CS4311
Design and Analysis of Algorithms

Lecture 12: Dynamic Programming IV

Subsequence of a String

- Let $S = s_1 s_2 \ldots s_m$ be a string of length m
- Any string of the form

$$s_{i_1} s_{i_2} \ldots s_{i_k}$$

with $i_1 < i_2 < \ldots < i_k$ is a subsequence of S

- E.g., if S = farmers

➔ fame, arm, mrs, farmers, are some of the subsequences of S

Longest Common Subsequence

- Let S and T be two strings
- If a string is both
  - a subsequence of S and
  - a subsequence of T,

  it is a common subsequence of S and T

- In addition, if it is the longest possible one, it is a longest common subsequence

Longest Common Subsequence

- E.g.,

  S = algorithms

  T = logarithms

- Then, aim, lots, ohms, grit, are some of the common subsequences of S and T
- Longest common subsequences:

  lorithms , lgrithms

Longest Common Subsequence

- Let $S = s_1s_2\ldots s_m$ be a string of length m
- Let $T = t_1t_2\ldots t_n$ be a string of length n

Can we quickly find a longest common subsequence (LCS) of S and T?

Optimal Substructure

Let $X = x_1x_2\ldots x_k$ be an LCS of

$S_{1,i} = s_1 s_2\ldots s_i$ and $T_{1,j} = t_1t_2\ldots t_j$.

Lemma:

- If $s_i = t_j$, then $x_k = s_i = t_j$, and $x_1x_2\ldots x_{k-1}$ must be the LCS of $S_{1,i-1}$ and $T_{1,j-1}$
- If $s_i \neq t_j$, then X must either be
  - (i) an LCS of $S_{1,i}$ and $T_{1,j-1}$, or
  - (ii) an LCS of $S_{1,i-1}$ and $T_{1,j}$

Optimal Substructure

Let $len_{i,j}$ denote the length of the LCS of $S_{1,i}$ and $T_{1,j}$

→ $len_{0,j} = len_{i,0} = 0$

Lemma: For any $i, j \geq 1$,

- if $s_i = t_j$, $len_{i,j} = len_{i-1,j-1} + 1$
- if $s_i \neq t_j$, $len_{i,j} = \max \{ len_{i,j-1} , len_{i-1,j} \}$

Length of LCS

Define a function Compute_L(i,j) as follows:

Compute_L(i, j) `/* Finding len`$_{i,j}$ `*/`

1. if (i == 0 or j == 0) return 0;
2. if (s_i = t_j)

   return Compute_L(i-1,j-1) + 1;
3. else

   return max {Compute_L(i-1,j), Compute_L(i,j-1)};

Compute_L(m, n) runs in $O(2^{m+n})$ time

Overlapping Subproblems

To speed up, we can see that :

To Compute_L(i,j) and Compute_L(i-1,j+1), has a common subproblem:

Compute_L(i-1,j)

In fact, in our recursive algorithm, there are many redundant computations !

Question: Can we avoid it ?

Bottom-Up Approach

- Let us create a 2D table L to store all $len_{i,j}$ values once they are computed

BottomUp_L( ) `/* Finding min #operations */`

1. For all i and j, set L[i,0] = L[0, j] = 0;
2. for (i = 1,2,…, m)

    Compute L[i,j] for all j;

    `// Based on L[i-1,j-1], L[i-1,j], L[i,j-1]`

4. return L[m,n] ;

Running Time = $\Theta(mn)$

Remarks

- Again, a slight change in the algorithm allows us to obtain a particular LCS
- Also, we can make minor changes to the recursive algorithm and obtain a memoized version (whose running time is $O(mn)$)

Example Run: After Step 1

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0									
I	0									
R	0									
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Example Run: After Step 2, i = 1

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1	1	1	1	1	1	1	1	1
I	0									
R	0									
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Example Run: After Step 2, i = 2

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1	1	1	1	1	1	1	1	1
I	0	1	1	1	1	2	2	2	2	2
R	0									
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Example Run: After Step 2, i = 3

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1	1	1	1	1	1	1	1	1
I	0	1	1	1	1	2	2	2	2	2
R	0	1	1	2	2	2	2	2	3	3
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Example Run: After Step 2, i = 4

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1	1	1	1	1	1	1	1	1
I	0	1	1	1	1	2	2	2	2	2
R	0	1	1	2	2	2	2	2	3	3
T	0	1	1	2	2	2	3	3	3	3
Y	0									
R	0									
O	0									
O	0									
M	0									

Example Run: After Step 2

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1	1	1	1	1	1	1	1	1
I	0	1	1	1	1	2	2	2	2	2
R	0	1	1	2	2	2	2	2	3	3
T	0	1	1	2	2	2	3	3	3	3
Y	0	1	1	2	2	2	3	3	3	4
R	0	1	1	2	2	2	3	3	4	4
O	0	1	2	2	2	2	3	4	4	4
O	0	1	2	2	2	2	3	4	4	4
M	0	1	2	2	3	3	3	4	4	4

Extra information to obtain an LCS

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1↖	1←	1←	1←	1←	1←	1←	1←	1←
I	0									
R	0									
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Extra Info: After Step 2, i = 2

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1↖	1←	1←	1←	1←	1←	1←	1←	1←
I	0	1↑	1↑	1↑	1↑	2↖	2←	2←	2←	2←
R	0									
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Extra Info: After Step 2, i = 3

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1↖	1←	1←	1←	1←	1←	1←	1←	1←
I	0	1↑	1↑	1↑	1↑	2↖	2←	2←	2←	2←
R	0	1↑	1↑	2↖	2←	2↑	2↑	2↑	3↖	3←
T	0									
Y	0									
R	0									
O	0									
O	0									
M	0									

