UCS415 MST with solutions

Computer Engineering (Thapar Institute of Engineering and Technology)

Roll Number:_______________________

Thapar Institute of Engineering & Technology

Department of Computer Science and Engineering

MID SEMESTER EXAMINATION

B. E. **(2nd Yr. COE/CSE, 3rd Yr. EM)**
16th March, 2023
Thursday, Time– 10:30 To 12:30 PM
Time: 2 Hours, Max Marks: 25

Course Code: UCS415
Course Name: Design and Analysis of Algorithms
Name of Faculty: Rajesh Mehta, Tarunpreet Bhatia, Randheer Bagi, Anil Singh, Vaibhav Pandey, Manisha Panjeta, Shruti Aggarwal

Note: Attempt all Questions in sequence. Answer all sub-parts of each question at one place. Do mention Page No. of your attempt at front page of your answer sheet. Assume missing data (if any).

Q.1	Solve the following recurrence relation using substitution/back-substitution method. $T(n) = 5T\left(\frac{n}{5}\right) + \frac{n}{\log n}$,where T(1) = 1	(3)

Solution

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ 5T\left(\frac{n}{5}\right) + \frac{n}{\log n} & n>1 \end{cases}$$

$$T(n) = 5T\left(\frac{n}{5}\right) + \frac{n}{\log n}$$

$$= 5\left[5T\left(\frac{n}{5^2}\right) + \frac{n/5}{\log\left(\frac{n}{5}\right)}\right] + \frac{n}{\log n}$$

$$= (5)^2 T\left(\frac{n}{5^2}\right) + \frac{n}{\log\left(\frac{n}{5}\right)} + \frac{n}{\log n}$$

Up to this step (1 mark)

$$= (5)^2\left[5T\left(\frac{n}{5^3}\right) + \frac{n/(5)^2}{\log\left(\frac{n}{5^2}\right)}\right] + \frac{n}{\log\left(\frac{n}{5}\right)} + \frac{n}{\log n}$$

$$= (5)^3 T\left(\frac{n}{5^3}\right) + \frac{n}{\log\left(\frac{n}{5^2}\right)} + \frac{n}{\log\left(\frac{n}{5}\right)} + \frac{n}{\log n}$$

k times

$$\frac{n}{5^k} = 1 \Rightarrow \log n = k \log_2 5$$

$$k = \frac{\log_2 n}{\log_2 5} = \log_5 n$$

$$= (5)^k T\left(\frac{n}{5^k}\right) + n\left[\frac{1}{\log\left(\frac{n}{5^{k-1}}\right)} + \frac{1}{\log\left(\frac{n}{5^{k-2}}\right)} + \cdots \frac{1}{\log\left(\frac{n}{5}\right)} + \frac{1}{\log n}\right]$$

1 mark

$$= n(1) + n\left[\frac{1}{(\log n - 0)} + \frac{1}{(\log n - 1)} + \cdots \frac{1}{\log n - (\log n - 1)}\right]$$

$$= n + n\left[\frac{1}{1} + \frac{1}{2} + \cdots \frac{1}{\log n - 1} + \frac{1}{\log n}\right]$$

