# Longest Common Subsequence

Andreas Klappenecker

# Subsequences

Suppose you have a sequence $X = < x_1, x_2, \ldots, x_m$ of elements over a finite set S.

A sequence $Z = < z_1, z_2, \ldots, z_k >$ over S is called a subsequence of X if and only if it can be obtained from X by deleting elements.

Put differently, there exist indices $i_1 < i_2 < \ldots < i_k$ such that

$$z_a = x_{i_a}$$

for all a in the range $1 <= a <= k$.

# Common Subsequences

Suppose that X and Y are two sequences over a set S.

We say that Z is a common subsequence of X and Y if and only if

- Z is a subsequence of X
- Z is a subsequence of Y

# The Longest Common Subsequence Problem

Given two sequences X and Y over a set S, the longest common subsequence problem asks to find a common subsequence of X and Y that is of maximal length.

# Naïve Solution

Let X be a sequence of length m,

and Y a sequence of length n.

Check for every subsequence of X whether it is a subsequence of Y, and return the longest common subsequence found.

There are $2^m$ subsequences of X. Testing a sequences whether or not it is a subsequence of Y takes O(n) time. Thus, the naïve algorithm would take $O(n2^m)$ time.

# Dynamic Programming

Let us try to develop a dynamic programming solution to the LCS problem.

# Prefix

Let $X = < x_1, x_2, \ldots, x_m >$ be a sequence.

We denote by $X_i$ the sequence

$$X_i = < x_1, x_2, \ldots, x_i >$$

and call it the $i^{th}$ prefix of $X$.

# LCS Notation

Let X and Y be sequences.

We denote by LCS(X, Y) the set of longest common subsequences of X and Y.

# Optimal Substructure

Let $X = < x_1,x_2,\ldots,x_m>$

and $Y = < y_1,y_2,\ldots,y_n>$ be two sequences.

Let $Z = < z_1,z_2,\ldots,z_k>$ is any LCS of X and Y.

a) If $x_m = y_n$ then certainly $x_m = y_n = z_k$

and $Z_{k-1}$ is in LCS($X_{m-1}$ , $Y_{n-1}$)

# Optimal Substructure (2)

Let $X = < x_1,x_2,\ldots,x_m>$

and $Y = < y_1,y_2,\ldots,y_n>$ be two sequences.

Let $Z = < z_1,z_2,\ldots,z_k>$ is any LCS of X and Y

b) If $x_m <> y_n$ then $x_m <> z_k$ implies that Z is in LCS($X_{m-1}$ , Y)

c) If $x_m <> y_n$ then $y_n <> z_k$ implies that Z is in LCS(X, $Y_{n-1}$)

# Overlapping Subproblems

If $x_m = y_n$ then we solve the subproblem to find an element in $LCS(X_{m-1}, Y_{n-1})$ and append $x_m$

If $x_m <> y_n$ then we solve the two subproblems of finding elements in

$LCS(X_{m-1}, Y_n)$ and $LCS(X_m, Y_{n-1})$

and choose the longer one.

# Recursive Solution

Let X and Y be sequences.

Let c[i,j] be the length of an element in LCS($X_i$, $Y_j$).

$$c[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j \\ \max(c[i,j-1],c[i-1,j]) & \text{if } i,j>0 \text{ and } x_i \mathrel{<>} y_j \end{cases}$$

# Dynamic Programming Solution

To compute length of an element in LCS(X,Y) with X of length m and Y of length n, we do the following:

- Initialize first row and first column of c with 0.
- Calculate c[1,j] for 1 <= j <= n,
- c[2,j] for 1 <= j <= n ...
- Return c[m,n]
- Complexity O(mn).

# Dynamic Programming Solution (2)

How can we get an actual longest common subsequence?

Store in addition to the array c an array b pointing to the optimal subproblem chosen when computing c[i,j].

# Animation

http://wordaligned.org/articles/longest-common-subsequence

# LCS(X,Y)

```
m ← length[X]

n ← length[Y]

for i ← 1 to m do
     c[i,0] ← 0

for j ← 1 to n do
     c[0,j] ← 0
```

# LCS(X,Y)

```
for i ← 1 to m do
   for j ← 1 to n do
      if x_i = y_j
         c[i, j] ← c[i-1, j-1]+1
         b[i, j] ← "D"
      else

     if c[i-1, j] ≥ c[i, j-1]
              c[i, j] ← c[i-1, j]
              b[i, j] ← "U"
          else
      c[i, j] ← c[i, j-1]
              b[i, j] ← "L"
```

# Greedy Algorithms

There exists a greedy solution to this problem that can be advantageous when the size of the alphabet S is small.