CMR Institute of Technology, Bangalore		
Department(s): Information Science & Engineering		
Semester: 4	Section(s): A/B/C	Lectures/week: 04
Subject: Analysis & Design of Algorithm		Code: BCS401
Course Instructor(s): Prof. Dinesh Kumaar/Prof. Kavitha		
Course duration: Feb 2026 – May 2026		
Google Classroom Link: https://classroom.google.com/c/ODQzNzM0ODg1MzA0?cjc=fmvezgz7		

IAT-1 Question Bank

1.) Define an algorithm. Discuss the characteristics of an algorithm.
2.) Write an algorithm to find the maximum element in an array of n elements. Give the mathematical analysis of this non recursive algorithm.
3.) Explain the general plan for analyzing the efficiency of a recursive algorithm. Write the algorithm to find a factorial of a given number. Derive its efficiency.
4.) Apply a quick sort algorithm to sort the list E, X, A, M, P, L, E in alphabetical order. Draw the tree of recursive calls made.
5.) List out the advantages and disadvantages of divide and conquer approach.
6.) Write the control abstraction for divide and conquer technique.
7.) Give the general plan for analyzing time efficiency of Recursive algorithms and also analyze the Tower of hanoi Recursive algorithm.
8.) Apply both merge & quick sort algorithms to sort the characters VTUBELAGAVI
9.) Consider the following algorithm.

```
ALGORITHM Mystery(n)
//Input: A nonnegative integer n
S ← 0
for i ← 1 to n do
S ← S + i * i
return S
```

a. What does this algorithm compute?
b. What is its basic operation?

c. How many times is the basic operation executed?
d. What is the efficiency class of this algorithm?
e. Suggest an improvement, or a better algorithm altogether, and indicate its efficiency class. If you cannot do it, try to prove that, in fact, it cannot be done.

10.) Sort the list E, X, A, M, P , L, E in alphabetical order by selection sort.
11.) Sort the following element using Merge Sort 10,5,7,6,1,7,8,3,2,9
12.) Explain in brief the basic asymptotic efficiency classes
13.) Compare the order of growth of log (n) & $\sqrt{n}$.
14.) Discuss how quicksort works to sort an array & trace the following data set. 1,1,9,9,5,5,6,6.

ANSWERS

What Is an Algorithm?
An algorithm is a finite, well-defined sequence of steps designed to perform a specific task or solve a particular problem. It takes an input, processes it through a series of operations, and produces an output. Algorithms are fundamental to computer science and are the backbone of software development.

🧠 Characteristics of an Algorithm
An effective algorithm typically possesses the following characteristics:

Finiteness: It must terminate after a finite number of steps.

Definiteness: Each step must be precisely defined.

Input: It should accept zero or more inputs.

Output: It should produce at least one output.

Effectiveness: All operations must be basic enough to be carried out, in principle, by a human using paper and pencil.

🔍 Algorithm to Find the Maximum Element in an Array

Algorithm: Find Maximum

plaintext

Copy

Edit

1. Initialize `max` to the first element of the array.
2. Iterate through the array starting from the second element:
   a. If the current element is greater than `max`, update `max`.
3. Return `max` as the maximum element.

Mathematical Analysis

Time Complexity: O(n), where n is the number of elements in the array. This is because each element is compared exactly once.

Space Complexity: O(1), as only a single variable (max) is used.

🔁 Efficiency Analysis of Recursive Algorithms

When analyzing the efficiency of recursive algorithms, consider the following general plan:

Identify the Base Case: Determine the simplest instance of the problem that can be solved directly.

Divide the Problem: Break the problem into smaller subproblems.

Conquer: Solve the subproblems recursively.

Combine: Merge the solutions of the subproblems to solve the original problem.

The efficiency is often analyzed using recurrence relations and solving them using methods like the master theorem.

📐 Example: Factorial Algorithm (Recursive)
Algorithm: Compute Factorial
plaintext
Copy
Edit
1. If n = 0, return 1.
2. Otherwise, return n * factorial(n - 1).
Efficiency Analysis
Time Complexity: O(n), as the function calls itself n times.

Space Complexity: O(n), due to the depth of the recursion stack.

🔁 Quick Sort Algorithm and Recursive Call Tree
Quick Sort is a divide-and-conquer algorithm that works as follows:

Divide: Choose a pivot element.

Partition: Rearrange the array so that elements less than the pivot come before it and elements greater come after it.

Conquer: Recursively apply the same process to the subarrays.

Example: Sorting "E X A M P L E"
Choose a pivot (e.g., 'E').

Partition the array: ['A', 'E', 'L', 'M', 'P', 'X'].

Recursively sort the subarrays ['A'] and ['L', 'M', 'P', 'X'].

Recursive Call Tree
scss

Copy
Edit
quicksort(E X A M P L E)
├── quicksort(A)
└── quicksort(L M P X)
    ├── quicksort(L)
    └── quicksort(M P X)
        ├── quicksort(M)
        └── quicksort(P X)
            ├── quicksort(P)
            └── quicksort(X)

⚖️ Advantages and Disadvantages of Divide and Conquer

Advantages

Efficiency: Often leads to algorithms with optimal time complexities.

Parallelism: Subproblems can be solved concurrently.

Simplicity: Breaks down complex problems into simpler subproblems.

Disadvantages

Overhead: Recursive calls can lead to significant overhead.

Memory Usage: Requires additional memory for recursion stacks.

Not Always Applicable: Not suitable for problems that don't naturally divide into subproblems.

🧩 Control Abstraction for Divide and Conquer

The general control abstraction for divide-and-conquer algorithms involves:

Divide: Split the problem into subproblems.

Conquer: Solve each subproblem recursively.

Combine: Merge the solutions of the subproblems to solve the original problem.

🏰 Analyzing the Tower of Hanoi Recursive Algorithm
The Tower of Hanoi problem involves moving a stack of disks from one peg to another, subject to certain rules. The recursive algorithm works as follows:

Move n-1 disks from source peg to auxiliary peg.

Move the nth disk to the destination peg.

Move the n-1 disks from auxiliary peg to destination peg.

Efficiency Analysis
Time Complexity: O(2^n) due to the recurrence relation T(n) = 2T(n-1) + 1.

Space Complexity: O(n), due to the depth of the recursion stack.

🔄 Sorting Algorithms: Merge Sort and Quick Sort
Merge Sort
Divide: Split the array into two halves.

Conquer: Recursively sort each half.

Combine: Merge the two sorted halves.

Time Complexity: O(n log n) in all cases.

Space Complexity: O(n), due to the auxiliary arrays used for merging.

Quick Sort
Divide: Choose a pivot element.

Partition: Rearrange the array around the pivot.

Conquer: Recursively sort the subarrays.

Time Complexity: O(n log n) on average; O(n^2) in the worst case.

Space Complexity: O(log n) on average; O(n) in the worst case.

📊 Basic Asymptotic Efficiency Classes
O(1): Constant time.

O(log n): Logarithmic time.

O(n): Linear time.

O(n log n): Linearithmic time.

O(n^2): Quadratic time.

O(2^n): Exponential time.

🔍 Comparing log(n) and n
log(n) grows much slower than n. For large values of n, log(n) is significantly smaller than n.

log(n) is often seen in algorithms that divide the problem size in each step (e.g., binary search).

🧭 Quick Sort on Data Set: [1, 1, 9, 9, 5, 5, 6, 6]
Choose a pivot (e.g., '5').

Partition the array: [1, 1, 5, 5, 6, 6, 9, 9].

Recursively sort the subarrays [1, 1], [5, 5], [6, 6], and [9, 9].