Dynamic Programming Strings

Monday, 5 July 2021 8:05 PM

1Q) Given 2 strings, Calculate Longest Common Subsequence? (length)

- Any order elements
- ordered based on index value

Ex1:

	0	1	2	3	4	5	6
S_1:	a	b	b	c	d	g	f
S_2:	b	a	c	d	e	g	f

Common subsequences:

- a c d g f
- b c d g f

len = 5

Ex2: N S_1:

	0	1	2	3	4	5	6
	h	l	a	g	h	i	p

M S_2:

	0	1	2	3	4	5
	l	g	i	g	h	m

l g i

len = 3

Dp state:

$Dp(S_1, S_2)$ = Calculate LCS of string (S_1 & S_2)

Build expression Can-I

S_1: a c d f (0 1 2 3)

S_2: g f l e

$= 1 + Dp(S_1[0-2], S_2[0-2])$

Can-II

S_1: a c d f (0 1 2 3)

S_2: e c f

$= \max\{Dp(S_1[0-2], S_2[0-2]),\ Dp(S_1[0-3], S_2[0-1])\}$

Dp state

$Dp(i, j)$: length of longest subsequence in $S_1[0-i]$ & $S_2[0-j]$

Dp expression

$Dp(i,j) = \begin{cases} \text{if}(i==-1 \| j==-1)\ \{\text{return } 0\} \\ \text{if}(S_1[i] == S_2[j])\ \{1 + dp(i-1, j-1)\} \\ \text{else}\ \{\max\{dp(i-1, j), dp(i, j-1)\}\} \end{cases}$

overlapping: ✓

Doubt?

S_1 = a c d f e e (0 1 2 3 4 5 6)

S_2 = g c l e e (0 1 2 3 4)

$Dp[6,4] = \{1 + Dp[5,3]\}$

$1 + Dp[4,2]$

$S_1[0-4]$: a c d f e

$S_2[0-2]$: g c l

$Dp(i, j)$ = Max subseque from $S_1[i, N-1]$, $S_2[j, M-1]$

$\begin{cases} \text{if}(S[i] == S[j]) \\ \quad Dp(i,j) = 1 + Dp(i+1, j+1) \\ \text{else} \\ \quad Dp(i,j) = \max\{Dp(i+1, j), Dp(i, j+1)\} \end{cases}$

Final ans: $Dp[0][0]$

Dp table:

$\{dp[N][M] = -1?\}$ → Global/local: pass to function

Top down App

Pass by reference / (Pass by value) → TC, SC } Both will increase

int LCS (string S_1, string S_2, int i, int j){

{ if(i == -1 || j == -1) return 0; } Base condition are covered in single line

if(dp[i][j] == -1)

This function is called for very first time

if($S_1[i] == S_2[j]$){

$dp[i][j] = 1 + LCS(S_1, S_2, i-1, j-1)$

}

else{

$dp[i][j] = \max\{LCS(S_1, S_2, i, j-1), LCS(S_1, S_2, i-1, j)\}$

}

return dp[i][j];

}

No need to worry about order of working loop

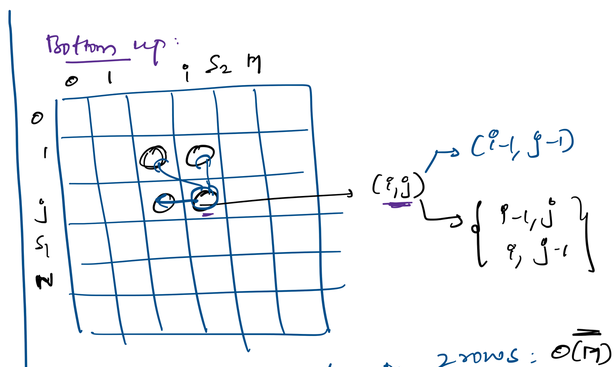
// Dp state: $(N*M)$

// TC for each dp: $O(1)$

LCS (S_1, S_2, $S_1.size()-1$, $S_2.size()-1$)

SC: $O(N*M)$

Bottom up:



