ADT - Abstract Data Type

- Data structure
  - How data is organized in memory?
  - How operations are performed on that data?
- From user perspective, all data structures are ADTs.

Array ADT

- Operations
  - Write element on given index: arr[i] = x
  - Read element from given index: y = arr[i]
- Advanced Operations
  - Sorting
  - Searching
  - Traversing

Queue ADT

- FIFO behaviour
- Operations
  - Insert in Queue -- Enqueue() / Push()
  - Delete from Queues -- Dequeue() / Pop()
  - Read next Element -- Peek()
  - isEmpty() -- True/False
  - isFull() -- True/False
- isFull() operation is applicable if storage capcity is fixed

Types of Queue

1. Linear queue
   - data is inserted from rear end and removed from front end
   - follows FIFO principle
   - may lead to poor memory utilization
2. Circular queue
   - data is inserted from rear end and removed from front end
   - follows FIFO principle
   - front and rear are moved circular so that memory will be utilized properly
3. Deque (Double Ended Queue)
   - data can be inserted or removed from both the ends of queue
   - operations : push front, pop front, peek front, push rear, pop rear, peek rear
   - do not follows FIFO principle
   - stack and queue can be implemented using deque
4. Priority Queue
   - every element is associated with a priority
   - alway high priority element is removed from the queue
   - do not follows FIFO principle

- can be implemented with array, linked list but implemented efficiently with heap data structure

Queue Applications

1. CPU Scheduling (Operating Systems)
   - Processes wait in a ready queue
   - The first process to arrive gets executed first (in simple scheduling)
2. Printer Spooling
   - Multiple print jobs are stored in a queue
   - Jobs are printed in the order they are received
3. Waiting Lines (Real Life)
   - Ticket counters, banks, supermarkets
   - First person in line is served first
4. Data Buffers
   - Used in keyboard input, network data packets, IO buffers
   - Data is processed in the order it arrives
5. Breadth-First Search (BFS)
   - Used in graphs and trees
   - Explores nodes level by level
6. Message Queues
   - WhatsApp, email servers, task queues
   - Messages/tasks are delivered in arrival order
7. Web Server Request Handling
   - Incoming client requests are queued
   - Handled one by one or by workers
8. Traffic Systems
   - Cars at toll booths or traffic signals
   - First car to arrive moves first
9. Call Center Systems
   - Incoming calls are placed in a queue
   - Answered in the order received
10. Resource Sharing
    - Shared resources (CPU, printer, disk)
    - Requests are queued to avoid conflicts
11. Multiplayer Games / Online Matchmaking
    - Players waiting to join a game are queued
    - Matched in arrival order

Stack ADT

- LIFO behaviour
- Operations
  - Insert -- Push()
  - Delete -- Pop()
  - Read next -- Peek()
  - isEmpty() -- True/False
  - isFull() -- True/False

- isFull() operation is applicable if storage capcity is fixed.

Stack Applications

1. Function Calls & Recursion
    - The call stack stores function calls, local variables, and return addresses
    - Enables recursion
    - Example: when a function calls another function, it's pushed onto the stack
2. Expression Evaluation
    - Used by compilers and calculators to:
        - Evaluate postfix (Reverse Polish) expressions
        - Convert infix -> postfix / prefix
        - Example: A + B -> AB+
3. Syntax Parsing (Compilers)
    - Checking balanced parentheses, brackets, braces
    - Used in syntax analysis of programming languages
4. Undo / Redo Operations
    - Text editors, graphic tools, IDEs
    - Undo = pop last action
    - Redo = push it back
5. Backtracking Algorithms
    - Used in problems where you explore choices and revert:
    - Maze solving
    - N-Queens problem
    - Sudoku
    - Depth-First Search (DFS)
6. Memory Management
    - Stack memory for:
        - Local variables
        - Function parameters
        - Faster than heap allocation
7. Reversal of Data
    - Reversing:
        - Strings
        - Arrays
        - Linked lists
    - Push elements -> pop in reverse order
8. Web Browser's history
    - Back/Forward navigation
    - Each visited page is pushed onto a stack
9. Operating Systems
    - Interrupt handling
    - Process execution states
    - Context switching
10. Expression & Algorithm Design
    - Depth-First Search (DFS)
    - Topological sorting

- Tree traversals (inorder, preorder, postorder)

Stack Vs Heap

1. OS Interview: What is Stack and Heap?
    - Program in execution is called as "Process".
    - Process is logically divided into multiple sections.
    - Text : Program code (binary)
    - Data : Variable (global)
    - Stack : Function call -- Function Activation Record/Stack Frame
    - Heap : Dynamic memory allocation -- new operator
    - ...
2. DSA Interview: What is Stack and Heap?
    - Stack - Utility data structure that has LIFO behaviour
        - Push and Pop are done from same end (a.k.a. top end)
        - operations: push, pop, peek, ... -- time: O(1)
        - can be implemented using array or linked list
    - Heap - Data structure that is Array implementation of Complete binary Tree
        - Types: Max heap, Min heap