Longest Common Subsequences and Substrings

Version October 28, 2016

Longest Common Subsequence

Given two sequences $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$,

Z is a *common subsequence* of X and Y of length k

if there are two strictly increasing sequence of indices i and j such that for $p = 1, 2, \ldots, k$, $x_{i_p} = y_{j_p} = z_p$.

Longest Common Subsequence

Given two sequences $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$,

Z is a *common subsequence* of X and Y of length k

if there are two strictly increasing sequence of indices i and j such that for $p = 1, 2, \ldots, k$, $x_{i_p} = y_{j_p} = z_p$.

Example:

X:	A	B		C		B	D	A	B
Y:		B	D	C	A	B		A	
Z:		B		C		B		A	

Longest Common Subsequence

Given two sequences $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$,

Z is a *common subsequence* of X and Y of length k

if there are two strictly increasing sequence of indices i and j such that for $p = 1, 2, \ldots, k$, $x_{i_p} = y_{j_p} = z_p$.

Example:

X:	A	B		C		B	D	A	B
Y:		B	D	C	A	B		A	
Z:		B		C		B		A	

Problem: Find a *longest common subsequence (lcs)* of X and Y in $O(mn)$ time

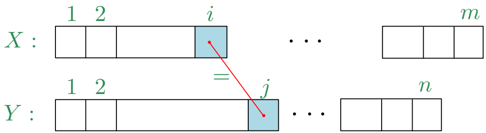
Solution: Use Dynamic Programming

Step 1: Space of Subproblems

For $1 \leq i \leq m$, and $1 \leq j \leq n$,

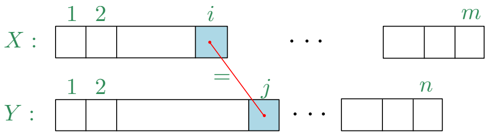
- Define $d_{i,j}$ to be the length of the longest common subsequence of $X[1..i]$ and $Y[1..j]$.
- Let D be the $m \times n$ matrix $[d_{i,j}]$.

Step 2: Recursive Formulation



Let $Z_k = (z_1, \ldots, z_k)$ be a LCS of $X[1..i]$ and $Y[1..j]$.

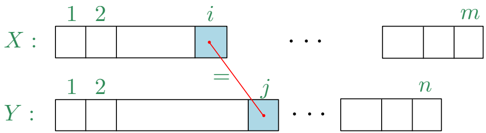
Step 2: Recursive Formulation



Let $Z_k = (z_1, \ldots, z_k)$ be a LCS of $X[1..i]$ and $Y[1..j]$.

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_k is $LCS(X[1..i-1], Y[1..j-1])$ followed by z_k

Step 2: Recursive Formulation

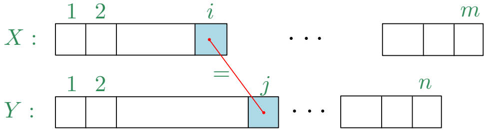


Let $Z_k = (z_1, \ldots, z_k)$ be a LCS of $X[1..i]$ and $Y[1..j]$.

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_k is $LCS(X[1..i-1], Y[1..j-1])$ followed by z_k

Case 2: If $x_i \neq y_j$, then Z_k is **either**
$LCS(X[1..i-1], Y[1..j])$ **or** $LCS(, X[1..i], Y[1..j-1])$

Step 2: Recursive Formulation



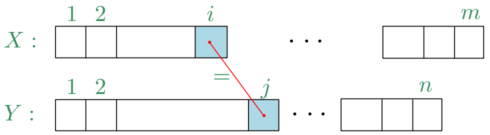
Let $Z_k = (z_1, \ldots, z_k)$ be a LCS of $X[1..i]$ and $Y[1..j]$.

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_k is $LCS(X[1..i-1], Y[1..j-1])$ followed by z_k

Case 2: If $x_i \neq y_j$, then Z_k is **either**
$LCS(X[1..i-1], Y[1..j])$ **or** $LCS(, X[1..i], Y[1..j-1])$

If $x_i \neq y_j$, the answer is the larger of the LCS's of those two cases.

Step 2: Recursive Formulation



Let $Z_k = (z_1, \ldots, z_k)$ be a LCS of $X[1..i]$ and $Y[1..j]$.

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_k is $LCS(X[1..i-1],\ Y[1..j-1])$ followed by z_k

Case 2: If $x_i \neq y_j$, then Z_k is **either**
$LCS(X[1..i-1],\ Y[1..j])$ **or** $LCS(,X[1..i], Y[1..j-1])$

If $x_i \neq y_j$, the answer is the larger of the LCS's of those two cases.

