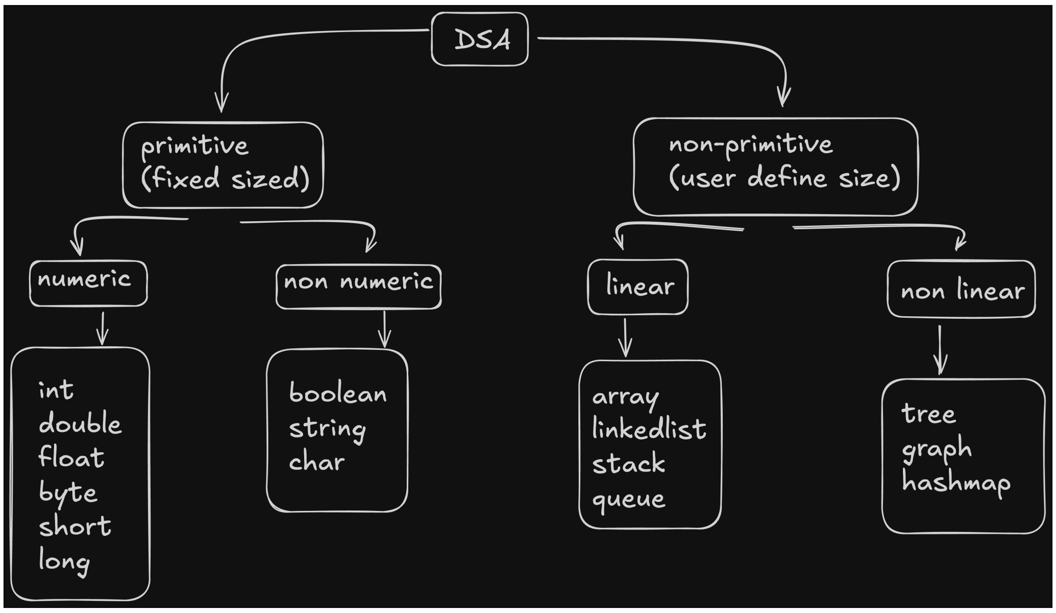
ValueDX data structure and algorithms

DSA:- it is a way of organising, storing and managing data into computer's memory.

Algorithm:- an algorithm is a step by step procedure or set of well define instructions for solving a specific task.



Primitive :- Primitive data types are the basic built-in data types provided by a programming language to represent simple values like integers, characters, and booleans.

Non-Primitive :- Non-primitive data types are user-defined or complex data types that are used to store multiple values or structured data.

Linear:- a linear data type stores elements in a sequential manner, where each element is connected to its previous and next element (except the first and last).

Non linear:- a non-linear data type stores data in a hierarchical or interconnected manner, where one element can be connected to multiple elements.

Time complexity:- it is a concept of measuring time required to execute code with respect to size of input.

Asymptotic Notations:- These are used to describe **how the running time (or space)** of an algorithm **grows** as the **input size (n)** increases.

🔷 **1. Big O Notation – O(n):-Describes the Worst-Case time complexity / upper bound**
🔷 **2. Omega Notation – Ω(n):-Describes the Best-Case time complexity / lower bound**
🔷 **3. Theta Notation – Θ(n):-Describes the Average or Exact Case / tight bound**

✅ **Types of Time Complexity**

1. Constant Time – O(1)
The algorithm takes the **same amount of time** regardless of input size.

2. Logarithmic Time – O(log n)
Time increases **logarithmically** as input size increases. Often seen in **binary search**.

3. Linear Time – O(n)
Time grows **directly proportional** to input size.

5. Quadratic Time – O(n^2)
Time grows with the **square of input size**. Seen in **nested loops** over the same input.

6. Cubic Time – O(n^3)
Time grows with the **cube of input size**. Happens in triple nested loops.

7. Exponential Time – O(2^n)
Time doubles with every additional input. Very inefficient. Seen in **brute-force** recursion (e.g., Fibonacci)

8. Factorial Time – O(n!)
Grows **faster than exponential**. Seen in **permutation-based** problems (e.g., traveling salesman).

Time Complexity	Name	Example Use Case
O(1)	Constant	Accessing array index
O(log n)	Logarithmic	Binary search

$O(n)$	Linear	Simple for loop
$O(n \log n)$	Linearithmic	Merge Sort, Quick Sort
$O(n^2)$	Quadratic	Bubble sort, nested loops
$O(n^3)$	Cubic	3D matrix operations
$O(2^n)$	Exponential	Recursive Fibonacci
$O(n!)$	Factorial	Permutations, n-Queens

Time complexity of every sorting algorithm

Sorting Algorithm	Worst Case Time Complexity
Bubble Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Insertion Sort	$O(n^2)$
Merge Sort	$O(n \log n)$
Quick Sort	$O(n^2)$

Heap Sort	O(n log n)

ADT (Abstract Data Type):- An **Abstract Data Type (ADT)** is a **theoretical concept** for a data structure that defines **what operations** can be performed on the data and **what behaviour** those operations should have.

#Stack

- Non primitive
- Linear
- Follows last in first out principle (LIFO)
- Single opening for insertion and deletion

- The top of a stack refers to the most recently added element that is currently in the stack. Top will remain-1 while stack is empty.
- The peek() operation (sometimes called top()) is used to look at the element on the top of the stack without removing it.

- There are two ways to implement stack one is through array and other is using linked list.

ADT of stack will be :- push, pop, top, peek, isEmpty, isFull

Application of stack :-
1. Undo Feature in Editors
2. Browser History

#QUEUE
- Non primitive
- Linear
- Follows first in first out
- Double opening in front and rear
- There are two pointers rear for insertion and front for deletion. Both are initially -1 while the queue is empty.
- There are multiple types of queue
  1. Simple queue
  2. Priority queue / circular queue :-

rear connects to front so there will be no wastage of empty space
3. Double ended queue :- using both rare/front we can perform insertion and deletion
4. Ascending / descending queue :- element are first kept in sorted then further operation will be performed. (Does not follow fifo principle)
5. peek() points front

ATD of stack will be :- enqueue, dequeue, rare, front, isEmpty, size, peek

Application of queue :-
1. Ticket Booking Systems
2. Traffic single

#LINKEDLIST
- Non primitive
- Linear
- There is a node like structure in which

first half holds the data and second half holds index of next node

- There are two pointer head and null

- There are multiple types of linked list
  1. Singly linked list :- travers only one way
  2. Doubly linked list :- three compartment structure in which first half holds the index of previous node middle half holds data last half holds index of next node. Travers both ways
  3. Circular linked list :- there will be only one pointer head and first and last node are connected in a circular form

ADT of linked list will be :-
insertAtBeginning(data), insertAtEnd(data), insertAt(index, data),

deleteFromBeginning(), deleteFromEnd(), deleteAt(index), isEmpty(), size(),
traverse().

Application of linked list :-

1. Music playlist
2. Use doubly linked list to navigate forward and backward through visited pages.