DALHOUSIE UNIVERSITY

CSCI 6057 - ADVANCED DATA STRUCTURES

WINTER 2022

ASSIGNMENT 4 - SOLUTION

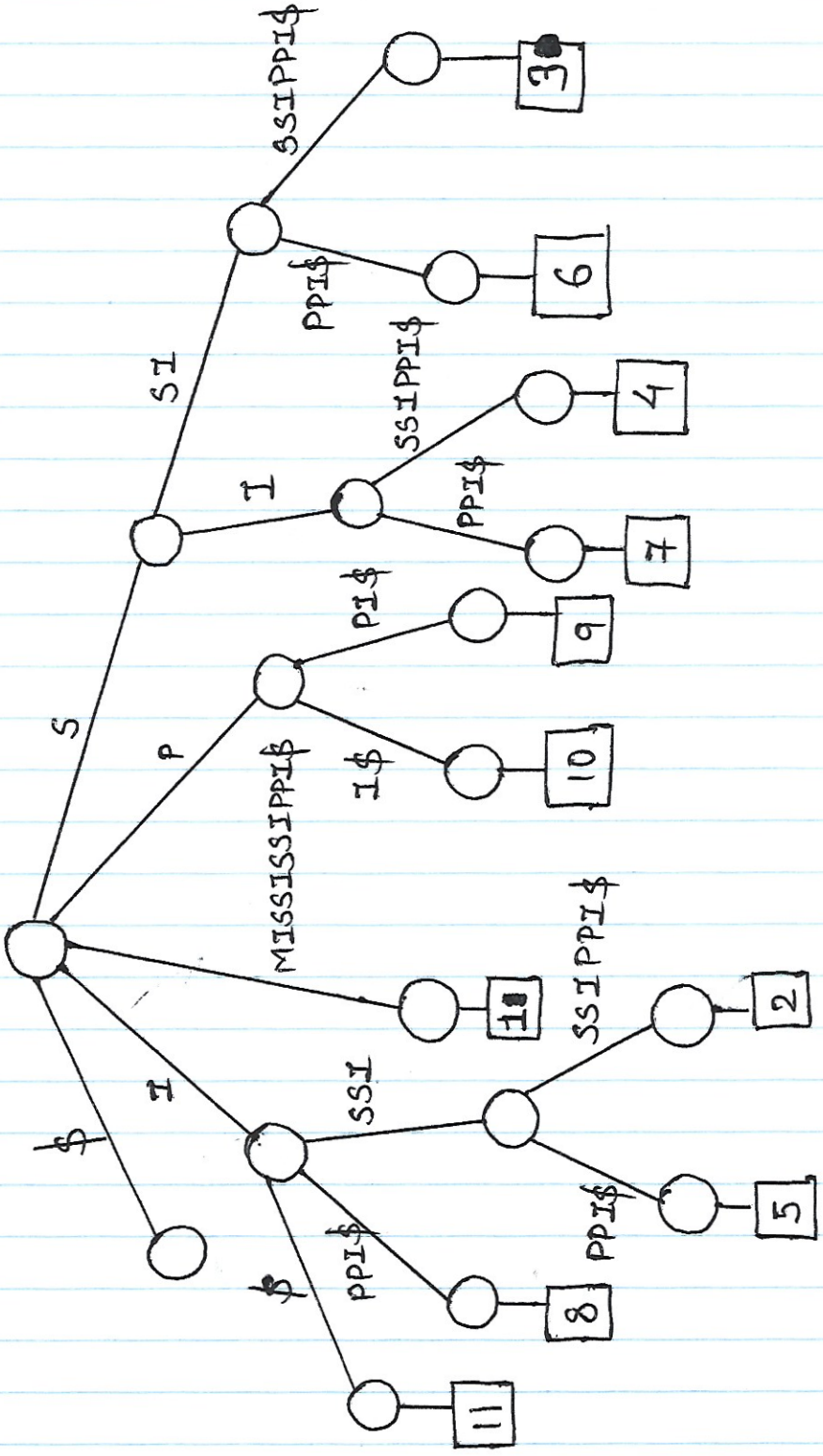
Banner: B00872269

SUBMITTED ON: MARCH 24, 2022

① SOLUTION:

Let string S =

M	I	S	S	I	S	S	I	P	P	I
1	2	3	4	5	6	7	8	9	10	11



(2)

(2) SOLUTION:

Given:

The recurrence relation:

$$T(n) = O[\lg \min\{i, n-i+1\}] + T(i-1) + T(n-i)$$

and with assumption $T(0) \leq c$ for some constant c.

