VIT

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

Final Assessment Test(FAT) - Apr/May 2025

Programme	**B.Tech.**	Semester	**Winter Semester 2024-25**
Course Code	**BCSE204L**	Faculty Name	**Prof. Lekshmi K**
Course Title	**Design and Analysis of Algorithms**	Slot	**A2+TA2**
		Class Nbr	**CH2024250501871**
Time	**3 hours**	Max. Marks	**100**

Instructions To Candidates

- Write only your registration number in the designated box on the question paper. Writing anything elsewhere on the question paper will be considered a violation.

Answer all the EIGHT questions.

- If any assumptions are required, assume the same and mention those assumptions in the answer script.
- Use of intelligence is highly appreciated.
- Your answer for all the questions should have both the 'design' component and the 'analysis component'
- The 'Design' component should consist: understanding of the problem, logic to develop the algorithm, illustration, pseudocode.
- The 'Analysis' component should consist: Computation of T(n), Time-complexity.

Course Outcomes

CO1: Apply the mathematical tools to analyze and derive the running time of the algorithms
CO2: Demonstrate the major algorithm design paradigms.
CO3: Explain major graph algorithms, string matching and geometric algorithms along with their analysis.
CO4: Articulating Randomized Algorithms.
CO5: Explain the hardness of real-world problems with respect to algorithmic efficiency and learning to cope with it.

Section - I
Answer all Questions (4 × 10 Marks)

01. Given an array $A[1...n]$ of n integers and two integers p and q belonging to A, design a pseudocode to compute the indices i, j with $1 \leq i, j \leq n$, where p and q can be placed respectively such that p and q do not get deleted by any $k-$tour. A $k-$tour is a tour of k, where k starts at any index value between 1 and n(including both 1 and n) and hops by 3 places to the right visiting every third place in the array A. When $k+3>n$, then the hop happens circularly. That is suppose $k=n-2$, then $k+3=n+1>n$. In this case k now circularly becomes 1. Likewise if $k=n$, then $k+3=n+3=3$. Effectively k here moves around the array indices circularly hopping by 3 at each move. In this process, the variable k hops around the array A, and for every hop, the element at the index equal to the current value of k is deleted from A. This tour of k will stop as soon as three or lesser elements remain in the array A. If the task cannot be accomplished your algorithm should output '0'.
[Rubrics: Logic: 2 marks, Illustration: 2 marks, algorithm: 4, Time-complexity : 2 marks]

[10] (CO2/K3)

02. Let $G(n \times n)$ be a square grid of n columns and n rows where n is an even positive integer. Given n queens $Q_1, Q_2, \ldots Q_n$, design an algorithm to place the n queens on G so that the following conditions are satisfied:

a. No two queens are in the same row, column, Primary-diagonal(running from top-left to bottom-right) or Secondary-diagonal(running from top-right to bottom-left).
b. All queens Q_i with i being odd are placed in the top half of the grid starting at $G[1][1]$ up to $G[n/2][n/2]$.
c. All queens Q_i with i being even are placed in the bottom half of the grid starting at $G[n/2+1][n/2+1]$ up to $G[n][n]$.
d. The queens arrive one by one as input in a sequential way and are not necessarily arriving in order of their indices. For e.g. Q_j can arrive before $Q_i, i<j$. As an example for $n=6$, the queens can arrive in the sequence $Q_2, Q_4, Q_5, Q_1, Q_3, Q_6$. Many such sequences exist.

[Rubrics: Logic: 2 marks, Illustration: 2 marks, algorithm: 4, Time-complexity : 2 marks]

[10] (CO5/K4)

03. Let $G = (V, E)$ be a network with $|V| = n$ vertices and $|E| = m$ edges. With each edge we associate a capacity function $c : (V \times V) \rightarrow Z$. Designate a vertex $s \in V$ as the source vertex and another vertex $t \in V$ as the sink vertex. A path in a graph G is a finite sequence of alternating vertices and edges, starting from one vertex and ending at another vertex. Two paths P_1 and P_2 are said to be disjoint paths if they do not have any vertex in common. An augmenting path from source s to sink t is a path P from s to t such that every edge on P has a positive capacity. Design an algorithm to compute the maximum flow from s to t by only considering augmenting paths that are disjoint from earlier augmenting paths from s to t. Here the source s and the sink t are exemptions to the disjoint criteria.

