Longest Common Subsequence

• C=c_1…c_g is a subsequence of A=a_1…a_m if C can be obtained by removing elements from A (but retaining order)
• LCS(A, B): A maximum length sequence that is a subsequence of both A and B

occuranec
occurrence
attacggct
tacgacca

Determine the LCS of the following strings

BARTHOLEMEWSIMPSON
KRUSTYTHECLOWN

String Alignment Problem

• Align sequences with gaps
  CAT TGA AT
  CAGAT AGGA

• Charge $\delta_x$ if character x is unmatched
• Charge $\gamma_{xy}$ if character x is matched to character y

Note: the problem is often expressed as a minimization problem, with $\gamma_{xx} = 0$ and $\delta_x > 0$

LCS Optimization

• A = a_1a_2…a_m
• B = b_1b_2…b_n

• Opt[j, k] is the length of LCS(a_1a_2…a_j, b_1b_2…b_k)

Optimization recurrence

If $a_j = b_k$, $\text{Opt}[j, k] = 1 + \text{Opt}[j-1, k-1]$

If $a_j \neq b_k$, $\text{Opt}[j, k] = \max(\text{Opt}[j-1,k], \text{Opt}[j,k-1])$
Give the Optimization Recurrence for the String Alignment Problem

- Charge $\delta_x$ if character $x$ is unmatched
- Charge $\gamma_{xy}$ if character $x$ is matched to character $y$

$$ \text{Opt}[j, k] = $$

Let $a_j = x$ and $b_k = y$
Express as minimization

String edit with Typo Distance

- Find closest dictionary word to typed word
- $\text{Dist}('a', 's') = 1$
- $\text{Dist}('a', 'u') = 6$
- Capture the likelihood of mistyping characters
- Different distance model for T9 on basic mobile phone

Dynamic Programming Computation

for (int $i = 0; i < n; i++$)
for (int $j = 0; j < m; j++$)
        $\text{Opt}[i, j] := 1 + \text{Opt}[i-1, j-1]$;
    else if ($\text{Opt}[i-1, j] >= \text{Opt}[i, j-1]$)
        $\text{Opt}[i, j] := \text{Opt}[i-1, j]$;
    else
        $\text{Opt}[i, j] := \text{Opt}[i, j-1]$;

Code to compute $\text{Opt}[n, m]$

for (int $i = 0; i < n; i++$)
for (int $j = 0; j < m; j++$)
        $\text{Opt}[i, j] := 1 + \text{Opt}[i-1, j-1]$, Best[$i, j$] := Diag;
    else if ($\text{Opt}[i-1, j] >= \text{Opt}[i, j-1]$)
    else

Storing the path information

Reconstructing Path from Distances
How good is this algorithm?

• Is it feasible to compute the LCS of two strings of length 300,000 on a standard desktop PC? Why or why not.

Observations about the Algorithm

• The computation can be done in $O(m+n)$ space if we only need one column of the Opt values or Best Values

• The computation requires $O(nm)$ space if we store all of the string information
Computing LCS in $O(nm)$ time and $O(n+m)$ space

- Divide and conquer algorithm
- Recomputing values used to save space

- Section 6.7 of the text, but we will not have time to cover in detail (so you are not responsible for section 6.7)

Divide and Conquer Algorithm

- Where does the best path cross the middle column?

- For a fixed $i$, and for each $j$, compute the LCS that has $a_i$ matched with $b_j$

Algorithm Analysis

- $T(m,n) = T(m/2, j) + T(m/2, n-j) + cnm$
- Solution: $T(m,n) \leq 2cnm$