COMP3600-DP-Question Set

UP Academy

September 2018

Question 1 (Longest Palindrome Sub-Sequence)

Propose an algorithm to find the longest palindrome sub-sequence when given a character array.(The palindrome is 'civic', 'racecar', the palindrome sub-sequence for 'character' here is 'carac' with length 5).
Answer
Step 1: The optimal Structure and Overlapped sub-problem
We denote two pointer i, j here, where i starts from 1 to the n, j starts from n to 1. The input array is denoted as like $A[i]$ or $A[j]$. Also, we let Z_k denotes the return result.
Firstly, if we found that $A[i] == A[j]$ and $i \leq j$, then the optimal state here will be $Z[k] = A[i] = A[j]$ and $Z[k-2]$ is the solution for $A[i+1], A[j-1]$ (Since we need to add both of A[i] and A[j] into the solution).
Then, if $A[i] \neq A[j]$ and $i \leq j$ implies $Z[k]$ is either a solution of $A[i+1], A[j]$ or $A[i], A[j-1]$.
Additionally, if two pointers meet together (i == j), then ? (Think by yourself)

Step 2: Construct the state table (function)

$$s[i,j] = \begin{cases} 0, \text{if } i > j \\ 1, \text{if } i == j \\ s[i+1, j-1] + 2, \text{if } A[i] == A[j] \text{ and } i < j \\ \max(s[i+1, j], s[i, j-1]) \text{if } A[i] != A[j] \text{ and } i < j \end{cases} \tag{1}$$

Step 3, Compute the length by algorithm
This algorithm might be little bit tricky, since it does not relate to the state table very well (You cannot start indexation for both i and j from 1 to the array's length). If you do it so, you will NOT get a correct answer. (why?) There is one way can resolve this issue : We use the length of sub-sequence as the index. For example, given a input string such as "character", We first initialize all the diagonal of the state table to set the value 1. Then start the length from 2, the algorithm will check all combinations of "c,h", "h,a","a,r", ... "t,e","e,r" to see their optimal solution by the state table. After then it works from 3 again, See https://www.geeksforgeeks.org/longest-palindromic-subsequence-dp-12/ for the detailed solution.
Actually, You can use LCS as a in-route to solve this problem. The solution is omitted here. The running time for both of these two methods are $O(n^2)$.

shift length = 1

	c	h	a	r	a	c	t	e	r
c	1	1							
h		1							
a			1						
r				1					
a					1				
c						1			
t							1		
e								1	
r									1

shift length = 2

	c	h	a	r	a	c	t	e	r
c	1	1							
h		1	1						
a			1	1					
r				1	1				
a					1	1			
c						1	1		
t							1	1	
e								1	1
r									1

shift length = 3

	c	h	a	r	a	c	t	e	r
c	1	1	1						
h		1	1	1					
a			1	1	3				
r				1	1	1			
a					1	1	1		
c						1	1	1	
t							1	1	1
e								1	1
r									1

shift length = 4

	c	h	a	r	a	c	t	e	r
c	1	1	1	1					
h		1	1	1	3				
a			1	1	3	3			
r				1	1	1	1		
a					1	1	1	1	
c						1	1	1	1
t							1	1	1
e								1	1
r									1

shift length = 5

	c	h	a	r	a	c	t	e	r
c	1	1	1	1	3				
h		1	1	1	3	3			
a			1	1	3	3	3		
r				1	1	1	1	1	
a					1	1	1	1	1
c						1	1	1	1
t							1	1	1
e								1	1
r									1

shift length = 6

	c	h	a	r	a	c	t	e	r
c	1	1	1	1	3	5			
h		1	1	1	3	3	3		
a			1	1	3	3	3	3	
r				1	1	1	1	1	3
a					1	1	1	1	1
c						1	1	1	1
t							1	1	1
e								1	1
r									1

shift length = 7

	c	h	a	r	a	c	t	e	r
c	1	1	1	1	3	5	5		
h		1	1	1	3	3	3	3	
a			1	1	3	3	3	3	3
r				1	1	1	1	1	3
a					1	1	1	1	1
c						1	1	1	1
t							1	1	1
e								1	1
r									1