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & \text{if } x_i = y_j \\ max\{d_{i-1,j}, d_{i,j-1}\} & \text{if } x_i \neq y_j \end{cases}$$

Step 3: Bottom-up Computation by Rows

Initialize first row and column of the matrix ($d[0, j]$ and $d[i, 0]$) to 0

Solution

Step 3: Bottom-up Computation by Rows

Initialize first row and column of the matrix ($d[0,j]$ and $d[i,0]$) to 0

Calculate $d[1,j]$ for $j = 1, 2, ..., n$

Solution

Step 3: Bottom-up Computation by Rows

Initialize first row and column of the matrix ($d[0, j]$ and $d[i, 0]$) to 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$
Then, $d[2, j]$ for $j = 1, 2, ..., n$

Step 3: Bottom-up Computation by Rows

Initialize first row and column of the matrix ($d[0, j]$ and $d[i, 0]$) to 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$
Then, $d[2, j]$ for $j = 1, 2, ..., n$
Then, $d[3, j]$ for $j = 1, 2, ..., n$

Step 3: Bottom-up Computation by Rows

Initialize first row and column of the matrix ($d[0, j]$ and $d[i, 0]$) to 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$
Then, $d[2, j]$ for $j = 1, 2, ..., n$
Then, $d[3, j]$ for $j = 1, 2, ..., n$

....

We fill the table row by row, filling in each row, left to right.

$D[i,j]$	$j = 0$	1	2	3	$\cdots$	$\cdots$	n
$i = 0$	0	0	0	0	$\cdots$	$\cdots$	0
1	0						
2	0						
$\vdots$	0						
m	0						

bottom

up

We also create another $m \times n$ matrix $p[i, j]$ for $1 \leq i \leq m$, and $1 \leq j \leq n$.
This stores which of the three choices led to the maximum value creating $d[i, j]$.
This is done by pointing an arrow towards the entry that led to that choice. These arrows will permit reconstructing the elements of the LCS.

i \ j		0	1	2	3	4	5	6
		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

```
LONGEST-COMMON-SUBSEQUENCE(X, Y)
    m ← length(X);
    n ← length(Y);
    // initialization
    for i ← 0 to m
        d[i, 0] ← 0;
    for j ← 0 to n
        d[0, j] ← 0;
    // dynamic programming
    for i ← 1 to m
        for j ← 1 to n
            if (x_i = y_j)
                d[i, j] ← d[i − 1, j − 1] + 1;
                p[i, j] ← "LU";  // "LU" indicates left up arrow
            else
```

```
        if (d[i − 1, j] ≥ d[i, j − 1])
            d[i, j] ← d[i − 1, j];
            p[i, j] ← "U";  // "U" indicates up arrow
        else
            d[i, j] ← d[i, j − 1];
            p[i, j] ← "L";  // "L" indicates left
        end if
      end if
    end for
  end for

  return d, p;
```

Since it takes only $O(1)$ time to fill in each of the $O(mn)$ table entries the algorithm runs in $O(mn)$ time.

Step 4: Construction of Optimal Solution

As mentioned before, we also maintain a $m \times n$ matrix p for storing arrows to reconstruct the elements of the LCS. The following recursive procedure prints out an LCS of X and Y.

PRINT-LCS(p, X, i, j)
  if $(i = 0 \,||\, j = 0)$ **return** NULL;
  if $(p[i, j] =$ "LU"$)$
    PRINT-LCS$(p, X, i-1, j-1)$;
    print x_i;
  else
    if $(p[i, j] =$ "U"$)$
      PRINT-LCS$(p, X, i-1, j)$;
    else PRINT-LCS$(p, X, i, j-1)$;
  end if

Longest Common Substring

A slightly different problem with a similar solution

Given two strings $X = x_1 x_2 ... x_m$ and $Y = y_1 y_2 ... y_n$, find their *longest common substring* Z, i.e., a largest largest k for which there are indices i and j with $x_i x_{i+1} ... x_{i+k-1} = y_j y_{j+1} ... y_{j+k-1}$.

For example:
X : DEADBEEF
Y : EATBEEF
Z : BEEF *//pick the longest contiguous substring*

Longest Common Substring

A slightly different problem with a similar solution

Given two strings $X = x_1x_2...x_m$ and $Y = y_1y_2...y_n$, find their *longest common substring* Z, i.e., a largest largest k for which there are indices i and j with $x_ix_{i+1}...x_{i+k-1} = y_jy_{j+1}...y_{j+k-1}$.

For example:
X : DEADBEEF
Y : EATBEEF
Z : BEEF *//pick the longest contiguous substring*

Show how to do this in time $O(mn)$ by dynamic programming.

Solution

Step 1: Space of Subproblems

For $1 \le i \le m$, and $1 \le j \le n$,

- First Attempt: Define $d'_{i,j}$ to be the length of the longest common substring of $X[1..i]$ and $Y[1..j]$. (Does this work?)

