MAY 2023 EXAMINATION
III B.E./ B.Tech. (4YDC) EXAM
CO34563/CO3463: DESIGN AND ANALYSIS OF ALGORITHMS

Time: 3 Hrs.] **[Max. Marks: 70**
[Min. Marks: 22

Note: Attempt all the questions. Each question have five sub-parts a, b, c, d and e. Sub-part a, b and c are compulsory and attempt any one part out of (d) and (e). Attempt all sub-parts of a question at one place.

Marks | CO | BL | PI-Code

Q. 1. (a) For each of the following questions, write either T (True) or F (False), and briefly justify your answer (a single sentence or picture should be sufficient). [2] CO1 BL3 1.2.1

(i) If $T(n) = \frac{9}{4} T\left(\frac{2n}{3}\right) + n^2$ and $T(1) = \Theta(1)$, then $T(n) = O(n^2)$.

(ii) n is $O((\log n)^{\log n})$

(b) If you are sorting a million items, roughly how much faster is heap-sort than insertion sort? (Note: log(1,000,000) = 20.) [2] CO1 BL2 4.2.1

(c) What is the output of the following function fun(arr, n)? And also give time complexity of the function. Where arr is an array of size n, and the function is defined as follows: [3] CO1 BL4 4.1.2

```
int fun ( arr , n)
{
 int a = 0; int b = 0; int c; int x = 0;
        for (int i = 0; i < 10; i++)
         {
            x = arr[i];
            b |= a & x;
            a ^= x;
            c = a & b;
            a &= ~c;
            b &= ~c;
        }
         return a;
}
```

(d) Let A be an array of n pairs of positive integers (x_i, y_i) with $x_i, y_i < n^2$ for all $i \in \{0,...,n-1\}$. The power of pair (x, y) is the integer $x+n^y$. Design an O(n)-time algorithm to sort the pairs in A increasing by power. [7] CO2 BL4 3.2.2

OR

(e) A Pythagorean Quad consists of four integers (a, b, c, d) such that $d^2=a^2+b^2+c^2$. Given an array A containing n distinct positive integers, Design an $O(n^2)$-time algorithm to determine whether four integers from A form a Pythagorean Quad, where integers from A may appear more than once in the Quad. Also give running time is worst-case of algorithm. [7] CO1 BL4 3.2.3

Q. 2. (a) For the following recurrences give the time and space that would be required by a simple dynamic programming algorithm to compute the answer where the values of the w function are given as input. Compute OPT(1; n) where OPT(i; i) = 0 for all i, and [2] CO1 BL2 4.1.1

OPT(i; j) = min { OPT(i; k) + OPT(k + 1; j) + w(k) : i ≤ k < j } for all i < j.

(b) For each of the following questions, write either T (True) or F (False), and briefly justify your answer (a single sentence or picture should be sufficient). [2] CO4 BL2 2.6.2

(i) Given an array A containing n comparable items, sort A using merge sort. While sorting, each item in A is compared with O(log n) with other items of A.

(ii) If there is an algorithm to solve 0-1 Knapsack in polynomial time, then there is also an algorithm to solve Subset Sum in polynomial time.

(c) Improve the following code using code tuning techniques, where ‘a’ is array of n integers: [3] CO2 BL3 2.3.1

```
While (i< n) and (x != a[i])
{
 If (√x < √a[i] )
   i= i+1;
 else
  i=i+2;
}
```

(d) You given as input n real numbers $x_1, \ldots x_n$. Design an efficient algorithm that uses the minimum number m of unit intervals $[\ell_i, \ell_i+1)$ $(1 \le i \le m)$ that cover all the input numbers. A number x_j is covered by an interval $[\ell_i, \ell_i+1)$ if $\ell_i \le x_i < \ell_i + 1$.For example, consider the input for n = 4: {1.1, 0.2, 1.555, 0.9}. Then the two intervals [0.1, 1.1) and [1.1, 2.1) cover all the input numbers (i.e. in this case m = 2). [7] CO2 BL6 3.4.3

OR

(e) Arpit and Abhinav are playing a game on a binary string S of length N. Arpit and Abhinav make alternating moves with Arpit going first. In one move, a player will select an index i $(1 \le i < N)$ such that $S_i \ne S_{i+1}$, and delete both S_i and S_{i+1} from the string S, and String is combine at deletion point. Note that N gets reduced by 2 when both characters are deleted. If a player cannot select any such index i, he loses the game. Design an algorithm to determine the winner of the game if both players play optimally. [7] CO2 BL6 3.2.4

Q. 3. Consider the longest increasing subsequence problem defined as follows. Given a list of numbers $\{a_1; : : : ; a_n\}$, an increasing subsequence is a list of indices $i_1; : : : ; i_k \in \{1; : : : ; n\}$ such that $i_1 < i_2 < : : : ; i_k$ and $a_{i1} \le a_{i2} \le : : : ; \le a_{ik}$. The longest increasing subsequence is the longest list of indices with this property. Give the answers of the following:

(a) What is the longest increasing subsequence of the list 5; 3; 4; 8; 7; 10? [2] CO2 BL3 1.8.1

(b) Consider the greedy algorithm that chooses the first element of the list, and then repeatedly chooses the next element that is larger. Is this a correct algorithm? Either prove its correctness or provide a counter example. [2] CO3 BL4 2.8.1

