STAMFORD UNIVERSITY BANGLADESH

Name : **Abdullah Al Monayem**

ID : **CSE 071 08128**

Subject : Algorithms (CSI - 231)

Batch : 71 - A

Email : **almonayem2019@gmail.com**

Contact No. : **01747-534818**

Ans. the. Q. No: 1

(a)

The LCS of "RAHMANI

and "JAHANGIR' is,

		J	A	H	A	N	G	I	R
	0	0	0	0	0	0	0	0	0
R	0	↑0	↑0	↑0	↑0	↑0	↑0	↑0	↖1
A	0	↑0	↖1	←1	↖1	←1	←1	←1	↑1
H	0	↑0	↑1	↖2	←2	←2	←2	←2	←2
M	0	↑0	↑1	↑2	↑2	↑2	↑2	↑2	↑2
A	0	↑0	↖1	↑2	↖3	←3	←3	←3	←3
N	0	↑0	↑1	↑2	↑3	↖4	←4	←4	←4
I	0	↑0	↑1	↑2	↑3	↑4	↑4	↖5	←5

we can see that the LCS is →

"AHANI"

Here,

We have two strings,

Let, $X = \{\overset{x_1}{R}, \overset{x_2}{A}, \overset{x_2}{H}, \overset{x_4}{M}, \overset{x_5}{A}, \overset{x_6}{N}, \overset{x_7}{I}\}$

and $Y = \{\underset{y_1}{J}, \underset{y_2}{A}, \underset{y_3}{H}, \underset{y_4}{A}, \underset{y_5}{N}, \underset{y_6}{G}, \underset{y_7}{I}, \overset{y_8}{R}\}$

First compare 'I' and 'R'. If they matched, find the subsequence in the remaining string and then append the 'I' with it

If "I" or, $x_7 \neq y_8$ ("R")

(i) Remove x_7 from X and find LCS from x_1 to x_{7-1} and y_1 to y_8

(ii) Remove y_8 from Y and find LCS from x_1 to x_7 and y_1 to y_{8-1}

In each step, we reduce the size of the problem into the subproblems. It is optimal substructure.

It is a dynamic programming approach because it has optimal substructure.

Subject :

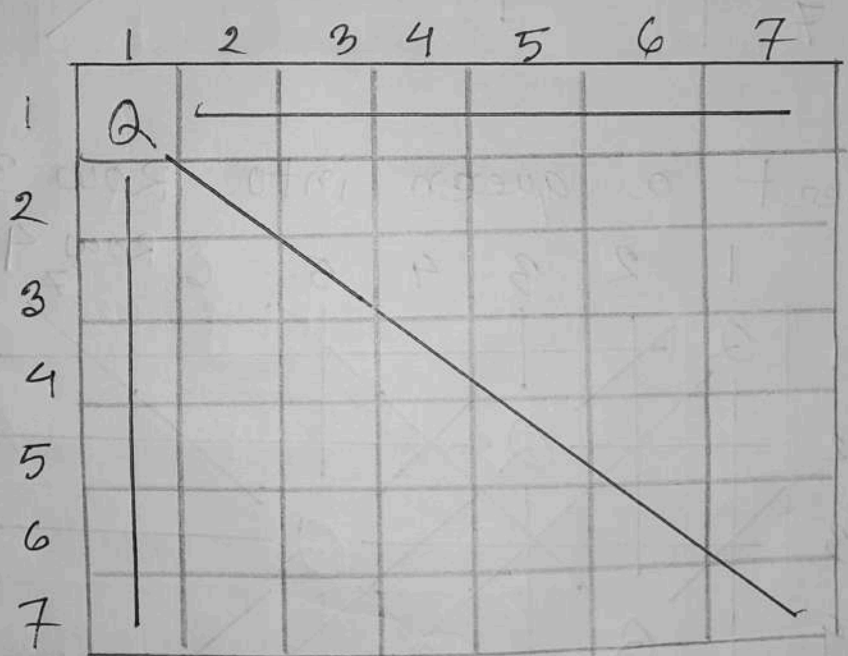
Date :

Ans. the. Q. NO: 1

(b)

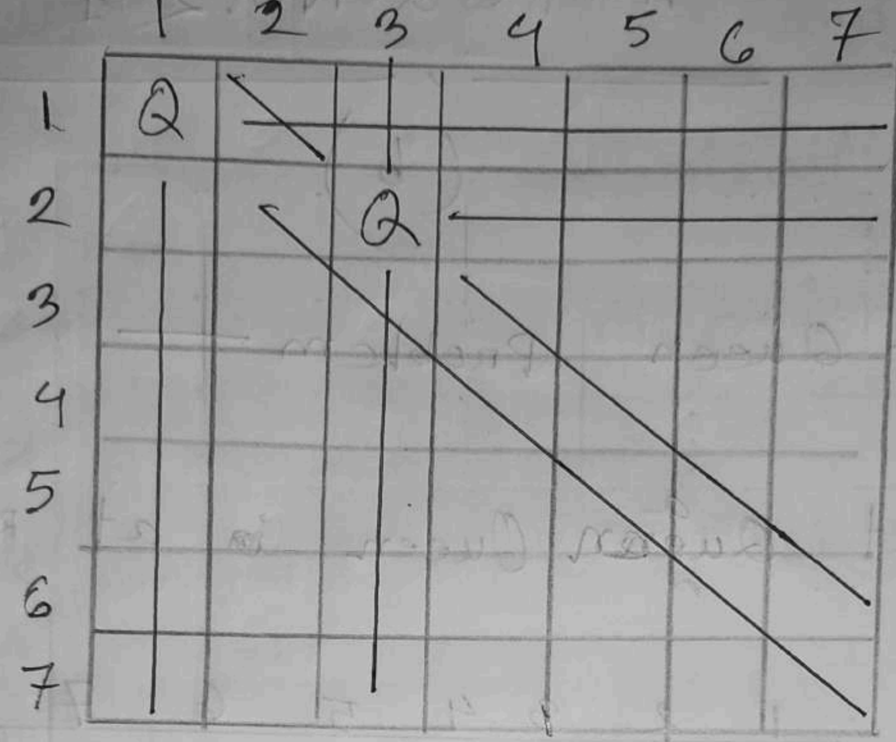
7 - Queen Problem —

Insert a Queen at ROW-1.

