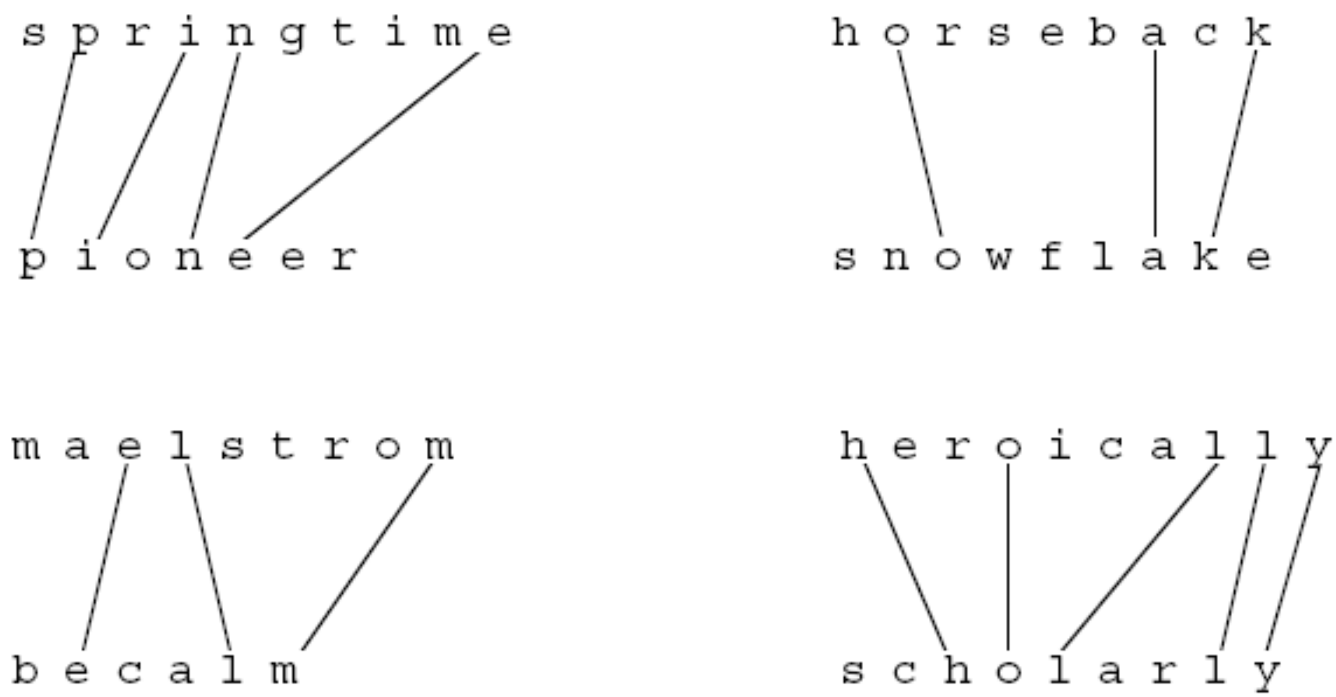
Longest common sequence

springtime	horseback
pioneer	snowflake
maelstrom	heroically
becalm	scholarly

Longest common sequence



Longest common sequence

Problem: Given 2 sequences, $X = \langle x_1, \ldots, x_m \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$. Find a subsequence common to both whose length is longest. A subsequence doesn't have to be consecutive, but it has to be in order.

Longest common sequence

Brute-force algorithm:

Longest common sequence

Brute-force algorithm:

For every subsequence of X, check whether it's a subsequence of Y.

Time complexity?

Longest common sequence

Brute-force algorithm:

For every subsequence of X, check whether it's a subsequence of Y.

Time: $\Theta(n2^m)$.

- 2^m subsequences of X to check.
- Each subsequence takes $\Theta(n)$ time to check: scan Y for first letter, from there scan for second, and so on.

Longest common sequence

Dynamic Programming ?

Longest common sequence

Optimal substructure

Notation:

$$X_i = \text{prefix } \langle x_1, \ldots, x_i \rangle$$
$$Y_i = \text{prefix } \langle y_1, \ldots, y_i \rangle$$

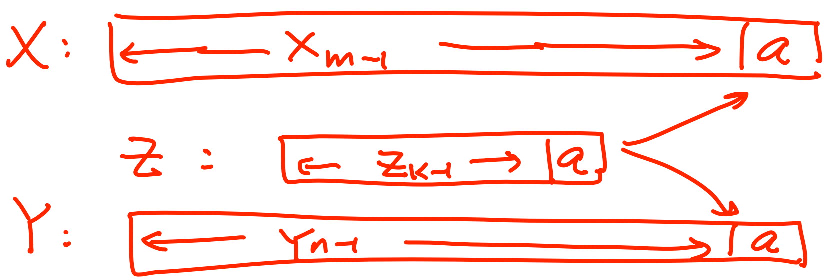
Theorem

Let $Z = \langle z_1, \ldots, z_k \rangle$ be any LCS of X and Y.

