Lecture 6 Dynamic Programming 3

Wednesday, January 29, 2020 2:55 PM

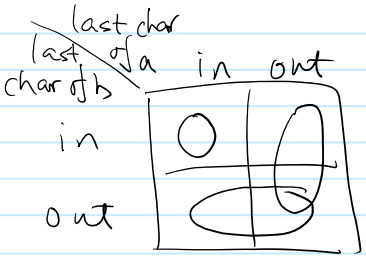
- Longest Common Subsequence (LCS)

a[] = ababcde

b[] = abbecd

for last character of a[] and b[]

want to decide whether they belong to the LCS.



attempt 1

Case 1: neither are in LCS

LCS = LCS('ababcd', 'abbec')

case 2: last char. of a[] is in LCS
last char of b[] is not.

LCS('ababcde', 'abbec')

not precise, letter 'e' in a[] may not be in LCS

attempt 2

case 1 last char of a[] is not in LCS

LCS = LCS('ababcd', 'abbecd')

case 2 last char of b[] is not in LCS

LCS = LCS('ababcde', 'abbec')

case 3 last chars of both a[] and b[] are in LCS

only valid if the last chars are equal.

a[] = 'ababcd', b[] = 'abbecd'

LCS = LCS('ababc', 'abbec') + 'd'

state: let $f[i,j]$ be the length of LCS for $a[1 \ldots i]$, $b[1 \ldots j]$

transition:

$$f[i,j] = \max \begin{cases} f[i-1,j] \\ f[i,j-1] \\ f[i-1,j-1]+1 & \text{if } a[i]=b[j] \end{cases}$$

base case: $i=0$ $f[0,j]=0$ for every j

$j=0$ $f[i,0]=0$ for every i

running time: # states $\times$ time for evaluating transition func.

nm (length of a, length of b) $\qquad O(1)$

$O(nm)$

- "voice recognition"

input: n # of sound segments

k # of phonemes

$a[1..n, 1..k]$ $a[i,j]$: score of assigning phoneme j to sound segment i

$b[1..k, 1..k]$ $b[i,j]$ score of phoneme j appear immediately after phoneme i

solution: sequence $v[1..n)$, $v[i] \in \{1, 2, \cdots, k\}$

$v[i]$: phoneme assigned to sound segment i

$$\text{score} \quad \underbrace{\sum_{i=1}^{n} a[i, v[i]]}_{\text{score from assigning phonemes to sound segments}} + \underbrace{\sum_{i=1}^{n-1} b[v[i], v[i+1]]}_{\text{score from how likely phonemes appear together.}}$$

goal: maximize the score

state: $f[i,j]$: max score for first i sound segments, sound segment i is phoneme j

transition function: if sound segment $i+1$ is phoneme p then $f[i,j] = f[i-1,p] + b[p,j] + a[i,j]$

$$f[i,j] = \max_{p=1,2,\ldots,k} \left(f[i-1,p] + b[p,j] \right) + a[i,j]$$

base case: $f[1,j] = a[1,j]$

$f[1,j] = a[1,j]$ for all j

for $i = 2$ to n

  for $j = 1$ to k

    evaluate transition function $f[i,j]$

return $\max_{j=1,\ldots,k} f[n,j]$

running time: # states $= n \times k$

time to evaluate transition function: $O(k)$

$n \times k \times O(k) = O(nk^2)$