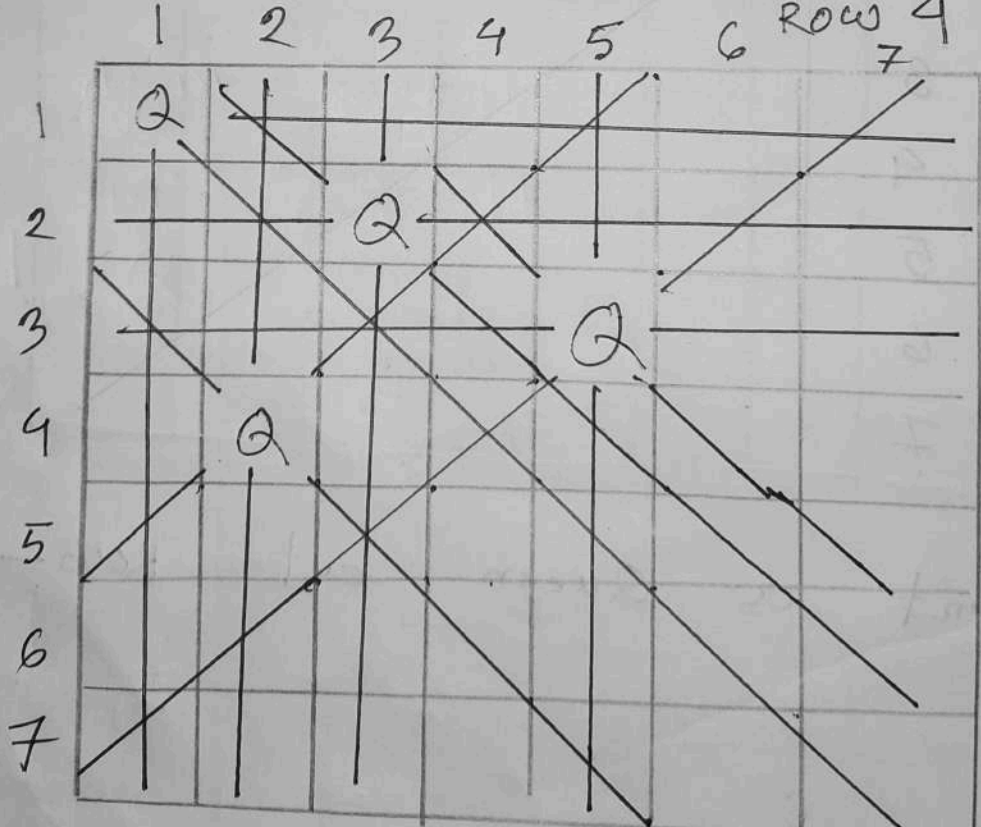


Insert a Queen at ROW-2

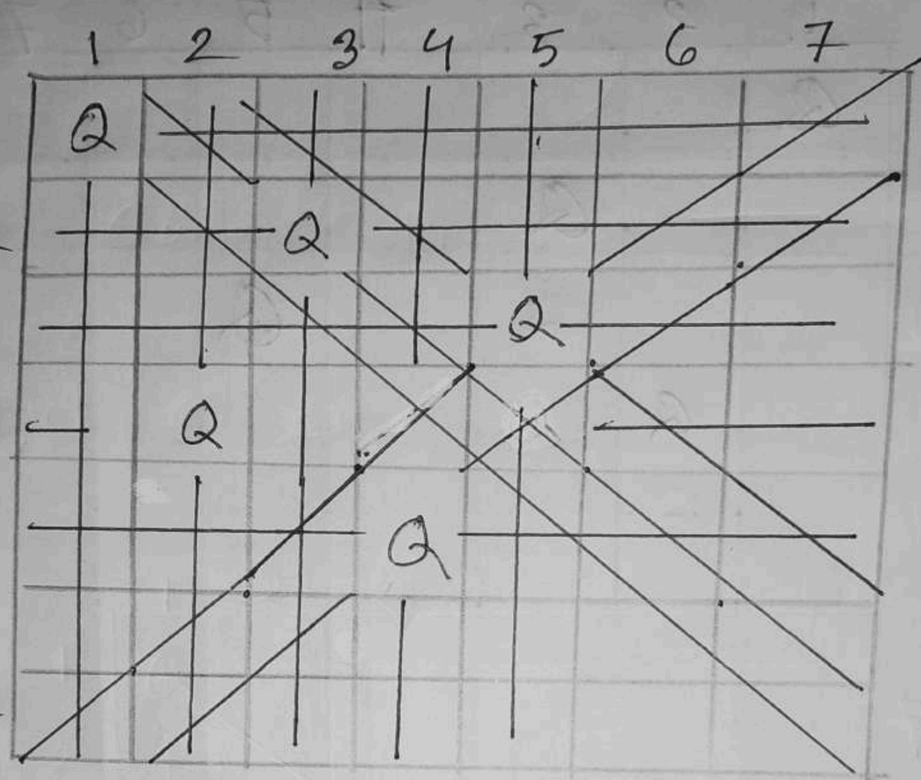
P.T.O



Insert a queen into Row 3 and Row 4



Insert a queen into Row 5



when we insert a queen in Row - 5, we see we can't insert a queen into row - 6.

So backtracking

P.T.O

Now,

	1	2	3	4	5	6	7
1	Q						
2			Q				
3					Q		
4		Q					
5							
6							
7							

Again backtracking,

	1	2	3	4	5	6	7
1	Q						
2			Q				
3					Q		
4							Q
5							
6							
7							

So, we insert a queen at Row-5

	1	2	3	4	5	6	7
1	Q						
2			Q				
3					Q		
4							Q
5		Q					
6							
7							

Again, insert a queen into Row-6

	1	2	3	4	5	6	7
1	Q						
2			Q				
3					Q		
4							Q
5		Q					
6				Q			
7							

P.T.O

And Finally,

	1	2	3	4	5	6	7
1	Q						
2			Q				
3					Q		
4							Q
5		Q					
6				Q			
7						Q	

So, the Output : (1, 3, 5, 7, 2, 4, 6)

(Ans.)

