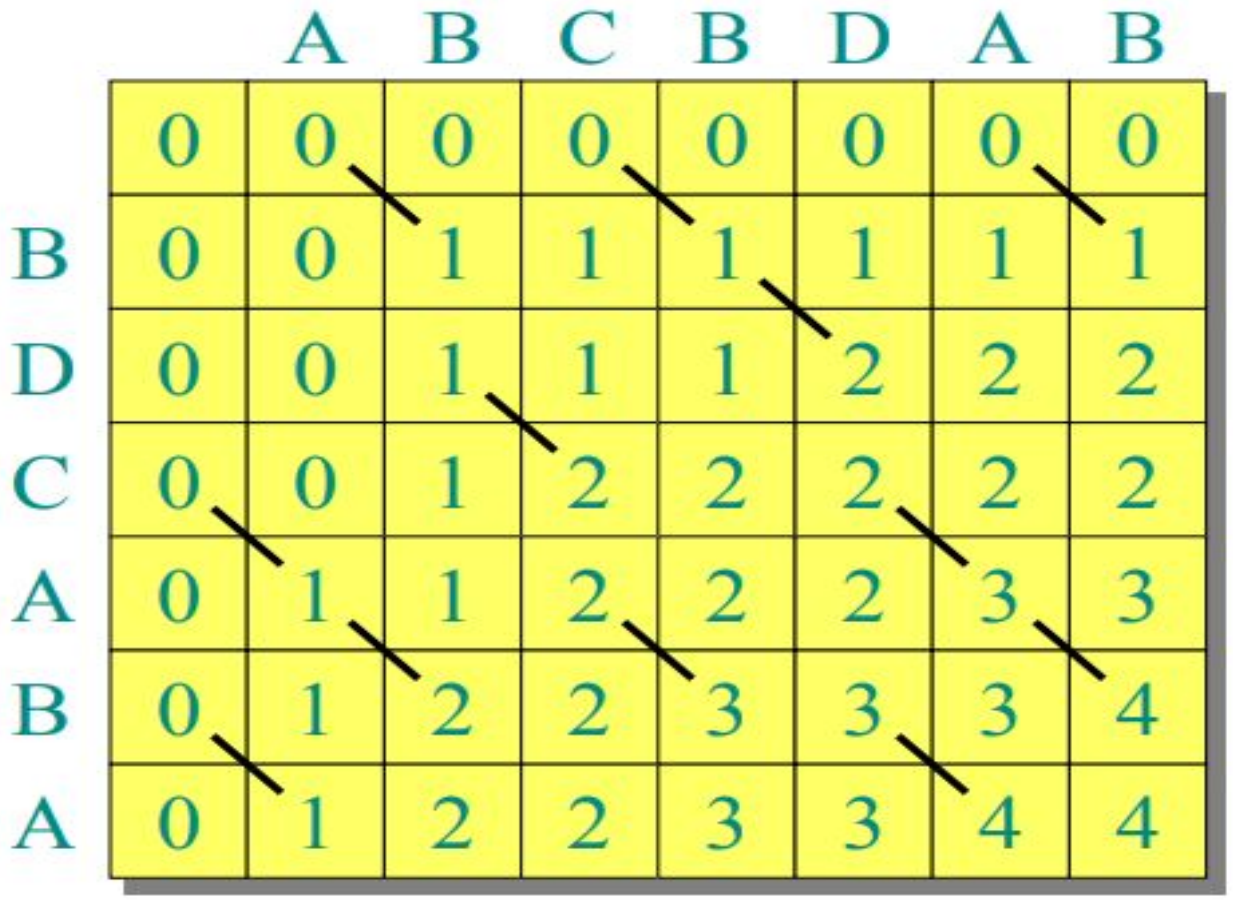
LCS-Length(X, Y) cont.

```
for i ← 1 to m do
   for j ← 1 to n do
       if x_i = y_j
         c[i, j] ← c[i-1, j-1]+1
         b[i, j] ← "D"
       else
        if c[i-1, j] ≥ c[i, j-1]
               c[i, j] ← c[i-1, j]
               b[i, j] ← "U"
          else
        c[i, j] ← c[i, j-1]
                b[i, j] ← "L"
return c and b
```

Print LCS

```
Print_LCS(X,i,j)
if i==0 or j==0
   return
if b[i,j]==‘D’
    Print_LCS(X, i-1,j-1)
    print  X[i]
else if b[I,j]== ‘U’
         Print_LCS(X, i-1,j)
           else
        Print_LCS(X, i, j-1)
```

LCS computation example



		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	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

LCS computation example