$$= n + n \log\log n$$

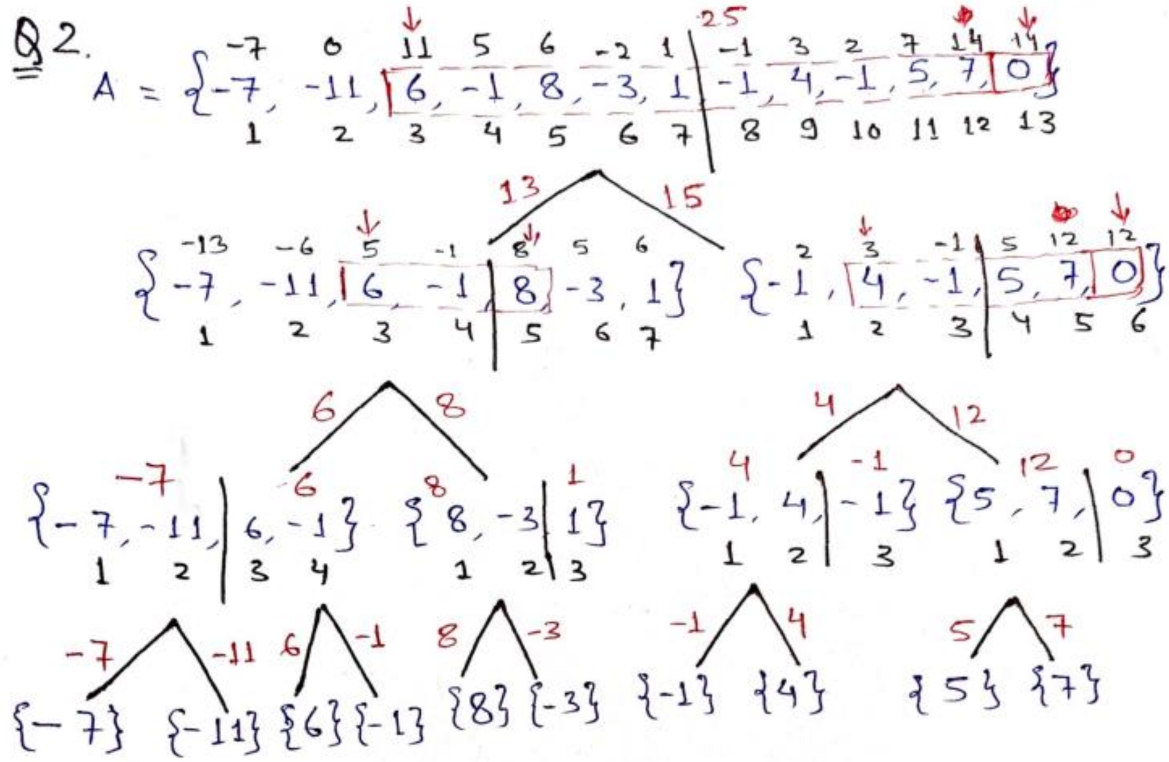
$$= O(n \log\log n)$$

1 mark

Q.2 Consider a sequence of n elements ($n = 13$), $A = \{-7, -11, 6, -1, 8, -3, 1, -1, 4, -1, 5, 7, 0\}$. You have to determine the maximum $Sum(i, j)$ where $Sum(i, j) = \sum_{k=i}^{j} A[k]$ *and* $1 \le i \le j \le n$ using Divide and Conquer approach. Show algorithmic steps with the help of appropriate data structure and write the recurrence relation. **(3)**

Solution

Q2.



Left sum = 13

Right sum = 15

Cross Sum = 25

Max Sub-array = {6, -1, 8, -3, 1, -1, 4, -1, 5, 7}

max sum = max (13, 15, 25) $\longrightarrow$ Marks = 2

= 25

Recurrence Relation:

