# Lecture 6: Dynamic Programming 3

Scriber: Xiaoming Liu

Feb 9, 2020

## 1 Longest Common Subsequence(LCS)

**Problem statement**: A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. Longest common subsequence (LCS) of 2 sequences is a subsequence, with maximal length, which is common to both the sequences. Given two sequences $a$ and $b$, find the longest common subsequence.

**State Description**: Let $F[i, j]$ be length of the longest common sequence for $a[1...i]$ and $b[1...j]$.

**Analysis**: There are three possible cases.

1. Last character of *a[]* is not in LCS. e.q. LCS = LCS('ababcd', 'abbecd').
2. Last character of *b[]* is not in LCS. e.q. LCS = LCS('ababcde', 'abbec').
3. Last characters of. both *a[]* and *b[]* are in LCS. The case only happens if the last characters are equal. e.q. for a[] = 'ababcd', b[] = 'abbecd', LCS = LCS('ababc', 'abbec') + 'd'

**Transition Function**

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

**Base Case**

$$\begin{cases} f[0, j] = 0 \ \forall 0 < j <= length(b) \\ f[i, 0] = 0 \ \forall 0 < i <= length(a) \end{cases}$$

**Running Time**:

$O(n * m)$(number of possible states) * $O(1)$(time to compute each state)

## 2 Voice Recognition

**Problem statement**: Given n segments of sounds, output the phonemes. Each sound might represent one of k phonemes. You are given a list of scores for all the k phonemes for each sound segment. For every pair of phonemes, a score for how likely one comes after the other is also given.
**Input**:

1. n: number of sound segments
2. k: number of phonemes
3. a[i, j]: score of assigning phoneme j to sound segment i, in which $\forall 1 <= i <= n$, $\forall 1 <= j <= k$.
4. b[i, j]: score of phoneme j appear immediately after phoneme i. $\forall 1 <= i <= k$, $\forall 1 <= j <= k$.

**Goal**: We want to obtain sequence $v[1...n]$, in which $v[i] \in 1, 2, ..., k$. $v[i]$ is the phoneme assigned to sound segment $i$.
More specifically, we wish to obtain:

$$\underset{v}{\text{argmax}} \sum_{i=1}^{n} a[i, v[i]] + \sum_{i=1\, n-1} b[v[i], v[i+1]]$$

**State**: f[i, j] refers to the max score for the first i sound segments while sound segment i is phoneme j.

**Transition Function**:

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

**Base Case** $f[1, j] = a[1, j]$
**Algorithm**:

**Algorithm 1** Viterbi Algorithm

$f[i, j] = a[i, j]$ for all j
**for** i = 2 to n:
    **for** j = 2 to k:
        evaluate transition function f[i,j]
**return** $\max_{j=1,...,k} f[n, j]$

**Running Time**:

$O(n*k)$(number of possible states) * $O(k)$(time to compute each state)