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$, then $z_k \neq x_m \Rightarrow Z$ is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$, then $z_k \neq y_n \Rightarrow Z$ is an LCS of X and Y_{n-1}.

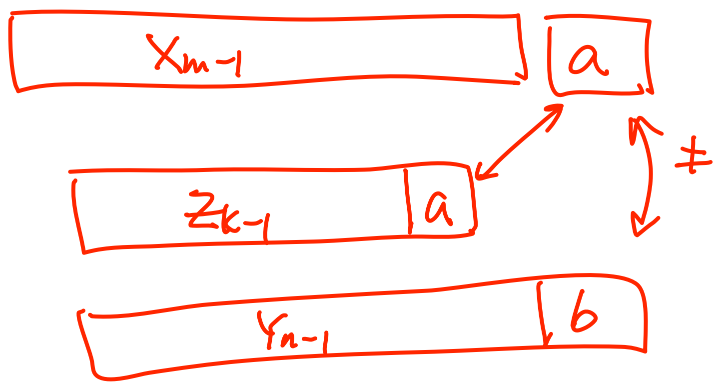
Three cases to consider :

1



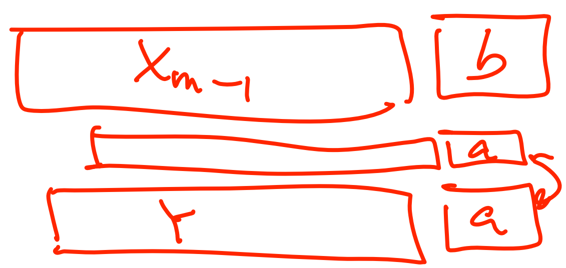
$$Z_{k-1} = LCS(X_{m-1}, Y_{n-1})$$

3



$$Z = LCS(X, Y_{n-1})$$

2



$$Z = LCS(X_{m-1}, Y)$$

Naive recursive algorithm:

LCS (X, Y): if X = empty or Y = empty : return empty

If lastSymbol(X) = lastSymbol(Y)

return LCS(prefix(X), prefix(Y)) ⊕ lastSymbol(X)

else

z_1 = LCS(prefix(X), Y)

z_2 = LCS(X, prefix(Y))

if len(z_1) > len(z_2)

return z_1

else return z_2

Longest common sequence

Recursive formulation

Define $c[i, j]$ = length of LCS of X_i and Y_j. We want $c[m, n]$.

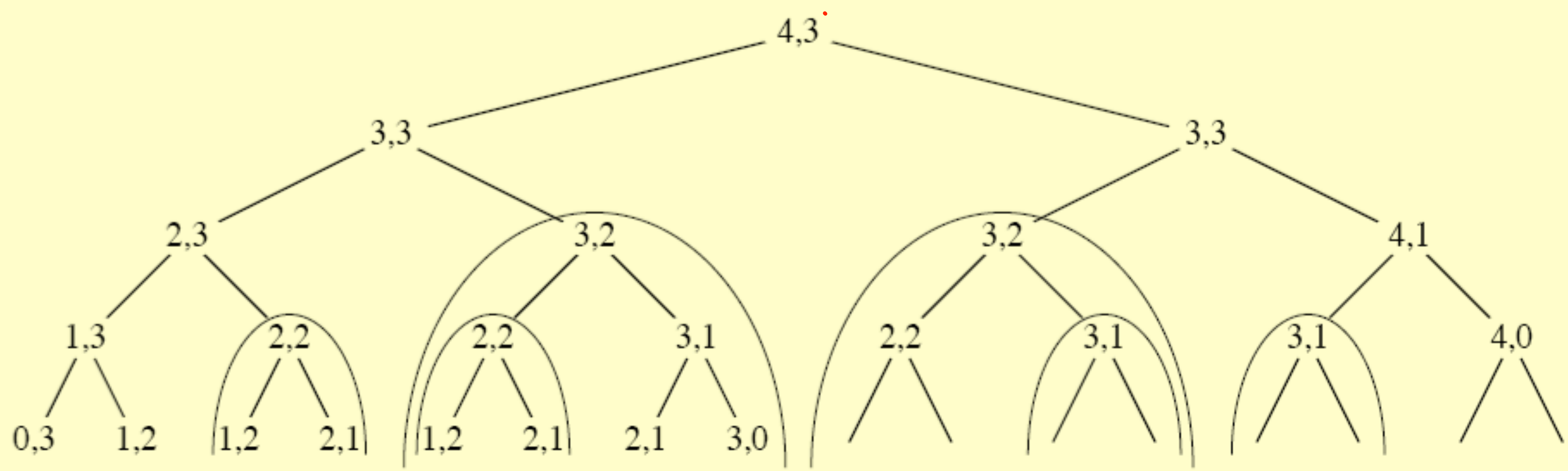
$$c[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 , \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j , \\ \max(c[i-1, j], c[i, j-1]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j . \end{cases}$$

