VIT
Vellore Institute of Technology

Final Assessment Test (FAT) – May 2022

Programme	B.Tech	Semester	Winter Semester 2021-22
Course Title	DESIGN AND ANALYSIS OF ALGORITHMS	Course Code	CSE2012
Faculty Name	Prof. S Venkatraman	Slot	C2+TC2
		Class Nbr	CH2021225000708
Time	3 Hours	Max. Marks	100

Answer all the questions.

- If any assumptions are required, assume the same and mention that assumption in the answer script.
- Use of intelligence is highly appreciated.
- Every question has 'design' and 'analysis' component.
- Design component of your answer should include : logic, pseudocode and an illustration on the functionality of the pseudocode.
- Analysis component of your answer should include: computation of the running time and the time complexity.
- Every question (other than question nos:5,9) carries 10 marks. Marks for the different components are : description of logic/technique(2 marks), pseudocode(3 marks), illustration (3 marks), running time (1 mark) and time-complexity(1 mark)
- Marks for the different components of question no.5(a), 5(b) : Computation of the complexity class(2 marks), Justification(3 marks)
- Marks for the different components of question no.9 : Problem formulation(2 marks), pseudocode(3 marks), illustration(3 marks), running-time(1 mark), time-complexity (1 mark)

Section-1 (10 X 10 Marks)

Answer All questions

1. A Swap operation of three numbers a_i, a_j, a_k in an n-digit number $N = a_1a_2...a_n$, written as $Swap(a_i, a_j, a_k)$ is defined as follows. [10]

   - a_i takes the position of a_j in N.
   - a_j takes the position of a_k in N.
   - a_k takes the position of a_i in N.

   $Swap(1, 5, 7)$ on 1257 gives a new number 7215. Every swap operation of three numbers on N gives a new number. Given a number N, design an algorithm to compute the triplet a_i, a_j, a_k such that number M obtained from N through the $Swap(a_i, a_j, a_k)$ is the maximum among all the numbers obtainable from N through the swap of three numbers. Analyse your algorithm with the running-time and the time-complexity.

2. Given an array A of n integers, design an algorithm to arrange the elements of N in such a way that $A[i] \leq A[i+1]$ and $|A[i] - A[i+1]|$ is always less than k, for any $1 \leq i \leq n-1$. In other words, the successive elements should be in an increasing order and the difference between the successive elements should be less than k. If any element of A could not be arranged satisfying the above specified conditions, your algorithm should delete those numbers from the array. For example, given the array $< 1, 7, 2, 3, 9 >$ and $k = 2$, your algorithm should output $< 1, 2, 3 >$. Analyse your algorithm with the running-time and the time-complexity. [10]

3. 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 $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$. Here the points p_i, $1 \leq i \leq m$ are the vertices of the convex hull P. Similarly, points q_i, $1 \leq i \leq n$ are the vertices of the convex hull Q. Two Polygons are said to intersect if there is at least one point (either boundary point or interior point) common between the two polygons. Given the points $p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_n$, design an algorithm to check whether P and Q intersect or not?. Here, n and m are any two positive integers. Analyse your algorithm with the running-time and the time-complexity. [10]

4. First-to-last-cycle is a function that operates on an array S, written as $f(S)$, described as follows: $f(S[1,2...,m]) = S'[1,2...,m]$ where $S'[i] = S[i+1]$, where $0 < i < m$, $S'[m] = S[1]$. 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 $P[1,2,...m]$ of length m. We define, $Shift(T,P) = s$, if $T[s+1, s+2, ..., s+m] = P[1,2,...,m]$ where $0 \leq s \leq m$ and $Shift(T,P) = -1$ if pattern P does not occur in text T. $Shift(T,P)$ may return one integer or a sequence of integers. Given a Text T and a Pattern P, design an algorithm to compute $Shift(T, f^k(P))$, where $k = 0,1,2,3,...m$. Note that $f^0(P) = P$. For example, if $T = abcbcbcacabdaacbabc$ and $P = abc$ then $Shift(T, f^0(P)) = 0, 16$, $Shift(T, f(P)) = 5$, $Shift(T, f^2(P)) = 8$. Analyse your algorithm with the running-time and the time-complexity. [10]

