Design of Data Structures - Unit 1 & 2 Notes

Unit 1: Introduction to Data Structures

What are Data Structures?

A data structure is a way of organizing and storing data so that it can be accessed and worked with efficiently.
Example: Books in a library → Arranged in racks (like arrays) or categorized by subject (like hash tables).

Classification

1. Primitive: int, float, char, boolean (basic types)
2. Non-Primitive:
- Linear: Array, Stack, Queue, Linked List
- Non-Linear: Tree, Graph

Operations on Data Structures

Traversal, Insertion, Deletion, Searching, Sorting

Arrays

Collection of same type elements stored in contiguous memory. Example: Seats in a cinema hall.

Algorithm: Traversing an Array

Algorithm Traverse_Array(A, n)
1. For i ← 0 to n-1 do
2. Print A[i]
End

Structures & Unions

Structure: Groups related data of different types (e.g., Student record).
Union: Memory shared by all members (efficient but only one active at a time).

Pointers & Dynamic Memory

Pointer stores address of variable. Dynamic allocation assigns memory at runtime (malloc, calloc). Example: ATM slip size depends on transaction.

Performance Analysis

Time Complexity (speed), Space Complexity (memory). Big-O: O(1), O(n), O(log n), $O(n^2)$.

Unit 2: Stacks, Recursion and Queues

Stack

Linear structure using LIFO (Last In First Out). Example: Stack of plates.

Stack Operations

PUSH(Stack, item):
1. If TOP = MAX-1 → Overflow
2. Else
TOP ← TOP + 1
Stack[TOP] ← item

POP(Stack):
1. If TOP = -1 → Underflow
2. Else
item ← Stack[TOP]
TOP ← TOP - 1
Return item

Applications of Stack

Used for reversing strings, Undo/Redo in editors, Expression conversion (Infix → Postfix, Postfix evaluation).

Recursion

Function that calls itself. Examples: Factorial, Fibonacci, Tower of Hanoi.

Factorial (Recursion)

FACT(n):
1. If n = 0 return 1
2. Else return n * FACT(n-1)

Fibonacci (Recursion)

FIB(n):
1. If n = 0 return 0
2. If n = 1 return 1
3. Else return FIB(n-1) + FIB(n-2)

Tower of Hanoi

TOH(n, source, temp, dest):
1. If n = 1 → Move disk source → dest
2. Else
TOH(n-1, source, dest, temp)
Move disk source → dest
TOH(n-1, temp, source, dest)

Queue

Linear structure using FIFO (First In First Out). Example: People waiting in line.

Queue Operations

ENQUEUE(Queue, item):
1. If REAR = MAX-1 → Overflow
2. Else
REAR ← REAR + 1
Queue[REAR] ← item

DEQUEUE(Queue):
1. If FRONT > REAR → Underflow
2. Else
item ← Queue[FRONT]
FRONT ← FRONT + 1
Return item

Circular Queue

Solves wasted space problem. Example: Round-robin CPU scheduling.

Deque (Double Ended Queue)

Insert/Delete allowed at both ends. Example: Train compartment doors.

Priority Queue

Each element has priority. Highest priority served first. Example: Emergency room in hospital.