(c) Consider the greedy algorithm that chooses the smallest element of the list, and then repeatedly chooses the smallest element that comes after this chosen one. Is this a correct algorithm? Either prove its correctness or provide a counter example. [3] CO3 BL4 2.8.1

(d) Consider a divide and conquer strategy that splits the list into the first half and second half, recursively computes $L = (\ell_i \ldots. \ell_{KL})$;$R = (r_1; : : : ; r_{kR})$ the longest increasing subsequences in each half, and then, if the last chosen element in the first half is less than the first chosen index in the second half (i.e. $a_{\ell kL} \le a_{r1}$) returns L * R, otherwise it returns the longer of L and R. Is this a correct algorithm? Either prove its correctness of provide a counterexample. [7] CO3 BL6 3.2.1

OR

(e) Design a dynamic programming algorithm for longest increasing subsequence. Prove its correctness and analyze its running time. [7] CO3 BL1 3.5.1

Q. 4.	**(a)**	Suppose a dynamic programming algorithm creates an nxm table and to compute each entry of the table it takes a minimum over at most m (previously computed) other entries. What would the running time of this algorithm be, assuming there is no other computations.	**[2]**	**CO1**	**BL3**	**2.5.2**
	(b)	Find the lower bound of the following recurrence problems. $T(n) = 1$ if $n = 1$ $= \sum_{k=1}^{n-1} T(k)T(n-k)$ if $n \geq 2$	**[2]**	**CO1**	**BL2**	**4.5.1**
	(c)	Suppose we wish to multiply three matrices of real numbers M1 x M2 x M3, where M1 is 20 by 10, M2 is 10 by 20, and M3 is 20 by 50. Find the total number of multiplications and optimal order in which to multiply the matrices so as to minimize the total number of scalar operations.	**[3]**	**CO2**	**BL3**	**1.2.1**
	(d)	Paint a row of n houses red, green, or blue so that no two adjacent houses have the same color. Design an algorithm to minimize total cost of the paint, where cost(i, color) is cost to paint i^{th} house with color is given.	**[7]**	**CO3**	**BL6**	**3.5.1**

	A1	A2	A3	A4	A5	A6	……
Red	7	6	7	8	9	20	
Green	3	8	9	22	12	8	
Blue	16	10	4	2	5	7	

OR

	(e)	You are given an exam with questions numbered 1, 2, 3, . . . , n. Each question i is worth p_i points. You must answer the questions in order, but you may choose to skip some questions. The reason you might choose to do this is that even though you can solve any individual question i and obtain the p_i points, some questions are so frustrating that after solving them you will be unable to solve any of the following f_i questions. Suppose that you are given the p_i and f_i values for all the questions as input. Devise the most efficient algorithm you can for choosing set of questions to answer that maximizes your total points, and compute its asymptotic worst case running time as a function of n.	**[7]**	**CO3**	**BL6**	**3.6.2**
Q. 5.	**(a)**	Write Cooks theorem. Explain significance of Cooks theorem.	**[2]**	**CO4**	**BL1**	**1.6.1**
	(b)	What are NP complete problems set? What is the principle of reducibility?	**[2]**	**CO4**	**BL1**	**1.6.1**
	(c)	Differentiate between parallel and data streaming algorithm with some suitable example.	**[3]**	**CO4**	**BL2**	**2.4.1**
	(d)	Recall that the NP-complete SUBSET-SUM problems asks, given a set of nonnegative integers S and a target K, whether S has a subset S' with $\sum_{i \in S'} i = K$. The AVERAGE-SUM problem asks, given just a set of non-negative integers S, whether S has a subset S'' with $\sum_{i \in S'} i = \frac{1}{\lvert S \rvert} \sum_{i \in S} i$, where |S| is the number of elements in S. It is similar to the SUBSET-SUM problem, except now the target value is always the average value in S. Give a polynomial-time algorithm for AVERAGE-SUM, or prove that it is NP-complete.	**[7]**	**CO4**	**BL2**	**1.7.1**

OR

	(e)	The Number Partition problem asks, given a collection of non-negative integers $y_1; : : : ; y_n$ whether or not it is possible to partition these numbers into two groups so that the sum in each group is the same. Prove that Number Partition is NP-complete by solving the following problems. (a) Show that Number Partition is in NP. (b) Show that Subset Sum ≤ P Number Partition	**[7]**	**CO4**	**BL2**	**1.7.1**

****

BL – Bloom's Taxonomy Levels

BL1 - Remembering, BL2 - Understanding, BL3 - Applying,
BL4 - Analyzing, BL5 - Evaluating, BL6 - Creating

CO – Course Outcomes

CO1. Describe the meaning of complexity of an algorithm & various notations to represent time and space complexity.

CO2. Apply and evaluate different algorithm design techniques for getting the effective solutions of specified problems.

CO3. Design and analyze different algorithms with its applications.

CO4. Explain the computability and non-computability and various complexity classes, and approaches to solve complex problems.

PI Code – Performance Indicator Code

1.x.y: Engineering knowledge:

2.x.y: Problem analysis:

3.x.y: Design/Development of Solutions:

4.x.y: Conduct investigations of complex problems:

5.x.y: Modern tool usage: