Computer Science
Spring 2021 Lecture 14:
Dynamic Programming:
Longest Common Subsequence

2 Longest Common Subsequence

Input: Two sequences (strings etc)

Output: Longest common subsequence.

Examples of common subsequences:

exercise "eerie" determine

morning triangle

toward thousand

towar_ thousan_

1: towa, thousan

2: towar, thousa

3 LCS: Recursive Solution

Let LCS(n,m) be the length of the longest common subsequence of $X[1 \ldots n]$ and $Y[1 \ldots m]$.

```
if 0==n or 0==m : return 0
if X[n] == Y[m]:
    return 1 + LCS(n-1, m-1)
else
    return max( LCS(n-1,m), LCS(n,m-1) )
```

LCS(n-1, m-1)
LCS(n-2, m)

LCS(n, m-2)
LCS(n-1, m-1)

LCS (n-1, m-1)? Overlapping subproblem

4 LCS: Iterative Solution

```
LCS(n,m):  // recursive for reference
  if n = 0 or m = 0 then
    return 0
  else if X[n] = Y[m] then
    return 1+ LCS(n − 1, m − 1)
  else
    return max( LCS(n − 1, m), LCS(n, m − 1))
```

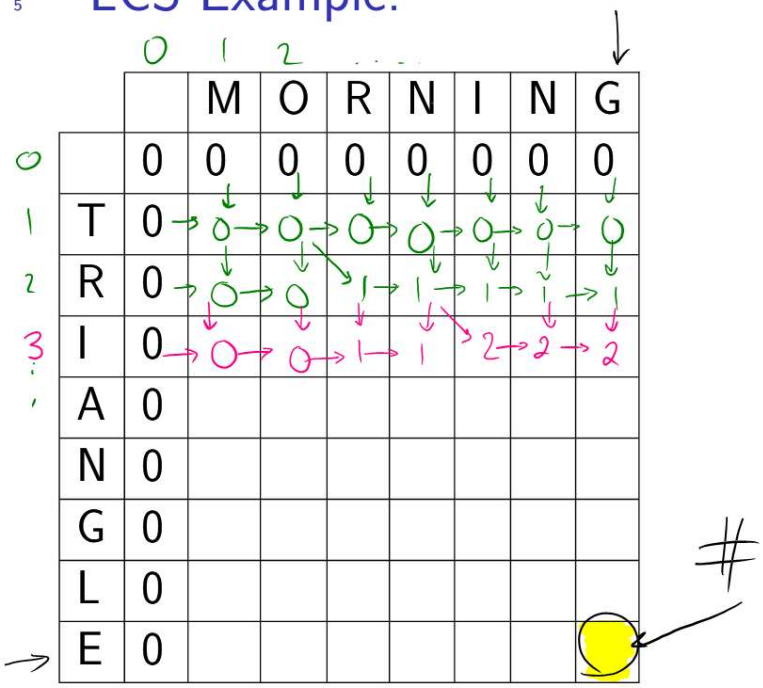
base

- smaller n OR
- equal n, smaller m

```
Declare LCS[0...n, 0...m]
for i=0...n   LCS[i,0]=0   } base
for j=0...m   LCS[0,j]=0   }
for i=1 ... n                } Θ(mn)
    for j=1 ... m   // fill in LCS[i,j]
        if X[i]==Y[j] : LCS[i,j] = 1+LCS[i-1,j-1]
        else          LCS[i,j]= max(LCS[i-1,j], LCS[i,j-1])
```

O(1)

5 LCS Example:



6 Finding the LCS itself

- We have LCS[] filled in. ← for all n, m

$i = n, j = m, S =$ empty stack
while i>0 and j>0 **do**
{ // is x[i] and y[j] part of output?
if X[i] == y[j]
push X[i] to stack. } equiv: 1 + LCS(i-1, j-1)
i-- j--
else if LCS[i,j] == LCS[i-1,j]
i--
else j--
}
While S not empty
output S.top()
S.pop()