USN | | | | | | | | | |

CMRIT — CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – July 2022

Sub:	Design and Analysis of Algorithms				Sub Code:	18CS42	Branch:	CSE
Date:	08/07/2022	Duration:	90 mins	Max Marks: 50	Sem/Sec:	IV/A,B&C		OBE

<u>Answer any FIVE FULL Questions</u>

		MARKS	CO	RBT
1	a. Define an algorithm. What are the characteristics of an algorithm?	[5]	CO1	L1
	Ans: An algorithm is a finite set of instructions that accomplishes a particular task. An algorithm must satisfy the following criteria Input: Zero or more quantities are externally supplied Output: At least one quantity is produced Definiteness: Each instruction is clear and unambiguous Finiteness: For all case, algorithm terminates after a finite number of steps Effectiveness: Every instruction must be basic			
	b. Explain the two common ways of representing a graph with an example	[5]	CO1	L1
	Ans: Adjacency matrix and adjacency list with examples			
2	Write an algorithm to find the maximum elements in an array. Identify the basic operation in the algorithm and identify its worst case, average case, and best case time complexity	[4+2+2+2]	CO2	L2

	ALGORITHM MaxElement(A[0..n − 1])
	//Determines the value of the largest element in a given array
	//Input: An array A[0..n − 1] of real numbers
	//Output: The value of the largest element in A
	maxval ←A[0]
	for i ←1 to n − 1 do
	    if A[i]>maxval
	        maxval←A[i]
	return maxval
	
	Taking comparison operator as the basic operation, n-1 comparisons are made irrespective of the input pattern. Hence the worst case, average case and best case all O(n)			

3	a. Explain any three asymptotic notations.	[6]	CO2	L1
	O notation: A function t (n) is said to be in O(g(n)), if there exist some positive constant c and some nonnegative integer n0 such that $t(n) \le cg(n)$ for all $n \ge n_0$. Θ Notation: A function t (n) is said to be in Θ (g(n)), if there exist some positive constants c1 and c2 and some nonnegative integer n0 such that $c_2 g(n) \le t(n) \le c_1 g(n)$ for all $n \ge n_0$. Ω-notation: A function t (n) is said to be in Ω (g(n)), if there exist some positive constant c and some nonnegative integer n0 such that $t(n) \ge cg(n)$ for all $n \ge n_0$.			
	b. State with reasons which of the following are true i. 0.3n(n-1) belongs to Theta(n^2) ii. 1/2n(n+1) belongs to Theta(n^2) iii. 100n+5 belongs to O(n^2). iv. 555nlogn belongs to O(n^2)	[4]	CO2	L3

	i. 0.3n(n-1) belongs to Theta(n^2) (TRUE) ii. 1/2n(n+1) belongs to Theta(n^2) (TRUE iii. 100n+5 belongs to O(n^2). (TRUE). FLASE will be given 0.5 marks if you say it belongs to O(n) iv. 555nlogn belongs to O(n^2) (TRUE). FLASE will be given 0.5 marks if you say it belongs to O(n log n)			
4	What is Decrease and Conquer? What are its 3 major variations? Explain with an example for each.	[10]	CO2	L2
	The decrease-and-conquer technique is based on exploiting the relationship between a solution to a given instance of a problem and a solution to its smaller instance. (2.5 mark) There are three major variations of decrease-and-conquer: Decrease by a constant (2.5 mark) Example: Exponentiation a^b = a^(b-1) * a Decrease by a constant factor (2.5 mark) Example: Exponentiation a^b = (a^(b/2))^2 if b is even a^b = a * (a^(b/2))^2 if b is odd Variable size decrease (2.5 mark) Example: GCD computation: gcd(m, n) = gcd(n, m mod n)			
5	a. Explain the general plan for analyzing the efficiency of a recursive algorithm.	[4]	CO2	L1
	Decide on a parameter indicating an input's size. Identify the algorithm's basic operation. Check whether the number of times the basic op. is executed may vary on different inputs of the same size. (If it may, the worst, average, and best cases must be investigated separately.) Set up a recurrence relation with an appropriate initial condition expressing the number of times the basic op. is executed. Solve the recurrence (or, at the very least, establish its solution's order of growth) by backward substitutions or another method.			
	b. Consider the following recursive algorithm. ALGORITHM Q(n) //Input: A positive integer n if n = 1 return 1 else return Q(n − 1) + n + 1 Set up a recurrence relation for the number of additions made by this algorithm and solve it.	[6]	CO2	L3
	Recurrence equation is T(n) = 0 for n=0; T(n) = T(n-1) + 2 for n>1 Solve using backward substitution T(n) = T(n-1) + 2 = [T(n-2) +2] + 2 = T(n-2) + 4 = [T(n-3) +2] + 2 = T(n-3) + 6 . . . = T(n-i) + 2i When i = n-1, equation becomes T(n) = T(n –(n-1) + 2(n-1) = T(1) + 2(n-1) Since T(1) is 0, T(n) = 2(n-1). Hence the algorithm is in Θ(n).			

6	Discuss how quick sort works to sort an array and trace for the following data. Draw the tree of recursive calls made.	[10]	CO1	L2

43	27	75	34	85	60	59	50	45