CSE 421 Algorithms

Richard Anderson

Lecture 19

Longest Common Subsequence

Longest Common Subsequence

- $C=c_1\ldots c_g$ is a subsequence of $A=a_1\ldots a_m$ 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

```
ocurranec          attacggct
occurrence         tacgacca
```

Instructor Example

Determine the LCS of the following strings

BARTHOLEMEWSIMPSON

KRUSTYTHECLOWN

Student Submission

String Alignment Problem

- Align sequences with gaps

```
CAT TGA  AT
CAGAT AGGA
```

- Charge δ_x if character x is unmatched
- Charge γ_{xy} if character x is matched to character y

LCS Optimization

- $A = a_1a_2\ldots a_m$
- $B = b_1b_2\ldots b_n$

- Opt[j, k] is the length of $LCS(a_1a_2\ldots a_j, b_1b_2\ldots b_k)$

Optimization recurrence

If $a_j = b_k$, Opt[j,k] = 1 + Opt[j-1, k-1]

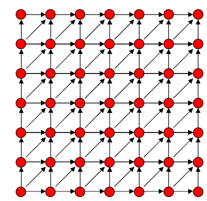
If $a_j \ne b_k$, Opt[j,k] = max(Opt[j-1,k], Opt[j,k-1])

Give the Optimization Recurrence for the String Alignment Problem

- Charge δ_x if character x is unmatched
- Charge γ_{xy} if character x is matched to character y

Student Submission

Dynamic Programming Computation

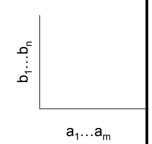


Write the code to compute Opt[j,k]

Student Submission

Storing the path information

```
A[1..m], B[1..n]
for i := 1 to m     Opt[i, 0] := 0;
for j := 1 to n     Opt[0,j] := 0;
Opt[0,0] := 0;
for i := 1 to m
        for j := 1 to n
                if A[i] = B[j]  { Opt[i,j] := 1 + Opt[i-1,j-1];  Best[i,j] := Diag; }
                else if Opt[i-1, j] >= Opt[i, j-1]
                        { Opt[i, j] := Opt[i-1, j], Best[i,j] := Left; }
                else    { Opt[i, j] := Opt[i, j-1], Best[i,j] := Down; }
```



How good is this algorithm?

- Is it feasible to compute the LCS of two strings of length 100,000 on a standard desktop PC? Why or why not.

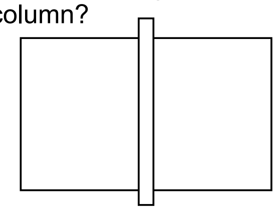
Student Submission

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 algorithm can be run from either end of the strings

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

Constrained LCS

- $LCS_{i,j}(A,B)$: The LCS such that
  - $a_1,\ldots,a_i$ paired with elements of $b_1,\ldots,b_j$
  - $a_{i+1},\ldots a_m$ paired with elements of $b_{j+1},\ldots,b_n$
- $LCS_{4,3}$(abbacbb, cbbaa)

A = RRSSRTTRTS
B=RTSRRSTST

Compute $LCS_{5,1}(A,B)$, $LCS_{5,2}(A,B)$,…,$LCS_{5,9}(A,B)$

Student Submission

A = RRSSRTTRTS
B=RTSRRSTST

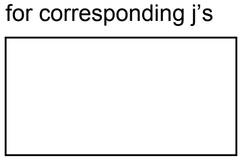
Compute $LCS_{5,1}(A,B)$, $LCS_{5,2}(A,B)$,…,$LCS_{5,9}(A,B)$

j	left	right
0	0	3
1	1	3
2	1	3
3	2	3
4	3	3
5	3	2
6	3	2
7	3	1
8	4	1
9	4	0

Instructor Example

Computing the middle column

- From the left, compute $LCS(a_1\ldots a_{m/2},b_1\ldots b_j)$
- From the right, compute $LCS(a_{m/2+1}\ldots a_m,b_{j+1}\ldots b_n)$
- Add values for corresponding j's



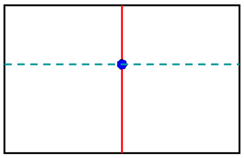
- Note – this is space efficient

Divide and Conquer

- $A = a_1,\ldots,a_m$ $\qquad$ $B = b_1,\ldots,b_n$
- Find j such that
  - $LCS(a_1\ldots a_{m/2}, b_1\ldots b_j)$ and
  - $LCS(a_{m/2+1}\ldots a_m,b_{j+1}\ldots b_n)$ yield optimal solution
- Recurse

Algorithm Analysis

- $T(m,n) = T(m/2, j) + T(m/2, n-j) + cnm$



Prove by induction that $T(m,n) \le 2cmn$

Instructor Example