$$\boxed{T(n) = 2T(n/2) + n}$$ $\longrightarrow$ Marks = 1

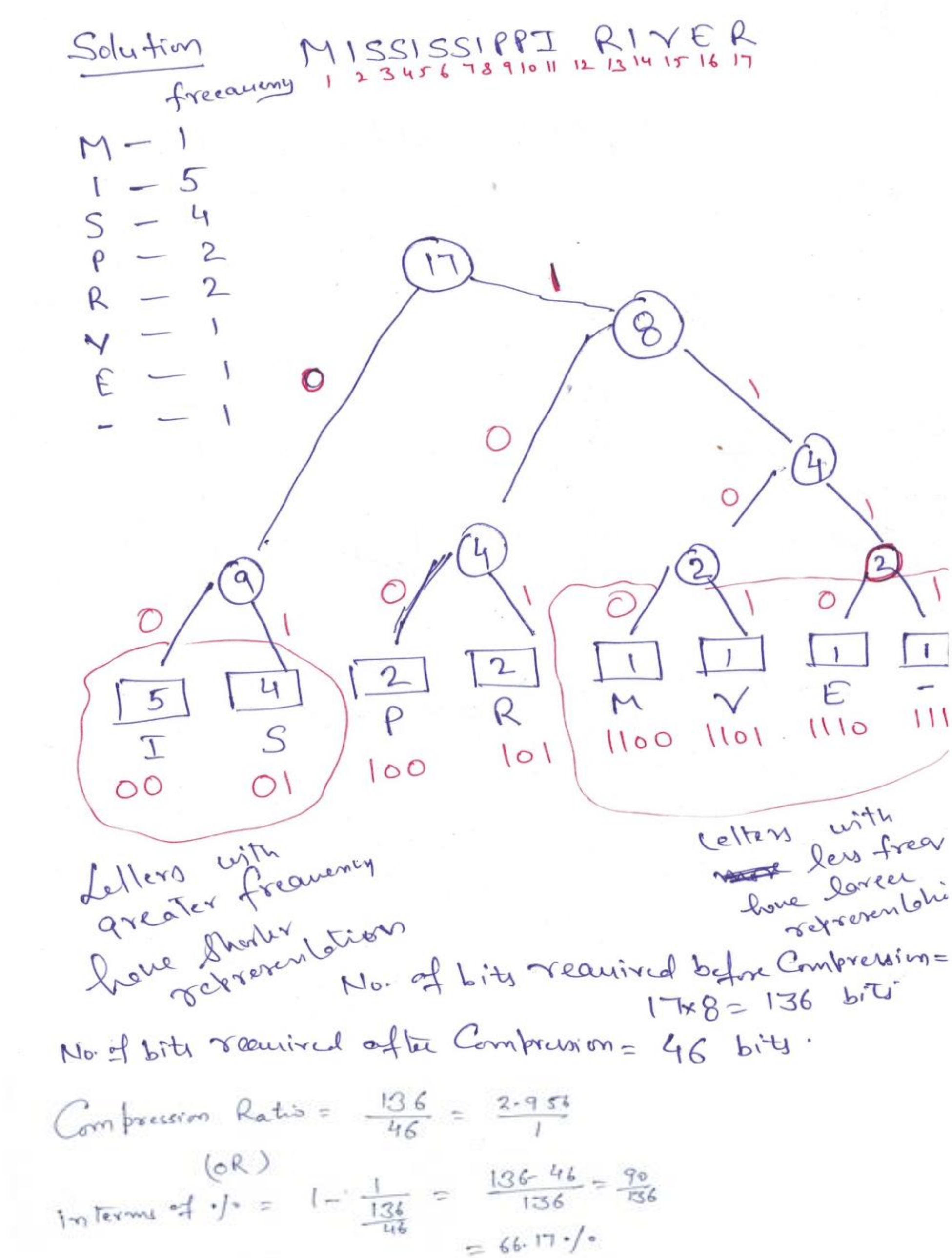
Total Marks = 3

Q.3 Consider the text message as shown in the Table 1. Apply and verify the text compression algorithm which is based on idea that more number of common letters required fewer bits and less common letters required more number of bits. Assume that the blank space is considered as "-" with frequency 1. Draw the appropriate data structure to find the length of message after compression. How many bits are required for transferring this message before and after compression? Also, deduce the compression ratio. **(4)**

Table 1

M	**I**	**S**	**S**	**I**	**S**	**S**	**I**	**P**	**P**	**I**		**R**	**I**	**V**	**E**	**R**

Solution



Note: Different tree structures can be drawn. But bits required to represent the symbols in every representation is same.

Marking Scheme:

Upto tree structure and bits representation : **2 marks**

No. of bits required before and after compression: **1 mark (1/2 +1/2)**

Compression ratio : **1 mark**

Q.4	LPS (Longest Palindrome Subsequence) is the maximum length of a common subsequence of a given sequences/strings which is palindrome. For example if the input string is "ABCDAC" then the longest palindrome subsequence is "CAC". Design and apply an efficient algorithm using dynamic programming (DP) approach for the LPS problem on the given input string. Compare the worst case run time complexity of DP and Brute Force Method for LPS problem.	(4)

Solution

Marks Division: LPS DP Algorithm [2 marks]
Applying given algorithm [1 mark]
Complexity Analysis [1 mark]

Solution:

a) Algorithm LPS(str, n){

```
        DP[n][n];
        For (i = 0 to n-1)
                DP[i][i] = 1;
        For (k = 2 to n)
                For (i = 0 to n-k)
                        j = i + k -1;
                        if ( str[i] == str[j] && k == 2)
                                DP[i][j] = 2;
                        else if (str[i] == str[j])
                                DP[i][j] = DP[i+1][j-1] +2;
                        else
                                DP[i][j] = max(DP[i][j-1],  DP[i+1][j]) ;
return DP[0][n-1];
}
```

LPS problem can be solved by LCS DP algorithm by considering the input as first string and reverse of input as second string.