[Rubrics: Logic: 2 marks, Illustration: 2 marks, algorithm: 4, Time-complexity : 2 marks]

[10] (CO3/K3)

04. a. Given a set of n intervals on a real line, represented as $I = \{[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n]\}$, an interval $[a_i, b_i]$ is said to be contained in another interval $[a_j, b_j]$ if $a_j \leq a_i$ and $b_i \leq b_j$. The problem is to count the number of intervals that are contained within at least one other interval.
Compute the class-complexity(P/NP/NPC) of the Problem with justification. [5 marks]

b. Let $n = a_1a_2a_3 \ldots a_k$ be a k-digit positive integer. For any two digits a_i and $a_j, 1 \leq i, j \leq k$, the **digit-distance** between a_i and a_j is defined as $(|a_i - a_j|)$ modulo q, where q is a prime number. The **Distance Shuffle Problem** is described as follows: Given n and q as input, the task is to re-shuffle the digits of $n, < a'_1a'_2a'_3 \ldots a'_n >$ so that sum of all digit-distances between consecutive digits is minimum. For example, if $n = 56439$ with $q = 3$, then, for the digit-shuffle of n is 34569 the total sum of all consecutive digit-distances is 3. For the digit-shuffle 45693, the sum of all consecutive digit-distances is 2 and is the minimum. Many such digit-shuffles exist for which the total sum of all consecutive digit-distances is minimum and so the answer is not unique. **Compute the complexity class(P / NP / NP-C) of the Distance Shuffle Problem with proper justification. [5 marks]**
[Rubrics: Classification: 2+2 marks, Justification: 3+3 marks]

[10] (CO5/K4)

Section - II
Answer all Questions (4 × 15 Marks)

05. Describe a problem called '**k-Restricted Common Subsequence(kRCS)} Problem**' as follows : You are given two strings: $S_1 = s_1s_2s_3 \ldots s_m$, $S_2 = t_1t_2t_3 \ldots t_n$ and a positive integer k. A 'k-Restricted Common Subsequence (**kRCS**) between S_1 and S_2, is a common subsequence(CS) of S_1 and S_2 that satisfies the following constraint:
For any two consecutive characters a and b in the CS of S_1 and S_2, let i represent the index position of a in S_1 and j represent the index position of b in S_2. Then
$|i - j| \leq k$ for all consecutive characters a and b in the CS.

For example, if S_1= "**AXBYCZ**", S_2= "**ABCXYZ**", $k = 2$,, then "**ABCZ**" is a CS of S_1 and S_2. Here we can see that "**ABCZ**" is not a **kRCS**. For the common subsequence "**ABCZ**", the index positions of C and Z in S_1 are 4 and 5 respectively so that $|4 - 5| = 1 < k$. However, the index positions of C and Z in S_2 are 2 and 5 respectively so that $|2 - 5| = 3 > k$. On the other hand, if we consider the common subsequence "**ABC**" we can easily see that it is a **kRCS**. You are given as input two strings S_1 and S_2, a gap constraint k. Design an algorithm based on the dynamic programming strategy to compute, the longest **kRCS** of S_1 and S_2. Your algorithm should return the length of the longest **kRCS** as well as the sequence. It should return -1 if no such sequence is possible. Your solution should include all the design components. For the purpose of the illustration, you should not take the example input provided in the question and you have to illustrate by taking sample strings with at least 6 characters where **kRCS** is possible.

[**Rubrics**: Logic for pseudocode: 2 marks, Recurrence Relation: 2, Illustration for pseudocode: dp table (2 marks) , constraint validation (2 marks), Output extraction(1 mark) , Pseudocode : 4 marks, Time-complexity with the reasons :2 mark]

[15] (CO3/K3)

06. Let $N = a_1a_2 \dots a_n$ be an n digit number and let a_i, a_j, a_k, a_l be any four digits of N where i, j, k and l are the positions of the digits a_i, a_j, a_k, a_l in N respectively.
A **shuffle operation** on a_i, a_j, a_k, a_l, denoted as Shuffle(a_i, a_j, a_k, a_l), is defined as follows.
a. The digit at position i moves to position j,
b. The digit at position j moves to position k,
c. The digit at position k moves to position l,
d. The digit at position l moves to position i.

