|  |  |
| --- | --- |
| **SESSION** | **APRIL 2025** |
| **PROGRAM** | **MASTER OF COMPUTER APPLICATIONS (MCA)** |
| **SEMESTER** | **III** |
| **COURSE CODE & NAME** | **DCA7104 ANALYSIS & DESIGN OF ALGORITHM** |
|  |  |
|  |  |

**Set-I**

**Q1. Define the term 'algorithm'. Discuss different algorithm design paradigms with a focus on their advantages and limitations. Include examples.**

**Ans 1.**

**Algorithm**

An algorithm is a well-defined set of step-by-step instructions or a computational procedure used to perform a task or solve a particular problem. In computer science, algorithms serve as the fundamental logic behind any program. They must be finite, effective, and unambiguous, taking a set of inputs and producing a desired output after a finite number of steps.

**Algorithm Design Paradigms**

Different algorithm design paradigms are systematic approaches used to develop efficient algorithms. The choice of paradigm can affect the performance and complexity of an algorithm.

Its Half solved only

Buy Complete assignment from us

**Price – 190/ assignment**

**MUJ Manipal University Complete SolvedAssignments MARCH 2025**

buy cheap assignment help online from us easily

we are here to help you with the best and cheap help

**Contact No – 8791514139 (WhatsApp)**

**OR**

**Mail us-** [bestassignment247@gmail.com](mailto:bestassignment247@gmail.com)

**Our website -** [www.assignmentsupport.in](http://www.assignmentsupport.in)

**Q2a) Derive the time complexity for Merge Sort algorithm using recurrence relation.**

**b) Differentiate between best, average, and worst-case complexities with examples.**

### **Ans 2.**

### **a) Time Complexity of Merge Sort Using Recurrence Relation**

Merge Sort is a classic Divide and Conquer algorithm that works by recursively dividing an array into halves, sorting each half, and merging the sorted halves back together. The performance of Merge Sort can be analyzed using recurrence relations.

Let represent the time complexity of Merge Sort for an array of size . Merge Sort divides

**Q3.a) Trace Quick Sort on the array: [23, 4, 42, 15, 16, 8]. Show all steps.**

**b) Explain recursive and non-recursive algorithm analysis with suitable examples.**

**Ans 3.**

### **a) Tracing Quick Sort on the Array [23, 4, 42, 15, 16, 8]**

Quick Sort is a Divide and Conquer sorting algorithm that selects a pivot element and partitions the array such that elements less than the pivot come before it, and elements greater come after it. The same process is then recursively applied to the sub-arrays.

Quick Sort on the given array: [23, 4, 42, 15, 16, 8]. Assume the last element is taken as the pivot.

**Step 1**: Pivot = 8 Initial array = [23, 4, 42, 15, 16, 8] We rearrange elements so that all smaller

**Set-2**

**Q4. Explain Divide and Conquer strategy. Implement and analyze the performance of Quick Sort. Explain with time complexity derivation.**

**Ans 4.**

### **Divide and Conquer Strategy**

Divide and Conquer is a fundamental algorithm design strategy where a problem is divided into smaller sub-problems of the same type, solved recursively, and their results combined to produce the final output. This strategy helps simplify complex problems by breaking them into manageable parts. It is widely used in algorithms like Merge Sort, Quick Sort, and Binary Search.

### Working of Divide and Conquer

The Divide and Conquer approach follows three main steps:

* **Divide**: Break the problem into smaller sub-problems.

**Q5. Apply Dynamic Programming to solve the Optimal Binary Search Tree (OBST) problem. Show step-by-step computation.**

**Ans 5.**

### **Optimal Binary Search Tree (OBST)**

An Optimal Binary Search Tree is a binary search tree where the total cost of all searches is minimized, assuming that each key has a known probability of being searched. The tree is “optimal” in the sense that frequently accessed nodes appear near the root, thus minimizing the average search time.

This problem is a classic application of Dynamic Programming (DP) because it involves

**Q6a. Describe P, NP, and NP-Complete problems. Why is the distinction important?**

**b) Illustrate Branch and Bound method with a Travelling Salesman Problem example.**

### **Ans 6.**

### **a) Classification of Problems: P, NP, and NP-Complete**

In computational complexity theory, problems are classified based on the time required to solve them using a deterministic Turing machine. The three major classes are P, NP, and NP-Complete.

**Class P (Polynomial Time)**

Problems in class P can be solved by a deterministic algorithm in polynomial time, i.e., the time taken to solve the problem increases at a polynomial rate with input size. Examples include