Prove $T(n) = O(n)$ using induction

Part 1:

Base case:

$T(1) = 0$ for $n = 0$
$T(0) \leq c$, c = constant

Part 2:

Inductive hypothesis:

Let's assume for an arbitary value n, T(n) is O(n). Further, let's propose a solution to prove the induction steps.
If we are able to prove the proposal, it will be our real solution.

Part 3:

Induction steps:

$O[\lg \min\{i, n-i+1\}] \leq c \cdot \min\{i, n-i+1\}$ will be used such that

$$T(n) = O[\lg \min\{i, n-i+1\}] + T(i-1) + T(n-i) \quad -(1)$$

⇓ becomes / transforms

$$T(n) \leq c \cdot \min(i, n-i+1) + T(i-1) + T(n-i) \quad -(2)$$

inequality

Let's assume ~~(equation)~~ (2) has a solution with

$$T(n) \leq c \cdot n \text{ for } n > K$$
$$T(n) \leq n \quad -(3)$$

The initial equation is:

$$T(n) \leq c \cdot \min(i, n-i+1) + T(i-1) + T(n-i) \quad -(4)$$
$$\text{for } 1 \leq i \leq n$$

$$T(n) \leq c \cdot (i+1) + T(i) + T(n-i) \quad -(5)$$
$$\text{for } 0 \leq i \leq n/2$$

(4)

Part 4:

Now for $n+1$, the equation will be:

$$T(n+1) \leq c_0(i+1) + T(i) + T(n+1-i)$$

$$T(n+1) \leq c_0(i+1) + T(i) + T(n-i+1)$$

$$T(n+1) \leq c_0(i+1) + T(i) + T(n-i) + 1$$

$$T(n+1) \leq T(n) + 1 \quad [\text{Substituting } (5)]$$

$$T(n+1) \leq n+1 \quad -(6)$$

∴ Hence, $T(n) = O(n)$ since our hypothesis of $T(n) = O(n)$ was proved with (6).

(5)

(3) SOLUTION:

Given:

A is a string of length m
B is a string of length n
} over a constant-size alphabet

Prove that to find longest common substring (LCS) of the string A and B, a suffix tree can be constructed in O(m+n) time.

The above hypothesis can be proved using UKKonen's algorithm to construct a generalized suffix tree for strings A and B with delimeters # and $ for A and B respectively.