Complexity Analysis:
DP: The size of computed matrix is n x n hence we will need to compute n^2 cells of the matrix. Hence the complexity for dynamic programming approach will be O(n^2).

Brute force: In worst case scenario the input string will not have any common character hence each function call will further call two recursive calls.
The recurrence relation will be T(n) = 2T(n-1) +c
If we solve this recurrence relation the complexity will be 2^n.

Q.5	Given a sequence of 4 matrices A_1, A_2, A_3 and A_4 and an array of dimensions as d[] = {5, 4, 3, 2, 6}. Your task is to find an optimal parenthesization of multiplying given matrices so that it would take a minimum number of multiplications using bottom-up approach. You need to show the intermediate computations.	(4)

Solution

2 marks for correct entries in m[][] with steps

1 mark for showing steps to calculate parenthesis

1 mark for correct entries in s[][] and final answer

$m[1][2] = m[1,1] + m[2, 2] + d_0d_1d_2 = 0 + 0 + 5.4.3 = 60$

$m[2][3] = m[2,2] + m[3, 3] + d_1d_2d_3 = 0 + 0 + 4.3.2 = 24$

$m[3][4] = m[3,3] + m[4, 4] + d_2d_3d_4 = 0 + 0 + 3.2.6 = 36$

0.5 marks for above 3 entries

$m[1, 3] = \min\{m[1,1] + m[2,3] + d_0d_1d_3, m[1,2] + m[3,3] + d_0d_2d_3\}$

$= \min\{64, 90\} = 64$

$m[2, 4] = \min\{m[2,2] + m[3,4] + d_1d_2d_4, m[2,3] + m[4,4] + d_1d_3d_4\}$

$= \min\{108, 72\} = 72$

$m[1, 4] = \min\{m[1,1] + m[2,4] + d_0d_1d_4, m[1,2] + m[3,4] + d_0d_2d_4, m[1, 3] + m[4, 4] + d_0d_3d_4\}$

