Dynamic Programming

Longest Common Subsequences

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.

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.

ACTGAACTCTGTGCACT

TGACTCAGCACAAAAAC

ACTGAACTCTGTGCACT

TGACTCAGCACAAAAAC

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.

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})

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 $LCS(X_{m-1}, Y)$ and $LCS(X, Y_{n-1})$ and choose the longer one.

Recursive Solution

Let X and Y be sequences.

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

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

Example

	y_j	B	D	C	A
x_j	0	0	0	0	0
A	0	↑ 0	↑ 0	↑ 0	↖ 1
B	0	↖ 1	← 1	← 1	↑ 1
C	0	↑ 1	↑ 1	↖ 2	← 2
B	0	↖ 1	↑ 1	↑ 2	↑ 2

Start at b[m,n]. Follow the arrows. Each diagonal array gives one element of the LCS.

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 xi = yj
          c[i, j] ← c[i-1, j-1]+1
          b[i, j] ← "↖"

       else
           if c[i-1, j] ≥ c[i, j-1]
               c[i, j] ← c[i-1, j]
               b[i, j] ← "↑"
           else
               c[i, j] ← c[i, j-1]
               b[i, j] ← "←"
return c and b
```

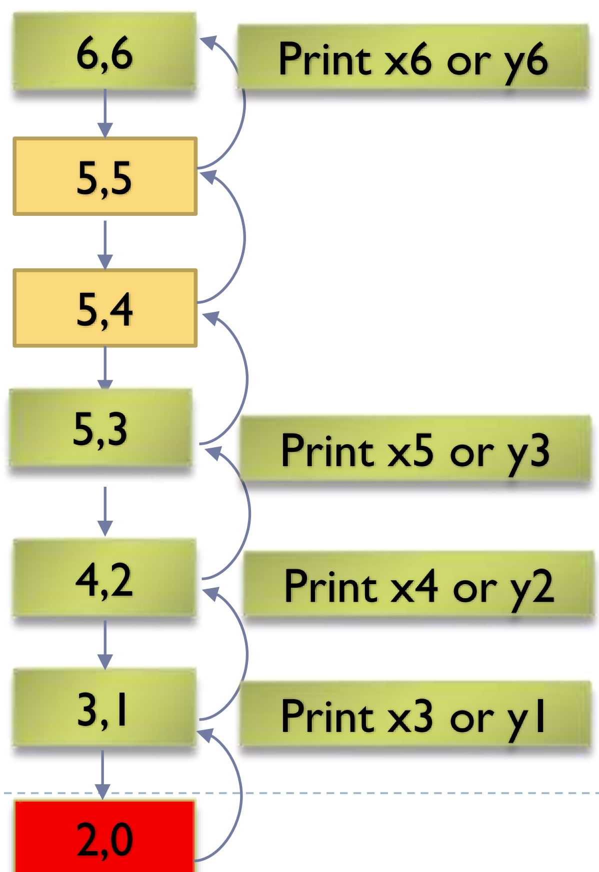
Constructing an LCS

```
PRINT-LCS(b, X, i, j)
if i = 0 or j = 0
    then return
if b[i, j] = "↖"
    then PRINT-LCS(b, X, i − 1, j − 1)
         print x_i
elseif b[i, j] = "↑"
    then PRINT-LCS(b, X, i − 1, j)
else PRINT-LCS(b, X, i, j − 1)
```

```
PRINT-LCS(b, X, i, j)
  if i = 0 or j = 0
      then return
  if b[i, j] = "↖"
      then PRINT-LCS(b, X, i − 1, j − 1)
           print x_i
  elseif b[i, j] = "↑"
      then PRINT-LCS(b, X, i − 1, j)
  else PRINT-LCS(b, X, i, j − 1)
```

Y : s e c r e t

X : b i s e c t



	Y_j	s	e	c	r	e	t
X_i		0	0	0	0	0	0
b	0	↑0	↑0	↑0	↑0	↑0	↑0
i	0	↑0	↑0	↑0	↑0	↑0	↑0
s	0	↖1	←1	←1	←1	←1	←1
e	0	↑1	↖2	←2	←2	←2	←2
c	0	↑1	↑2	↖3	←3	←3	←3
t	0	↑1	↑2	↑3	↑3	↑3	↖4
		s	e	c			t

Problem

X : s t o n e

Y : l o n g e s t

Find the Longest Common Subsequence


Thank you!