5. (a) A word $a_1a_2a_3, ..., a_n$ is said to be a palindrome if $a_1 = a_n$, $a_2 = a_{n-1}$ and so on. The word 'madam' is a palindrome and the 'cat' is not a palindrome. Given a word w, the task of the 'palindrome-check problem' is to decide whether the given word w is a palindrome or not. With proper justification, Compute the complexity class (P class or NP class or NP complete class) of 'palindrome-check problem'. Also check whether the problem falls into more than one complexity class and if so, justify your answer. [10]

(b) Let A be an array which contains n numbers. The task of the 'frequency-count problem' is to identify the numbers which occur one time, numbers which occur two times, numbers which occur three times and so on and the numbers which occur n times. With proper justification, Compute the complexity class of 'frequency-count problem'. Also check whether the problem falls into more than one complexity class and if so, justify your answer.

6. Given a flow network $G = (V, E, c, s, t)$ where V is the set of vertices ; E is the set of edges connecting two vertices of V, c is the set of capacities, s is in V designated as source vertex and t is in V designated as target vertex. Every edge $(u, v) \in E$ has a positive capacity. An edge e in G is upper-binding if increasing e's capacity by 1 increases the value of the maximum flow in G. Similarly, an edge is lower-binding if decreasing its capacity by 1 decreases the value of the maximum flow in G. Given a flow network G with the capacities, design an algorithm to identify all the upper binding edges and all the lower binding edges of G. Analyse your algorithm with the running-time and the time-complexity. [10]

7. Given a graph $G = (V, E)$, V is the set of vertices and E is the set of edges. A simple path in a graph is a path without repeated vertices. Let any two vertices in V be designated as Source S and Terminal T. A path, say, $S - v1 - v2 - T$ is said to be a path with 2 vertices. That is, for the calculation of the number of vertices in a path, we exclude the source and the destination of the path. Given the graph G, vertex S, vertex T and an integer k, design an algorithm to check whether there exist a simple path from S to T with at least k vertices and return the sequence of vertices that form the simple path with at least k vertices. Analyse your algorithm with the running-time and the time-complexity. [10]

8. An $n \times n$ grid has n^2 cells. A diagonal of the $n \times n$ grid that starts from the top-most right corner (right corner of the grid is the one which is to your right when you face the grid) of the grid and ends at the left-most bottom corner of the grid is called as anti-diagonal of the grid. Given an $n \times n$ power grid, the problem is to place n power stations in n cells in such a way that the following conditions are staisfied. (i). Any column of the grid cannot have more than one power station. (ii). Any row of the grid cannot have more than one power station. (iii). Anti-diagonal cannot have more than one power station. [10]

Given n, design an algorithm that will return the position of the cells where n power stations can be placed in the given grid. Analyse your algorithm with the running-time and the time-complexity.

9. Propose a problem P (of your choice) in detail which can be solved by a dynamic programming based algorithm, with justification. You are not supposed to describe any problem described in this question paper. Design an algorithm for the problem chosen by you. Analyse your algorithm with the running-time and the time-complexity. [10]

10. You are organizing a function for which you have invited n guests. Every guest will be picked up from their house and dropped at the venue of the function. Every car will start from the venue of the function to pick up the guest and drop the guest at the venue. Every guest g_i will have a pair (s_i, d_i), where s_i is the start-time of the car from the venue to pick the guest and the d_i is the drop-time of the guest g_i at the venue. Given the details (s_i, d_i) of the guests g_i, $i = 1,2,...n$, design an algorithm to compute the minimum number of cars to be booked for the purpose. For example, If $g_1 : (8:15, 9:05)$, $g_2 : (8:40, 9:25)$, $g_3 : (9:10, 9:45)$, $g_4 : (9:47, 10:50)$, $g_5 : (9:30, 10:20)$ then minimum of two cars are required. Analyse your algorithm with the running-time and the time-complexity. [10]