shift length = 8

	c	h	a	r	a	c	t	e	r
c	1	1	1	1	3	5	5	5	
h		1	1	1	3	3	3	3	3
a			1	1	3	3	3	3	3
r				1	1	1	1	1	3
a					1	1	1	1	1
c						1	1	1	1
t							1	1	1
e								1	1
r									1

shift length = 9

	c	h	a	r	a	c	t	e	r
c	1	1	1	1	3	5	5	5	5
h		1	1	1	3	3	3	3	3
a			1	1	3	3	3	3	3
r				1	1	1	1	1	3
a					1	1	1	1	1
c						1	1	1	1
t							1	1	1
e								1	1
r									1

Step 4, Compute the solution

As a guide, if we use the state table given above , we should have the following results:

	c	h	a	r	a	c	t	e	r
c	1	1	1	1	D	LD	L	L	L
h		1	1	1	D	L/D	L/D	L/D	L/D
a			1	1	LD	L	L	L	L/D
r				1	1	1	1	1	LD
a					1	1	1	1	1
c						1	1	1	1
t							1	1	1
e								1	1
r									1

We start finding out our solution by the maximum "LD", which is C = 5. Then follow the directed arrow until the length = 1, we have our solution $C \to A \to R$ while R is the ending point, to obtain the full answer, we just need to backtrack from R to the C.

Question 2* Transitive Closure of a Graph

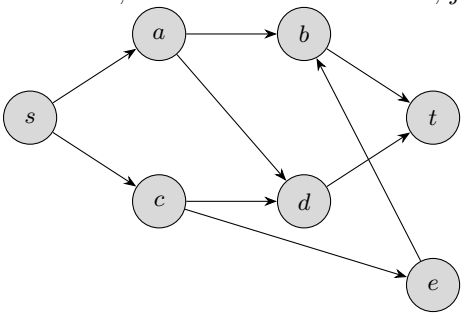
Note: Before reading this question, you must have a really good background in MATH2301, especially in Relations Part (No Graph Theory knowledge required here), otherwise you can just skip this question.

0.1 Transitive Closure Definition

Consider a directed graph $G = (V, E)$, $\forall i, j \in V$, if there exists a simple path from i to j, then these two edge should be connected directly as well. The transitive closure is to connect all of these unconnected edges, but they have a simple path edge.

0.2 Graph Representation by using adjacent Matrix

We use binary number $\{0, 1\}$ to represent if two vertices are connected or not. If these two vertices are connected, then we mark the ith row, jth column as 1, otherwise it's 0. E.g.,



with its adjacent matrix,

$$\begin{bmatrix} \text{s}\to s & \text{s}\to a & \text{s}\to b & \text{s}\to c & \text{s}\to d & \text{s}\to e & \text{s}\to t \\ \text{a}\to s & \text{a}\to a & \text{a}\to b & \text{a}\to c & \text{a}\to d & \text{a}\to e & \text{a}\to t \\ \cdots & & & & & & \\ \cdots & & & & & & \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The transitive closure for this graph is:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

0.3 Solved by using Dynamic Programming

We can apply the dynamic programming to compute the transitive closure of a graph. Since the graph can be represented as Matrix with binary number. One well-known algorithm here is *Floyd-Warshall Algorithm*, it solves the problem in $O(|V|^3)$, where $|V|$ means the carnality of vertices set. The algorithm says the TC has the following optimal structure:
Before the computing, we first need to add reflexive for all of the vertices since there always such a simple path from a to a. (This is the initialization)

$$T^0[i][j] = 1 \text{ if } i == j \vee (i,j) \in E \tag{2}$$

For other cases, we have (The k here denotes the kth vertex or "dealing with kth row" for easy understanding.)

$$T^k[i][j] = T^{k-1}[i][j] \vee (T^{k-1}[i][k] \wedge T^{k-1}[k][j]) \tag{3}$$