Ans. the. Q. NO : 1

(C)

Step – 1

O_i	P	Q	R	S	T	U	V	W	X
P_i	5	46	15	10	20	10	55	12.6	32
W_i	1	10	2	2	3.6	1.2	5.6	7	4
$u_i = \frac{P_i}{\omega_i}$	5	4.6	7.5	5	5.6	8.3	9.8	1.8	8

Step - 2

O_i	V	U	X	R	T	P	S	Q	W		
P_i	55	10	32	15	20	5	10	46	12.6		
w_i	5.6	1.2	4	2	3.6	1	2	10	2		
$u_i = \frac{P_i}{w_i}$	9.8	8.3	8	7.5	5.6	5	5	4.6	1.8		
$x_i(1)$	0	0	0	0	0	0	0	0	0		
$x_i(2)$	1	0	0	0	0	0	0	0	0		
$x_i(3)$	1	1	0	0	0	0	0	0	0		
$x_i(4)$	1	1	1	0	0	0	0	0	0		
$x_i(5)$	1	1	1	1	0	0	0	0	0		
$x_i(6)$	1	1	1	1	1	0	0	0	0		
$x_i(7)$	1	1	1	1	1	1	0	0	0		
$x_i(8)$	1	1	1	1	1	1	1	0	0		
$x_i(9)$	1	1	1	1	1	1	1	0.66	0		
$x_i(10)$	1	1	1	1	1	1	1	0.66	0		

Total capacity,
$m = 26$

Rest capacity,

$\to U = 26 - 0 = 26$

$\to U = 26 - 5.6 = 20.4$

$\to U = 20.4 - 1.2 = 19.2$

$\to U = 19.2 - 4 = 15.2$

$\to U = 15.2 - 2 = 13.2$

$\to U = 13.2 - 3.6 = 9.6$

$\to U = 9.6 - 1 = 8.6$

$\to U = 8.6 - 2 = 6.6$

$\to U = 6.6 - \left(10 \times \frac{6.6}{10}\right)$
$= 0$

Maximum Profit = $\sum p_i x_i$

$= (55 \times 1) + (10 \times 1) + (32 \times 1) + (15 \times 1)$

$+ (20 \times 1) + (5 \times 1) + (10 \times 1) + (4.6 \times 6.6)$

$= 177.4$

Total weight = $\sum w_i x_i$

$= (5.6 \times 1) + (1.2 \times 1) + (4 \times 1) +$

$(2 \times 1) + (3.6 \times 1) + (1 \times 1) +$

$(2 \times 1) + (10 \times 0.66)$

$= 26$

Fraction taken of the items:

$(x_P, x_Q, x_R, x_S, x_T, x_U, x_V, x_W, X_X)$

$= (1, 0.66, 1, 1, 1, 1, 1, 1, 0, 1)$

(Ans.)

If we're not allowed to take fraction of an object, then it turns into dynamic problem rather than greedy problem. It is called "0-1" Knapsack.

Then, we need to find an optimal substructure to find the global optimal solution, rathen than a greedy technique to solve the problem.

Ans. the. Q. No: 1

(d)

Backtracking is usually a recursive approach to enumerate all possible solutions. Whenever a dead-end is encountered, the algorithm backs up and systematically tries to find a different solution.

There are greedy algorithms that are guaranteed to find ~~a different solution. There are~~ the globally optimal result, but there are also greedy algorithms which will only find suboptimal results.

P.T.O

Dynamic Programming and Greedy mostly solves optimizations problem, better.

That's why some problems like, N-Queen problem, is much more efficient to solve with backtracking other than Greedy or DP.

Am. the. Q. NO : 2

(a)

Given,

$Q = 25,000$

$R = 68000$

$S = 3000$

$T = 13000$

$U = 1000$

$V = 76000$

$W = 15000,$

$X = 5000$

$Y = 18000$

$Z = 26000$

Now, Let, $U = 1$, $S = 3$, $X = 5$,

$T = 13$, $W = 15$, $Y = 18$, $Q = 25$, $Z = 26$,

$R = 68$. and $V = 76$

Subject :

Date :

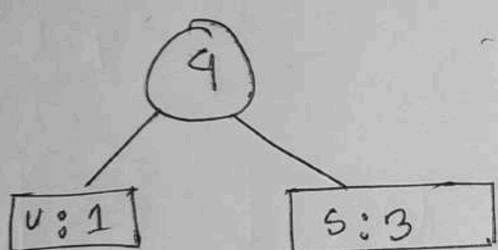
Step -1 :

U : 1 | S : 3 | X : 5 | T : 13 | W : 15

Y : 18 | Q : 25 | Z : 26 | R : 68

V : 76

Step-2 :