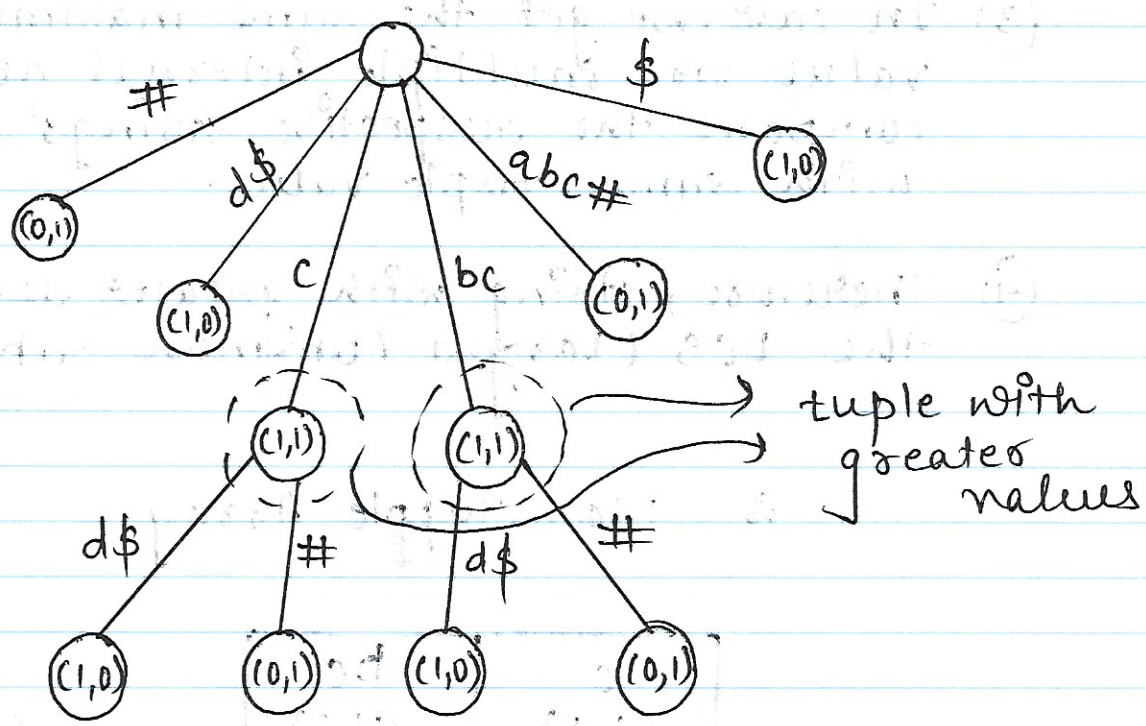
Let A = abc and B = bcd

After adding delimiters:

A = abc# B = bcd$

The generalized suffix tree looks something like this:

(next page)



Now, we will perform a Depth First Traversal (DFS) on the suffix tree and count the number of occurrences of # and $ as a tuple ($, #). And finally choose the LCS by comparing values of tuple ($,#)

A rough algorithm can be given as:

(1) While counting the occurrences of $ and # at each node, maintain an array of max count.

(2) If at any internal node, the value of the tuple ($, #) is greater than the current value, update the entry in the array.

(3) In case, we get the same maximum tuple value for multiple internal nodes, compare the respective strings' length with same tuple value.

(4) Therefore, string with greater length is the LCS (Largest Common Substring).

∴ Max Tuple Array

c	bc
(1,1)	(1,1)

len(c) < len(bc)

∴ LCS ⇒ bc

(8)

(4) SOLUTION:

(i) Table E has one entry for each possible bit vector of length 4 and store the number of 1s in this bit vector.

Let's construct the table E.

	Bit vector	(no. of 1s in bit vector)
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	1
3	0 0 1 1	2
4	0 1 0 0	1
5	0 1 0 1	2
6	0 1 1 0	2
7	0 1 1 1	3
8	1 0 0 0	1
9	1 0 0 1	2
10	1 0 1 0	2
11	1 0 1 1	3
12	1 1 0 0	2
13	1 1 0 1	3
14	1 1 1 0	3
15	1 1 1 1	4

(9)

Now, for table f, length of bit vectors of length $\frac{\lg n}{2}$ is:

$$O(\sqrt{n} \times \lg n \times \lg\lg n) \quad -(1)$$

($\sqrt{n}$: row; $\lg n$: column; $\lg\lg n$: length of superblock/chunk)

$\left(2^{\frac{\lg n}{2}} = \sqrt{n}\right)$

Now, since table E has only one column, $\frac{\lg n}{2} \simeq \lg n = 1$ in equation (1)

$\therefore$ E occupies $O(\sqrt{n} \times \lg\lg n)$

(ii)

Let's take $x = 0110$

$n = 4$ $i = 2$ (no. of 1's)

Now to perform bitwise right-shift (>>),

The no. of bits shifted $= n - i - 1$
$= 4 - 2 - 1$
$= 1$

$\therefore$ 0110 >> 0011

Now, from table E, the no. of ones in

0011 is '2'. Thus, the result is 2.

Now, in modern computers, the above operation i.e. right-shift consumes O(1) time.

Now Rank =

rank of superblock +
relative rank of block within superblock +
relative rank of element using table E

= O(1)

Hence, using table E, the rank can be calculated in O(1) i.e., constant time.