Step 1: Space of Subproblems

For $1 \leq i \leq m$, and $1 \leq j \leq n$,

- First Attempt: Define $d'_{i,j}$ to be the length of the longest common substring of $X[1..i]$ and $Y[1..j]$. (Does this work?)
- Second Attempt: Define $d_{i,j}$ to be the length of the longest common substring ending at x_i and y_j. (Does this work?)

Solution

Step 1: Space of Subproblems

For $1 \le i \le m$, and $1 \le j \le n$,

- First Attempt: Define $d'_{i,j}$ to be the length of the longest common substring of $X[1..i]$ and $Y[1..j]$. (Does this work?)
- Second Attempt: Define $d_{i,j}$ to be the length of the longest common substring ending at x_i and y_j. (Does this work?)
- Let D be the $m \times n$ matrix $[d_{i,j}]$.
    - How does D provide answer?

Step 2: Recursive Formulation

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and

Z_{k-1} is a LCS of X and Y ending at x_{i-1} and y_{j-1}

Step 2: Recursive Formulation

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_{k-1} is a LCS of X and Y ending at x_{i-1} and y_{j-1}

Case 2: If $x_i \neq y_j$, then there can't be a common substring ending at x_i and y_j!

Step 2: Recursive Formulation

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_{k-1} is a LCS of X and Y ending at x_{i-1} and y_{j-1}

Case 2: If $x_i \neq y_j$, then there can't be a common substring ending at x_i and y_j!

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & \text{if } x_i = y_j \\ 0 & \text{if } x_i \neq y_j \end{cases}$$

Step 2: Recursive Formulation

Case 1: If $x_i = y_j$, then $z_k = x_i = y_j$ and
Z_{k-1} is a LCS of X and Y ending at x_{i-1} and y_{j-1}

Case 2: If $x_i \neq y_j$, then there can't be a common substring ending at x_i and y_j!

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & \text{if } x_i = y_j \\ 0 & \text{if } x_i \neq y_j \end{cases}$$

Finally, we can find length of longest common substring by finding maximum $d_{i,j}$ among all possible ending positions i and j.

$$\text{LCSubString}(X, Y) = \max\{d_{i,j}\}$$

Step 3: Bottom-up Computation

Similar to Longest Common Subsequence we set the first row and column of the matrix $d[0, j]$ and $d[i, 0]$ to be 0

Solution

Step 3: Bottom-up Computation

Similar to Longest Common Subsequence we set the first row and column of the matrix $d[0, j]$ and $d[i, 0]$ to be 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$

Solution

Step 3: Bottom-up Computation

Similar to Longest Common Subsequence we set the first row and column of the matrix $d[0, j]$ and $d[i, 0]$ to be 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$
Then, the $d[2, j]$ for $i = 1, 2, ..., 2,$

Solution

Step 3: Bottom-up Computation

Similar to Longest Common Subsequence we set the first row and column of the matrix $d[0, j]$ and $d[i, 0]$ to be 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$
Then, the $d[2, j]$ for $i = 1, 2, ..., 2,$
Then, the $d[3, j]$ for $i = 1, 2, ..., 2,$

Step 3: Bottom-up Computation

Similar to Longest Common Subsequence we set the first row and column of the matrix $d[0, j]$ and $d[i, 0]$ to be 0

Calculate $d[1, j]$ for $j = 1, 2, ..., n$
Then, the $d[2, j]$ for $i = 1, 2, ..., 2,$
Then, the $d[3, j]$ for $i = 1, 2, ..., 2,$

etc., filling the matrix row by row and left to right.

For this problem we do not need to create another $m \times n$ matrix for storing arrows. Instead, we use l_{max} and p_{max} to store the largest length of common substring and its i position respectively. This suffices to reconstruct the solution.

Solution

```
LONGEST-COMMON-SUBSTRING(X, Y)
    m ← length(X);
    n ← length(Y);
    l_max ← 0;
    p_max ← 0;
    // initialization
    for i ← 0 to m
        d[i, 0] ← 0;
    for j ← 0 to n
        d[0, j] ← 0;
    // dynamic programming
    for i ← 1 to m
        for j ← 1 to n
            if (x_i ≠ y_j)
                d[i, j] ← 0;
```

```
        else
            d[i, j] ← d[i − 1, j − 1] + 1;
            if (d[i, j] > l_max)
                l_max ← d[i, j];
                p_max ← i;
            end if
        end if
    end for
end for

return l_max, p_max;
```

The dynamic programming algorithm runs in $O(mn)$ time.

Step 4: Construction of Optimal Solution

Since we maintained l_{max} and p_{max}, we can use them to print out the longest common substring of X and Y in the following procedure.

PRINT-LCSUBSTRING(X, p_{max}, l_{max})
    if ($l_{max} = 0$) **return** NULL;
    for $i \leftarrow (p_{max} - l_{max} + 1)$ **to** p_{max}
        print x_i;