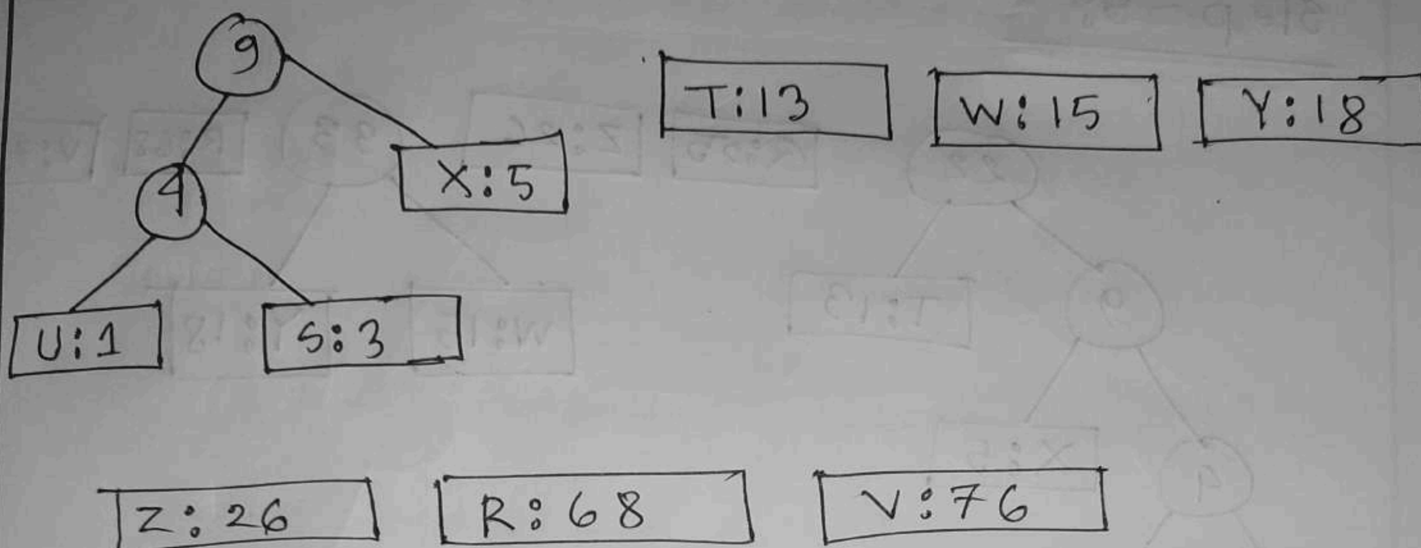


X : 5 | T : 13 | W : 15

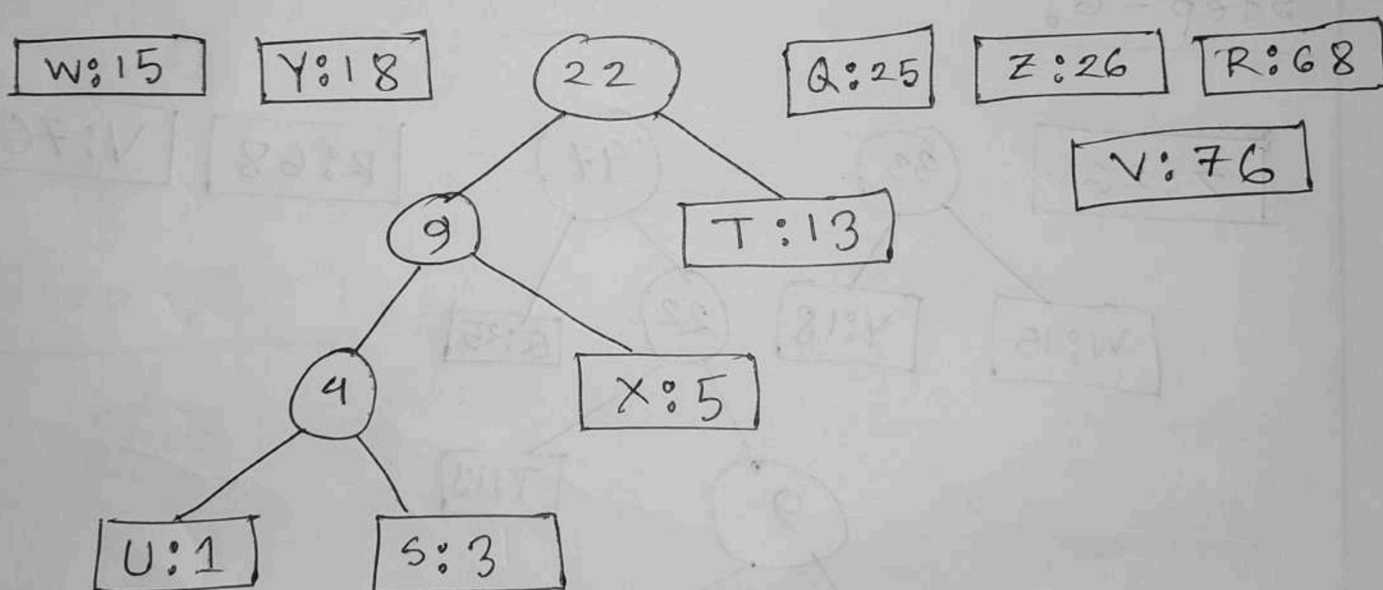
Y : 18 | Q : 25 | Z : 26 | R : 68

V : 76

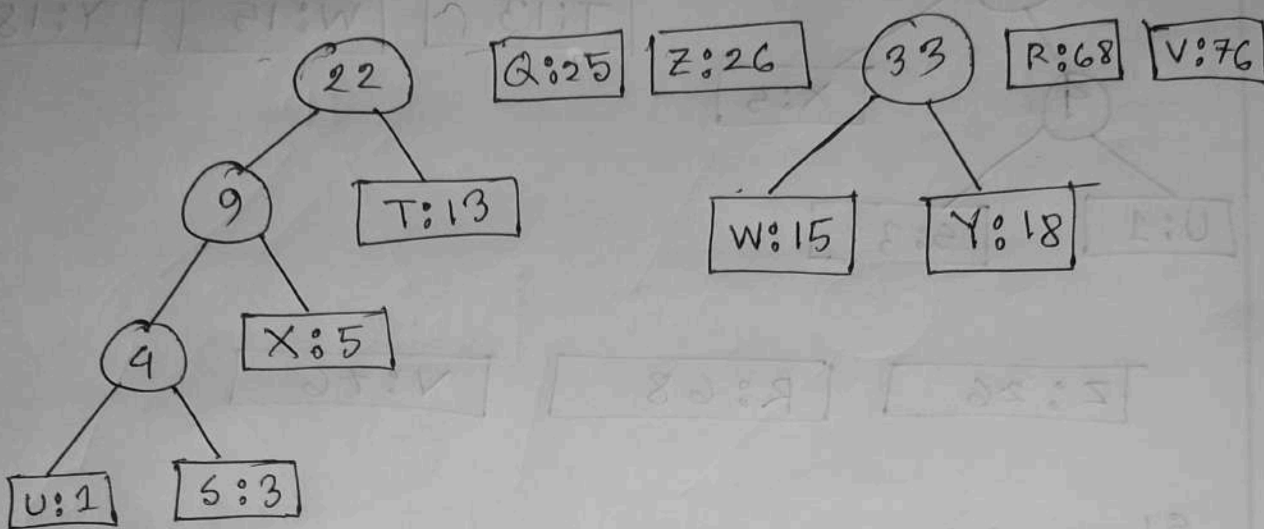
Step-3 :