Longest common sequence

Recursive formulation

Define $c[i, j]$ = length of LCS of X_i and Y_j. We want $c[m, n]$.

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



- Lots of repeated subproblems.
- Instead of recomputing, store in a table.

Longest common sequence

Compute length of optimal solution

```
LCS-LENGTH(X, Y, m, n)
for i ← 1 to m
    do c[i, 0] ← 0
for j ← 0 to n
    do c[0, j] ← 0
for i ← 1 to m
    do for j ← 1 to n
        do if x_i = y_j
            then c[i, j] ← c[i − 1, j − 1] + 1
                b[i, j] ← "↖"
            else if c[i − 1, j] ≥ c[i, j − 1]
                then 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
```

If $b[i,j] =$ "←":
last symbol of $LCS(X_i, Y_j)$ comes from X

If $b[i,j] =$ "↑"
last symbol of $LCS(X_i, Y_j)$ comes from Y

If $b(i,j) =$ "↖", then
last symbol of $LCS(X_i, Y_j)$ comes from both.

Longest common sequence

Compute length of optimal solution

LCS-LENGTH(X, Y, m, n)

for $i \leftarrow 1$ **to** m
  do $c[i, 0] \leftarrow 0$
for $j \leftarrow 0$ **to** n
  do $c[0, j] \leftarrow 0$
for $i \leftarrow 1$ **to** m
  do for $j \leftarrow 1$ **to** n
    do if $x_i = y_j$
      then $c[i, j] \leftarrow c[i-1, j-1] + 1$
        $b[i, j] \leftarrow$ "$\nwarrow$"
      else if $c[i-1, j] \geq c[i, j-1]$
        then $c[i, j] \leftarrow c[i-1, j]$
          $b[i, j] \leftarrow$ "$\uparrow$"
        else $c[i, j] \leftarrow c[i, j-1]$
          $b[i, j] \leftarrow$ "$\leftarrow$"
return c and b

O(?)

Longest common sequence

Compute length of optimal solution

```
LCS-LENGTH(X, Y, m, n)
for i ← 1 to m
    do c[i, 0] ← 0
for j ← 0 to n
    do c[0, j] ← 0
for i ← 1 to m
    do for j ← 1 to n
           do if x_i = y_j
                 then c[i, j] ← c[i − 1, j − 1] + 1
                      b[i, j] ← "↖"
                 else if c[i − 1, j] ≥ c[i, j − 1]
                         then 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
```

O(?)

O(n)

Longest common sequence

Compute length of optimal solution

```
LCS-LENGTH(X, Y, m, n)
for i ← 1 to m
    do c[i, 0] ← 0
for j ← 0 to n
    do c[0, j] ← 0
for i ← 1 to m
    do for j ← 1 to n
           do if x_i = y_j
                 then c[i, j] ← c[i − 1, j − 1] + 1
                      b[i, j] ← "↖"
                 else if c[i − 1, j] ≥ c[i, j − 1]
                         then 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
```

O(mn)

O(n)

Longest common sequence

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

X = ABCBDAB
Y = BDCABA

LCS(X,Y) = BCBA

Longest Common Sequence

- Naïve approach has exponential time complexity
- Dynamic programming formulation has polynomial time complexity

Dynamic Programming

- Thinking points:
  - Memoization?
  - Is memoization better than bottom-up processing? If so, why?

Summary

- Longest common sequence
  - Dynamic Programming
- Relevant material
  - MIT video lecture on dynamic programming
    - http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-046JFall-2005/VideoLectures/detail/embed15.htm
  - http://www.algorithmist.com/index.php/Longest_Common_Subsequence