For example, applying Shuffle$(1, 5, 7, 8)$ on the number 12857 results in 82715. Each such Shuffle of four digits generates a new number.
Given an n-digit number N, design two different algorithms using two distinct techniques to identify the quartet (i, j, k, l) such that the number M, obtained by performing Shuffle(a_i, a_j, a_k, a_l) on N, is the **maximum possible** among all such shuffles. Your algorithm must:
a. Clearly describe both algorithms and the techniques they use,
b. Provide a clear illustration of both the algorithms,
c. Analyze the **runtime** and **time complexity** of both the algorithms.

[Rubrics: Logic: 2+2 marks, Illustration: 1+1 marks, algorithm: 3 + 3 marks, Time-complexity: 2+1 marks]
[15] (CO1/K2)

07. Define a function $f(S)$, that performs a left cyclic shift on an array S of length m. It transforms S into a new array S' such that:
$f(S[1, 2, \dots, m]) = S'[1, 2, \dots, m]$ where:
- $S'[i] = S[i + 1]$ for $1 \leq i < m$
- $S'[m] = S[1]$

In other words, each element in S is moved one position to the left, and the first element wraps around to the last position.
For example: $f(abcd) = bcda$, $f^2(abcd) = f(f(abcd)) = f(bcda) = cdab$, $f^3(abcd) = f(f(f(abcd))) = dabc$.
Text T is an array $T[1, 2, 3.., n]$ of length n and the pattern P is an array $P[1, 2, \dots m]$ of length m. We define an operation: $Relocate(T, P) = s$, if $T[s + 1, s + 2, \dots, s + m] = P[1, 2, \dots, m]$ where $0 \leq s \leq m$ and $Relocate(T, P) = -1$ if pattern P does not occur in text T. $Relocate(T, P)$ may return one integer or a sequence of integers. **Given a Text T and a Pattern P, design an algorithm to compute $Relocate(f^k(T), f^k(P))$, where $k = 0, 1, 2, 3, \dots m$.** Note that $f^0(T) = T$ and $f^0(P) = P$. For example, if $T = abcbcbcacabdaacbabc$ and $P = abc$, then:
$Relocate(f^0(T), f^0(P)) = 0, 16$
$Relocate(f^1(T), f^1(P)) = 4, 16$
$Relocate(f^2(T), f^2(P)) = 6, 16$
Analyze your algorithm with the running-time and the time-complexity.
[Rubrics: Logic: 4 marks, Illustration: 4 marks, algorithm: 5 marks, Time-complexity: 2 mark]
[15] (CO2/K3)

08. The convex hull of a set X of points is the smallest convex polygon P for which each point in X is either on the boundary of P or in its interior. Let P and Q be two convex hulls in a two-dimensional plane covering the set X and the set Y respectively, where $P = \{p_1, p_2, \dots, p_m\}$ and $Q = \{q_1, q_2, \dots, q_n\}$. Here, p_i, $1 \leq i \leq m$ are the vertices of the convex hull P and q_i, $1 \leq i \leq n$ are the vertices of the convex hull Q. Two convex hulls P and Q are said to intersect if, any two edges of P and Q intersect. Given two convex hulls P and Q which intersect, and a point x on the 2D plane, write an algorithm to check if the point x lies in both the convex hulls of P and Q. Analyze your algorithm with the running-time and the time-complexity.

[**Rubrics**: Logic: 4 marks, Illustration: 3 marks, algorithm: 6 marks, Time-complexity: 2 mark]
[15] (CO2/K3)

BL-Bloom's Taxonomy Levels - (K1-Remembering,K2-Understanding,K3-Applying,K4-Analysing,K5-Evaluating,K6-Creating)