9
4
X:5
U:1
S:3
T:13
W:15
Y:18
Z:26
R:68
V:76

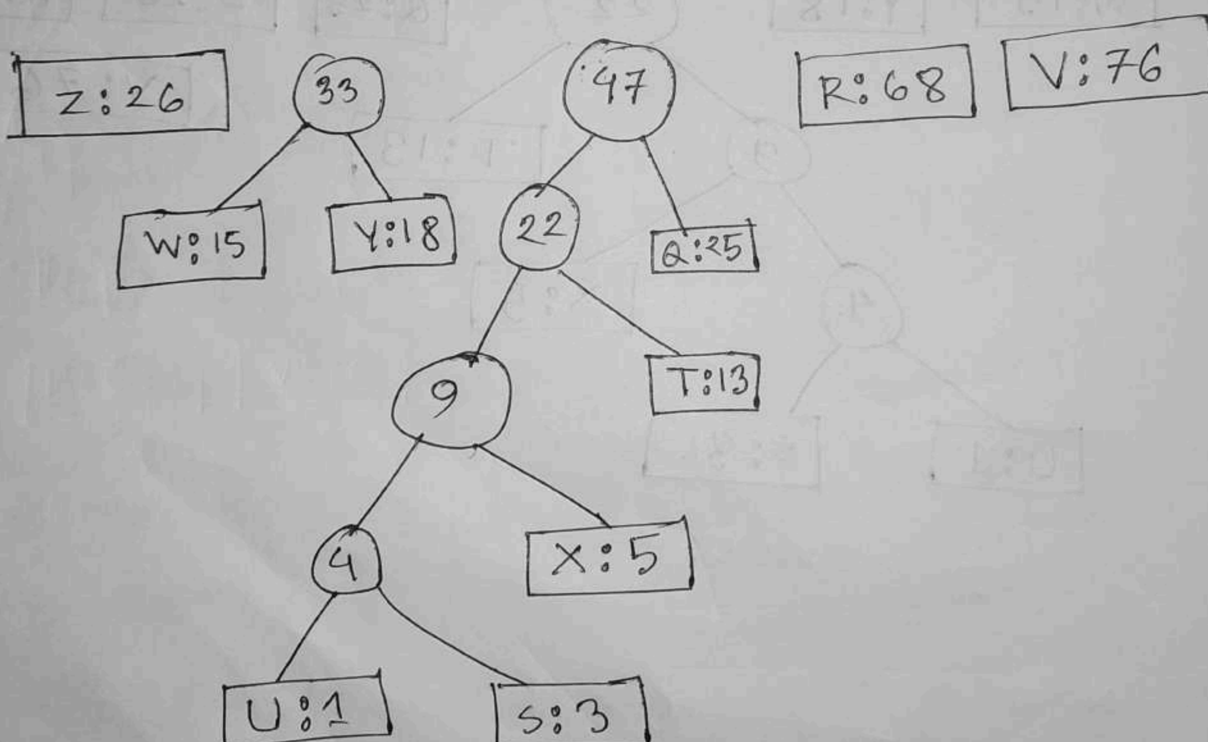
Step-4:


W:15
Y:18
22
Q:25
Z:26
R:68
V:76
9
T:13
4
X:5
U:1
S:3

Step - 5:



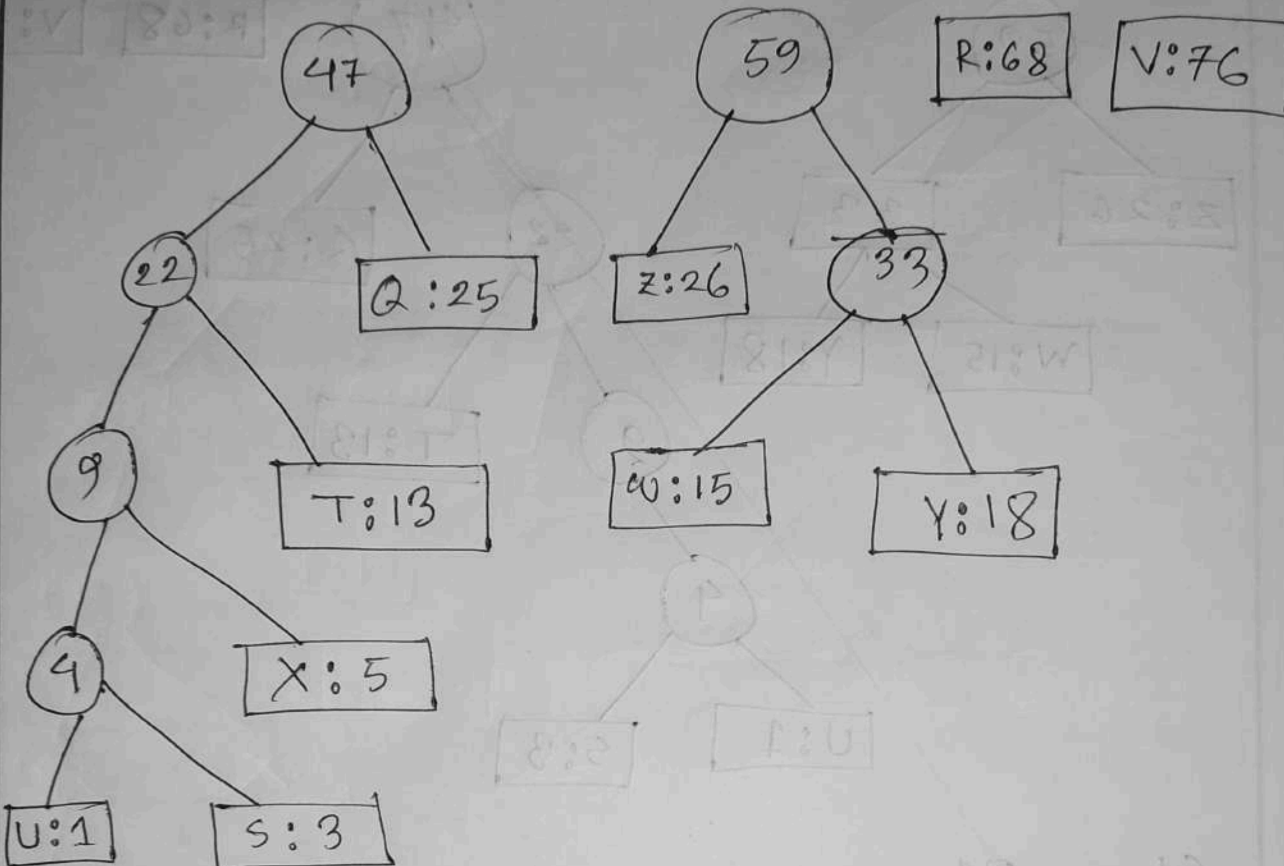
Step - 6:



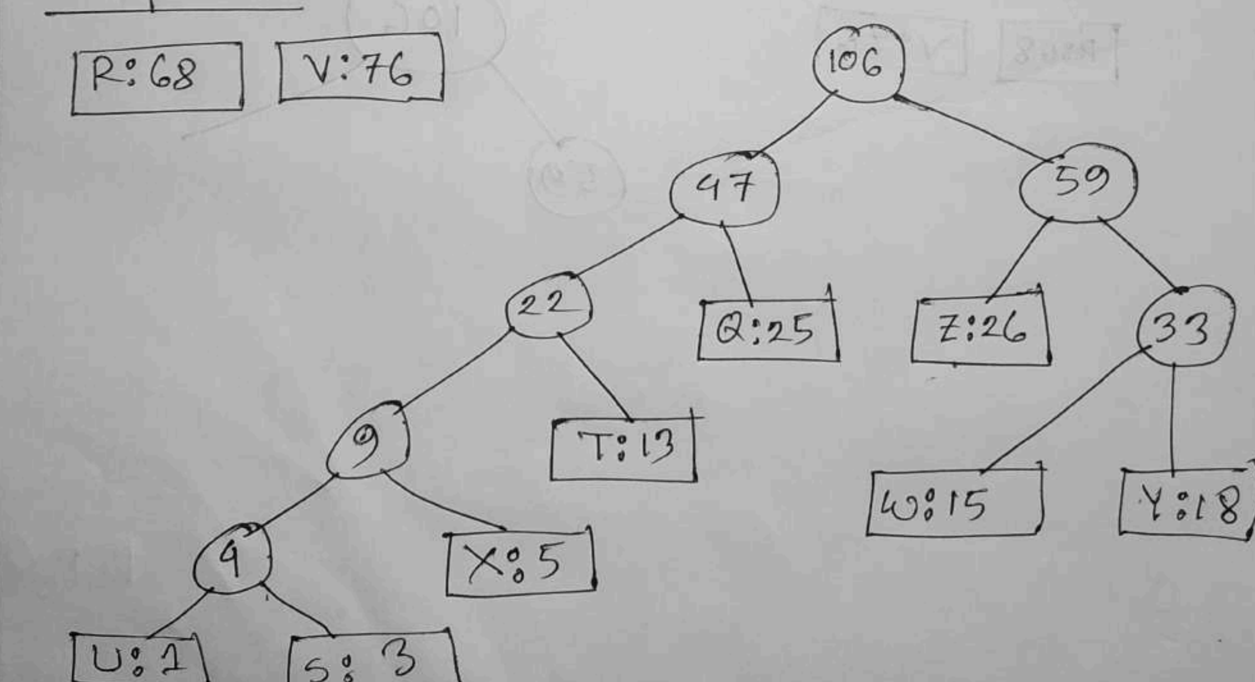
P.T.O

Subject : Page -20 Date :

Step-7:

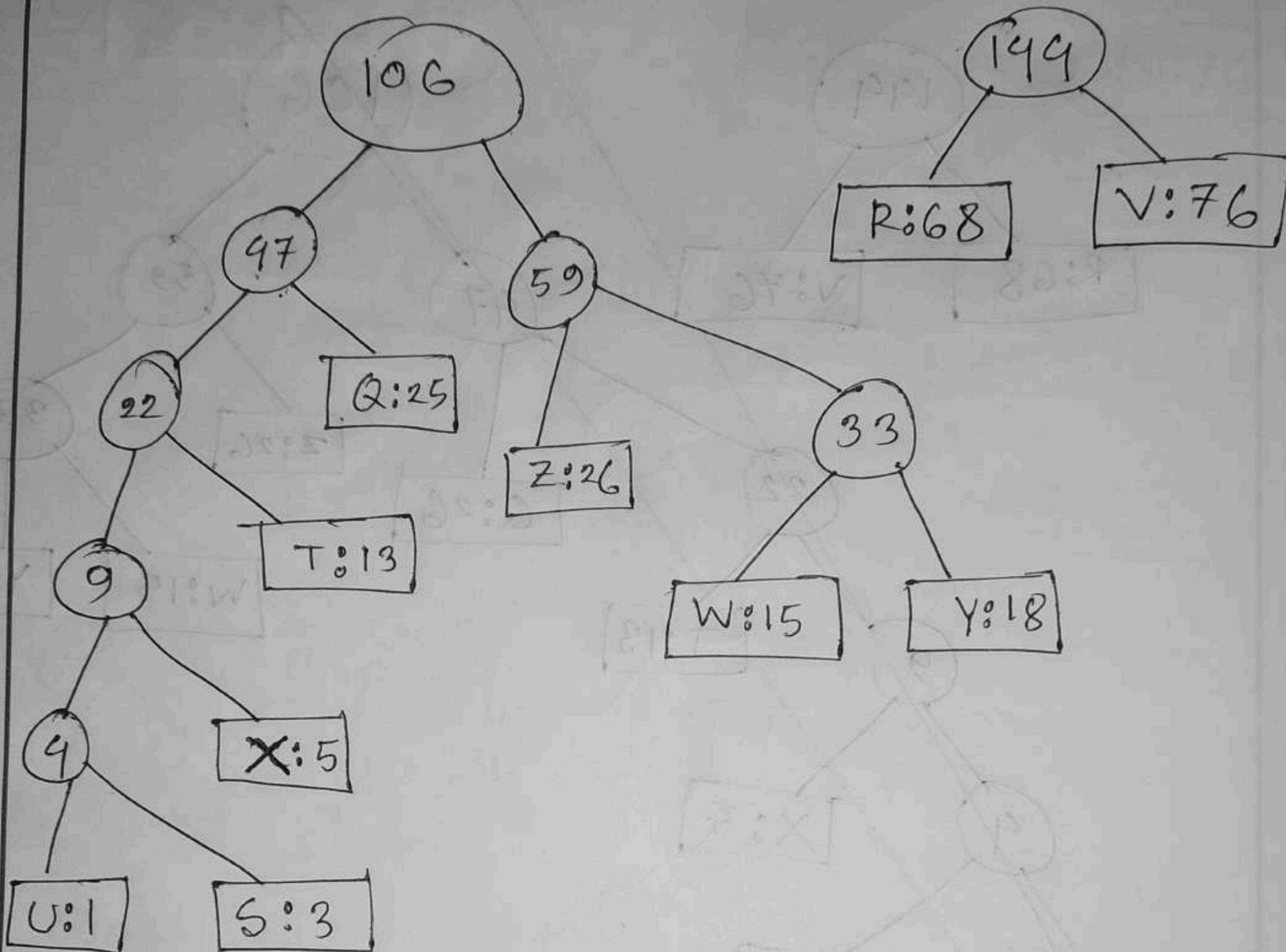


Step-8:



Subject : Page — 21 Date :

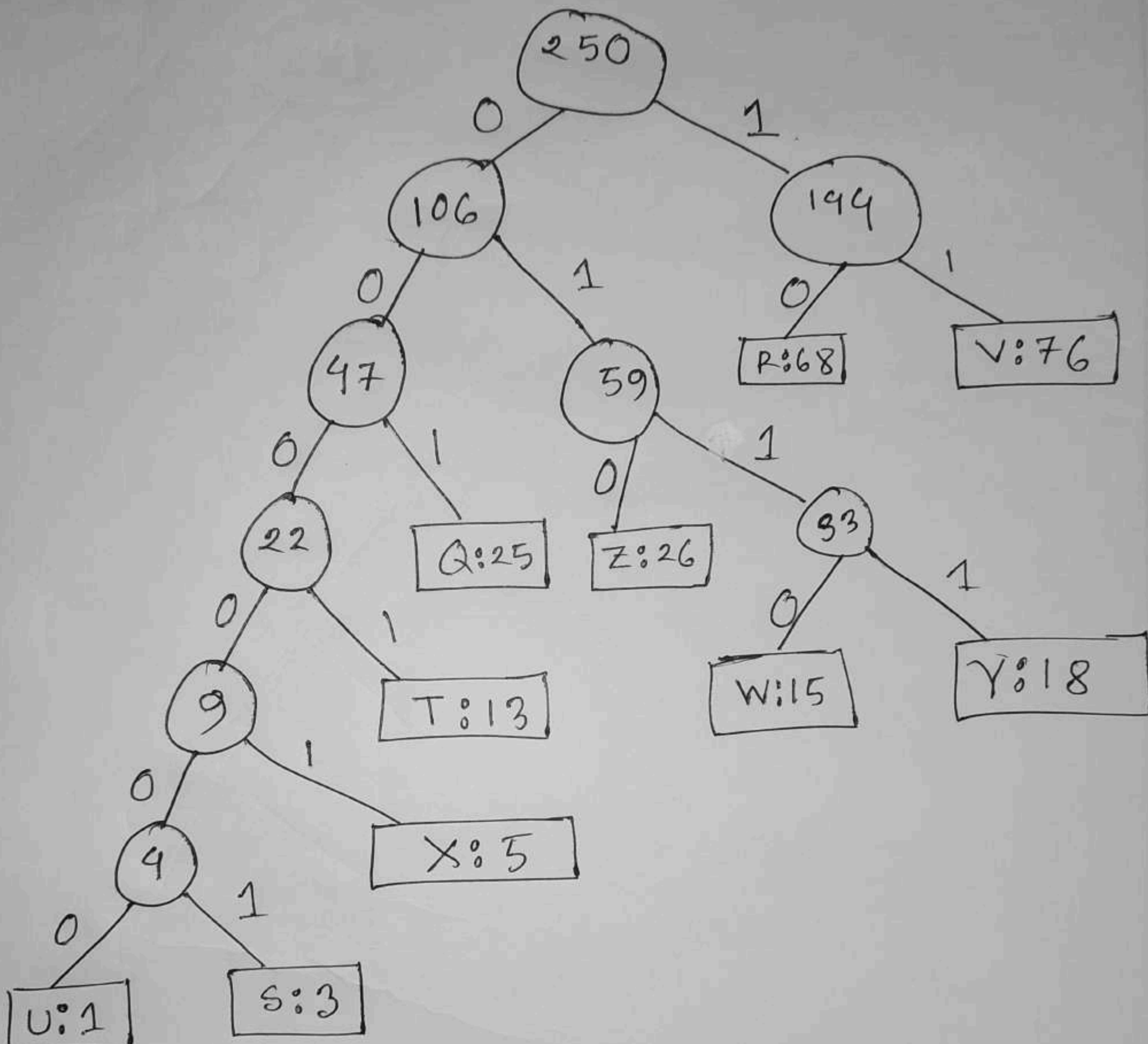
Step-9 :



P.T.O

Tanha Enterprise

Step-10 :



The Huffman codes are :

R = 10, V = 11, Z = 010, Q = 001

T = 0001, W = 0110, Y = 01111,

X = 00001, U = 000000 and

S = 000001

Ans.the.Q.NO:2

(b)

Divide and conquer

- Partion the problem into independent subproblems, solve the subproblems recursively, and then combine their solutions to solve the original problem