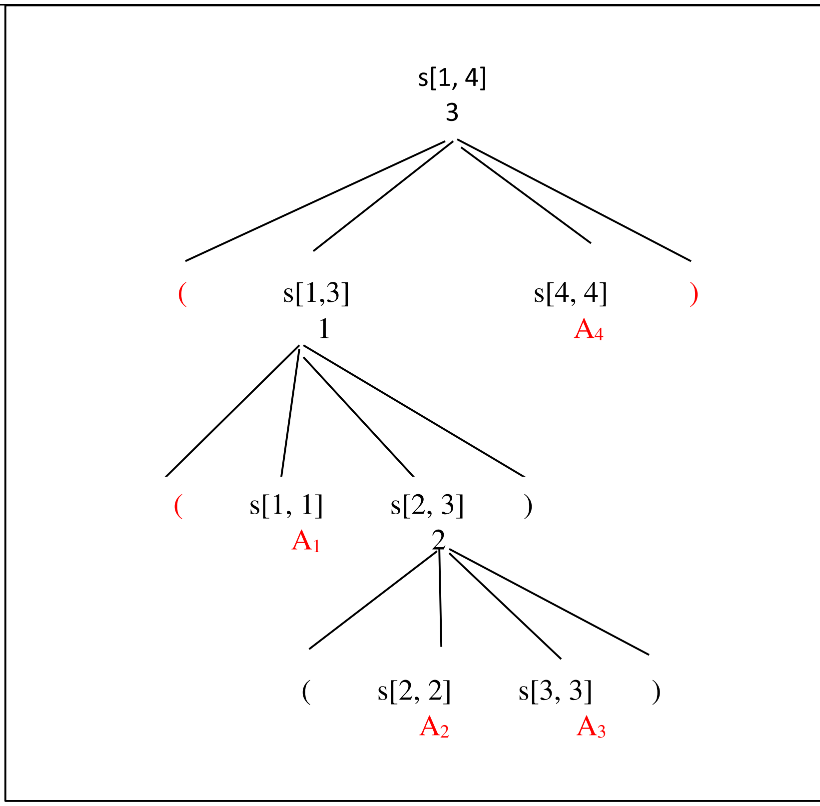
$= \min\{192, 186, 124\} = 124$

1.5 marks for above 3 entries

	1	2	3	4
1	0	60	64	124
2		0	24	72
3			0	36
4				0

s[][]

	1	2	3	4
1	1	1	1	3
2		2	2	3
3			3	3
4				4



Optimal parenthesis = ((A_1 (A_2, A_3)) A_4)

Q.6 Consider the four items with given weights and profits (as shown in Table 2) in the context of 0-1 Knapsack problem. **(4)**

Table 2

Item	1	2	3	4
Weight	5	3	4	2
Profit	5	6	8	3

a) Write the recursive equation to find the optimum value of items that can be packed in a knapsack of a given capacity using bottom-up approach. Also find the optimum value of items that can be packed in a knapsack of capacity of 7 (seven).

b) Also, write the algorithm for finding the optimal set of items in the knapsack.

Solution

a) Recursive equation:

$$B[k,w]=\begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k]+b_k\} \end{cases}$$

Calculation of optimal value:

Items	0	1	2	3	4	5	6	7 ←capacity
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	5	5	5
2	0	0	0	6	6	6	6	6
3	0	0	0	6	8	8	8	14
4	0	0	3	6	8	9	11	14

Optimal value = 14

	b) Algorithm for evaluating optimal set of items: i = n , k = W while i, k > 0 if $B[i, k] \neq B[i\text{-}1, k]$ then *mark the i^{th} item as in the knapsack* $i = i\text{-}1,\ k = k\text{-}w_i$ else $i = i\text{-}1$ Marking scheme: 6 a) 3 marks (1 mark for recursive equation and 2 marks for optimal value) 6 b) 1 mark for pseudo code	
Q.7	An interview is scheduled on a particular day where each candidate is to be interviewed by a single interviewer. The start time and end time of the interview for each candidate is given in Table 3. Design and apply the efficient algorithm for determining the minimum number of interviewers to interview all the candidates, with a minimum of 3 candidates for each interviewer. You need to also show the candidate list for each interviewer.	(3)

Table 3

Candidate	Interview Start Time	Interview End Time
1	9: 10 am	10:10 am
2	11: 00 am	11: 20 am
3	10:30 am	10: 45 am
4	12: 00 pm	12: 30 pm
5	11:20 am	11: 55am
6	10:10 am	10: 30 am
7	10: 45 am	11: 15 am
8	11:30 am	12:00 pm
9	9:00 am	10: 00 am
10	12:40 pm	1:00 pm
11	10:10 am	11:00 am
12	9:55 am	10: 30 am
13	11:05 am	11:35 am
14	11:35 am	12: 00 pm

Solution

Algorithm 1.5 marks

3 interviewer schedule 0.5 each

The algorithm should use activity selection using a greedy approach to schedule candidates and allocate them to interviewers:

Sort the list of candidates by their end time/start time in ascending order.
Initialize an empty list of interviewers.
For each candidate in the sorted list:
a. If the list of interviewers is empty, create a new interviewer and add this candidate to their schedule.
b. Otherwise, for each interviewer in the list:
i. If the interviewer's schedule has less than 3 candidates, and the candidate's time slot does not overlap with any of the scheduled candidates' time slots, add this candidate to the interviewer's schedule and break the loop.
ii. If all interviewers have schedules with 3 or more candidates or if the candidate's time slot

	overlaps with any of the scheduled candidates' time slots, create a new interviewer and add this candidate to their schedule. Output the interview schedule and candidate list for each interviewer. No of interviewers required: 3 Interviewer 1: Candidate No. : 9,6,3,2,5,4,10 Interviewer 2: Candidate No. : 1,11,13,14 Interviewer 3: Candidate No. : 12,7,8	