Date: / /
Page No.

Name:- Parmar Kulpesh P.

Roll No:- 92400584194

Subject:- D.S.A.

Class:- MCA-1C

Q-1. Describe categories of Data structure.

In Data structure and algorithm, data structure are typically categorized based on their characteristics:

- Primitive Data Structure
- Non-Primitive Data Structure

1) Primitive Data Structure:-

These are the Basic building blocks for data manipulation They include:
(i) Integer (ii) Flots (iii) Characters (iv) Boolean.

2) Non-Primitive Data Structure:-

These structure are more complex and can be classified into:

a) Linear Data Structure:-

In these structure, elements are arrenged in a Sequential manner. Each element is connected to its previous and next element, forming a linear Sequence.

- Arrays:- Fixed-size collections of ~~connected nodes.~~ elements of the same type.

- Linked Lists:- A series of connected nodes, where each node contains data and a refrence to the next node.

  - Singly Linked List
  - Doubly Linked List
  - Circular Linked List

- Stacks:- Follow Last In First Out (LIFO) principle. Where the last element added is the first to be removed.

- Queues:- Follow First In First Out (FIFO) principle. Where the first element added in the First to be removed.

b) Non-Linear Data Structures:-

In these structure, elements are not arrenged sequentially and can have multiple connection.

- Tree:- Hierarchical Structure with nodes connected by edges.

  - Binary Trees:- Each node has at most two Children.

    - Binary Search Tree:- A binary tree with ordered nodes.

- Graphs:- A set of Vertices connected by edges. can be directed or undirected.

  - Weighted Graphs:- Edge have weights.

  - Unweighted Graphs:- No weights on edges.

Q-2. Differentiate top down and bottom up approach of algorithm.

1) Top-Down Approch:-

- Defination:- The problem is solved by breaking it down into smaller subproblems recursively. The solution to the larger problem is constructed by combining solution to the smaller problems.

- Implementation:- Typically uses recursion. Function call themselves to solve subprogram.

- Example:- Recursive algorithms for problems like Fibonacci sequence calculation or solving problems using dynamic programming.

- Advantages:- Easier to implement and understend due to a natural recursive structure.

- Disadvantage:- May have higher overhead due to recursive function calls.

2) Bottom-up Approch:-

- Defination:- start with the smallest subproblems and combiness their solution to build up to the solution of the larger problem.

- Implementation:- often uses iteration instead of recursion, filling in a table or array with Solution to subproblem.

- Example:- Dynamic programming Solutions like the Fibonacci sequence using an iterative approch or filling a DP table for problems like the Longest common subsequence.

- Advantages:- Generally more efficient in terms of space and time, as it avoids the overhead of recursive calls.

- Disadvantages:- May be less intuitive for problems that are inherently recursive.

Q-3 Describe complexity and its types.

In data structure and algorithms complexity refers to the resources required by an algorithm to solve a problem, primarily focusing on time and space understanding complexity helps in evaluating the efficeiency of algorithm.

Types of Complexity:-

- Time Complexity
- Space Complexity

1) Time Complexity:-

Measures the time an algorithm takes to Complete as a function of the size of the input.

- Common notations:-

Big O Notation:- Upper bound
Omega Notation:- Lower bound
Theta Notation:- Tight bound

· Examples:-

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

2) Space Complexity:-

Measures the amount of memory an algorithm uses in relation to the input size.

Similar to time complexity, it can be analyzed Using O Notation.

- Types:-

  (i) Fixed Part:- Space required for constants, Simple variable, Fixed-size variable, etc.

  (ii) Variable Part:- Space required for dynamic structures that depend on the input size.

- Example:- An algorithm that uses an array of size n has a space complexity of O(n).