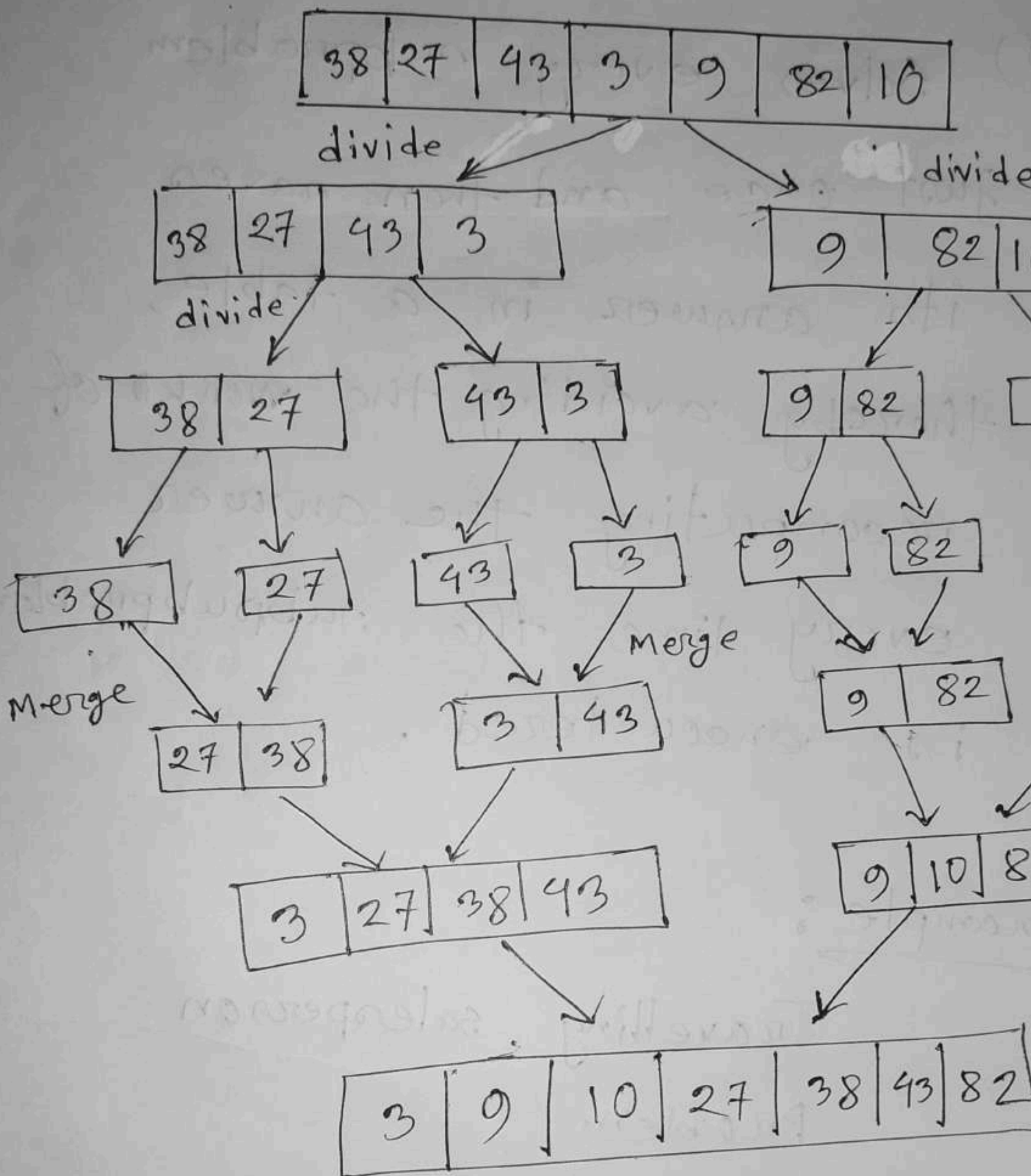
Example:

Merge sort

P.T.O

Subject :

Date :



Dynamic approch:

(i) Applicable when the subproblems are not independent, that is, when subproblems share subproblems.

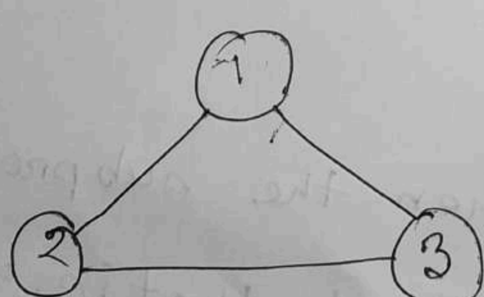
P.T.O

Tanha Enterprise

(ii) Solves every subproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time the subsubproblem is encountered.

Example:

Travelling salesperson problem



$$\begin{bmatrix} 0 & 2 & 3 \\ 5 & 0 & 9 \\ 6 & 13 & 0 \end{bmatrix}$$

$g(2, \varphi) = c_{21} = 5$

$g(3, \varphi) = c_{31} = 6$

$g(2, \{3\}) = c_{23} + g(3, \{\varphi\})\}$

$= (9 + 6) = 15$

$g(3, \{2\}) = c_{32} + g(2, \{\varphi\})\}$

$= (13 + 5) = 18$

$g(1, \{2, 3\}) = \min\{c_{12} + g(2, \{3\}),$

$c_{13} + g(3, \{2\})\}$

$= \min\{(2 + 15), (3 + 18)\}$

$= 17$

Path : $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$

Cost : 17 (Ans.)

Ans. the. Q. NO: 2

(c)

Matrix-chain-ORDER(P)

$n \leftarrow length[P]-1$

for $i \leftarrow 1$ to n do

$m[i,i] \leftarrow 0$

for $l \leftarrow 2$ to n do

for $i \leftarrow 1$ to $n-l+1$ do

$j \leftarrow i+l-1$

$m[i,j] \leftarrow \infty$

for $k \leftarrow i$ to $j-1$ do

$q \leftarrow m[i,k] + m[k+1,j] +$

$p_{i-1} p_k p_j$

P.T.O

if $q < m[i, j]$ then

~~m[i, j] then~~

$m[i, j] \leftarrow q$

$s[i, j] \leftarrow k$

return m and s

Ans. the. Q. NO: 2

(d)

This algorithm searches for the local optima and optimizes the local best solution to find the global optima. It begins by sorting all the edges and then selects the edge with the minimum cost. It continuously selects the best next choices given a condition that no loops are formed. The complexity of the greedy algorithm is $O(N^2 \log_2 N)$ and there is no guarantee that a global optimum solution is found.