Extra Info: After Step 2

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1↖	1←	1←	1←	1←	1←	1←	1←	1←
I	0	1↑	1↑	1↑	1↑	2↖	2←	2←	2←	2←
R	0	1↑	1↑	2↖	2←	2↑	2↑	2↑	3↖	3←
T	0	1↑	1↑	2↑	2↑	2↑	3↖	3←	3←	3←
Y	0	1↑	1↑	2↑	2↑	2↑	3↑	3←	3←	4↖
R	0	1↑	1↑	2↑	2↑	2↑	3↑	3↑	4↖	4↑
O	0	1↑	2↖	2↑	2↑	2↑	3↑	4↖	4↑	4↑
O	0	1↑	2↖	2↑	2↑	2↑	3↑	4↑	4↑	4←
M	0	1↑	2↑	2↑	3↖	3←	3←	4↑	4↑	4↑

LCS obtained by tracing from L[m,n]

		D	O	R	M	I	T	O	R	Y
	0	0	0	0	0	0	0	0	0	0
D	0	1↖	1←	1←	1←	1←	1←	1←	1←	1←
I	0	1↑	1↑	1↑	1↑	2↖	2←	2←	2←	2←
R	0	1↑	1↑	2↖	2←	2↑	2↑	2↑	3↖	3←
T	0	1↑	1↑	2↑	2↑	2↑	3↖	3←	3←	3←
Y	0	1↑	1↑	2↑	2↑	2↑	3↑	3←	3←	4↖
R	0	1↑	1↑	2↑	2↑	2↑	3↑	3↑	4↖	4↑
O	0	1↑	2↖	2↑	2↑	2↑	3↑	4↖	4↑	4↑
O	0	1↑	2↖	2↑	2↑	2↑	3↑	4↑	4↑	4←
M	0	1↑	2↑	2↑	3↖	3←	3←	4↑	4↑	4↑

Reducing Space Usage

- Space usage of table L: O(mn)
- Note: To fill L, each row depends only on values of the previous row
- Suppose we only need the length of LCS

  We can do so by only keeping current row and previous row ➔ reduce space to O(n)

- Note: We can also fill L column by column

  Space usage: O(min { m,n })

Reducing Space Usage

Question: How about getting the LCS?

Can we do so with O(n) space?

Solution I:

- Use O(mn) time to find the last row
- Use O(mn) time to find the 2nd last row
- …
- Use O(mn) time to find the first row

➔ Total time: $O(m^2n)$

Solution II: Hirschberg's Trick

Let $S_{1,m/2}$ and $S_{m/2+1,m}$ denote the first half and the second half of S, respectively

Consider X, which is the LCS of S and T.

Let X' and X" denote the portion of X which comes from $S_{1,m/2}$ and $S_{m/2+1,m}$

- Here, X' or X" may be empty, and they may be of unequal length

Solution II: Hirschberg's Trick

Observation:

If X' and X'' come from $T_{1,r}$ and $T_{r+1,n}$ for some r, then

- X' is an LCS of $S_{1,m/2}$ and $T_{1,r}$
- X'' is an LCS of $S_{m/2+1,m}$ and $T_{r+1,n}$

Corollary: The reverse of X'' is LCS of the reverse of $S_{m/2+1,m}$ and reverse of $T_{r+1,n}$

Solution II: Hirschberg's Trick

Let $len_{i,j}$ = length of the LCS of $S_{1,i}$ and $T_{1,j}$

Let $rev_{i,j}$ = length of the LCS of $S_{i,m}$ and $T_{j,n}$
= length of the LCS of reverse of $S_{i,m}$ and reverse of $T_{j,n}$

Lemma: $len_{m,n} = \max_r \{ len_{m/2,r} + rev_{m/2+1,r+1} \}$

And, if $r = r^*$ achieves the above max,

- X' is an LCS of $S_{1,m/2}$ and T_{1,r^*}
- X'' is an LCS of $S_{m/2+1,m}$ and $T_{r^*+1,n}$

Solution II: Hirschberg's Trick

Based on the previous lemma, we can find r* as follows:

Step 1: Fill L for row 1 to row m/2
(from top-left corner)

Step 2: Fill L for row m to row m/2 + 1
(from bottom-right corner)

Step 3: Find r* from rows m/2 and m/2+1

Example Run: Step 1

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T											
Y											
R											
O											
O											
M											

Example Run: Step 1

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T											
Y											
R											
O											
O											
M											

Example Run: Step 1

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T											
Y											
R											
O											
O											
M											

Example Run: Step 1

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T											
Y											
R											
O											
O											
M											

Example Run: Step 1

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y											
R											
O											
O											
M											

Example Run: Step 2

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y											
R											
O											
O											
M											

Example Run: Step 2

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y											
R											
O											
O											
M											

Example Run: Step 2

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y											
R											
O											
O											
M											

Example Run: Step 2

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y											
R											
O											
O											
M											

Example Run: Step 2

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y		2	2	2	1	1	1	1	1	1	0
R											
O											
O											
M											

Example Run: Step 3 (Find r*)

		D	O	R	M	I	T	O	R	Y	
D											
I											
R											
T	0	1	1	2	2	2	3	3	3	3	
Y		2	2	2	1	1	1	1	1	1	0
R											
O											
O											
M											

Solution II: Hirschberg's Trick

- After finding r^*, we can recursively find
  (i) LCS of $S_{1,m/2}$ and T_{1,r^*}
  (ii) LCS of $S_{m/2+1,m}$ and $T_{r^*+1,n}$

- Total Space: $O(n)$ because space can be reused!
- Total Time:

  $T(m,n) = T(m/2,r^*) + T(m/2, n-r^*) + \Theta(mn)$

➔ By recursion-tree, $T(m,n) = \Theta(mn)$