In Bottom you can do in 2 rows: $O(M)$

TC: $O(N*M)$

2Q) Edit Distance

Given $S_1 \to S_2$, find min steps to make S_1 a S_2, $S_1 == S_2$

In a single operation: Replace, Insert, Delete } In S_1

S_1: d f a c l

S_2: f g l

- Delete d
- Add c
- Repla a → g

S_1: f c h

S_2: a e k l

- Replace f → a
- Replace h → f
- Add l at back

Solutions:

$\boxed{\max(S_1.size(), S_2.size()) - LCS(S_1, S_2)}$ Suggested

[Cases verify & check]

(Dp state)

$dp[i][j]$ = (Minimum number of states required to convert $S_1[0-i]$ to $S_2[0-j]$)

(Dp expression).

Base Conditions

$\begin{cases} \text{if}(i==-1 \ \&\& \ j==-1)\ \{S_1 \text{ converted } S_2 \text{ return } 0\} \\ \text{if}(i==-1)\ \{\text{// In } S_2 \text{ How many more characters } j+1 \text{ return } j+1, \text{ [Insert operation]}\} \\ \text{if}(j==-1)\ \{\text{// In } S_1 \text{ How many character } i+1 \text{ return } i+1, \text{ [Delete operation]}\} \\ \text{if}(S[i] == S[j])\ \{dp[i-1][j-1]\} \\ \text{else}\ \{dp[i][j] = \min\{ \end{cases}$

j is index

Case1: Replace last $S_1[i]$ with $S_2[j]$ & continue

$1 + dp[i-1][j-1]$

Case2: Delete $S_1[i]$ & continue

$1 + dp[i-1][j]$

Case3: Insert $S_2[j]$ after $S_1[i]$ & continue

$1 + dp[i][j-1]$ }

S_1: a b c d g (0 1 2 3 4)

S_2: d c g (0 1 2)

Can1: Replace d → g

$Dp(S_1[0,2], S_2[0,1])$

Can2: Delete d

$Dp(S_1[0,2], S_2[0,2])$

Can3: Insert g after d

$Dp(S_1[0,3], S_2[0,1])$

Can 1

S_1: a b c

S_2: d c g

$1 + Dp(S_1[0,2], S_2[0,1])$

Can 2

S_1: a b c

S_2: d c g

$1 + Dp(S_1[0,2], S_2[0,2])$

Can 3

S_1: a b c d g

S_2: d c g

$1 + Dp(S_1[0,3], S_2[0,2])$

Dp state: $N*M$

TC: $O(N*M)$

SC: $O(N*M)$

[Can be optimized to 2 rows]

3Q) Wildcard Pattern Matching

Note: In your strings S_1 & S_2, there can be ?, *

S_1, S_2 } Check if they are same or not?

2Q) Distinct Subsequences: How many subsequence of A has string B?

To string S_1

A: R a b b b f

B: R a b b f

- R a b b f f
- R a b b f f
- R a b b f f

$Dp(i, j)$ = { How many subsequence in $S_1[0-i]$, = $S_2[0-j]$ }

Dp expression

$dp[i,j] = \begin{cases} \text{if}(i==-1 \ \&\& \ j==-1)\ \text{return } 1; \\ \text{if}(i==-1)\ \{\text{// We are not able to get } S_2, \text{ return } 0;\} \\ \text{if}(j==-1)\ \{\text{// We are able to get } S_2, \text{ return } 1\} \\ \text{if}(S_1[i] \neq S_2[j])\ \{ \text{ } i^{th} \text{ character } S_1 \text{ cannot be ans} \\ \quad dp[i-1, j] \\ \text{else}\ \{ \\ \quad dp[i-1, j-1] + dp[i-1, j] \end{cases}$

S_1: a b c (0 1 2)

S_2: a b c (0 1 2)

S_1: [0 2] + S_1[0 2]

S_2 [0 1] + S_2 [0 2]

return dp[i][j]

Dp Table: $N*M$

TC: $N*M * O(1)$

SC: $N*M$

Can be further optimized 2 rows