what is Data Structures ?

→ A data Structure is a way a organising data and Storing data, So that it can be accessed and modified efficiently.

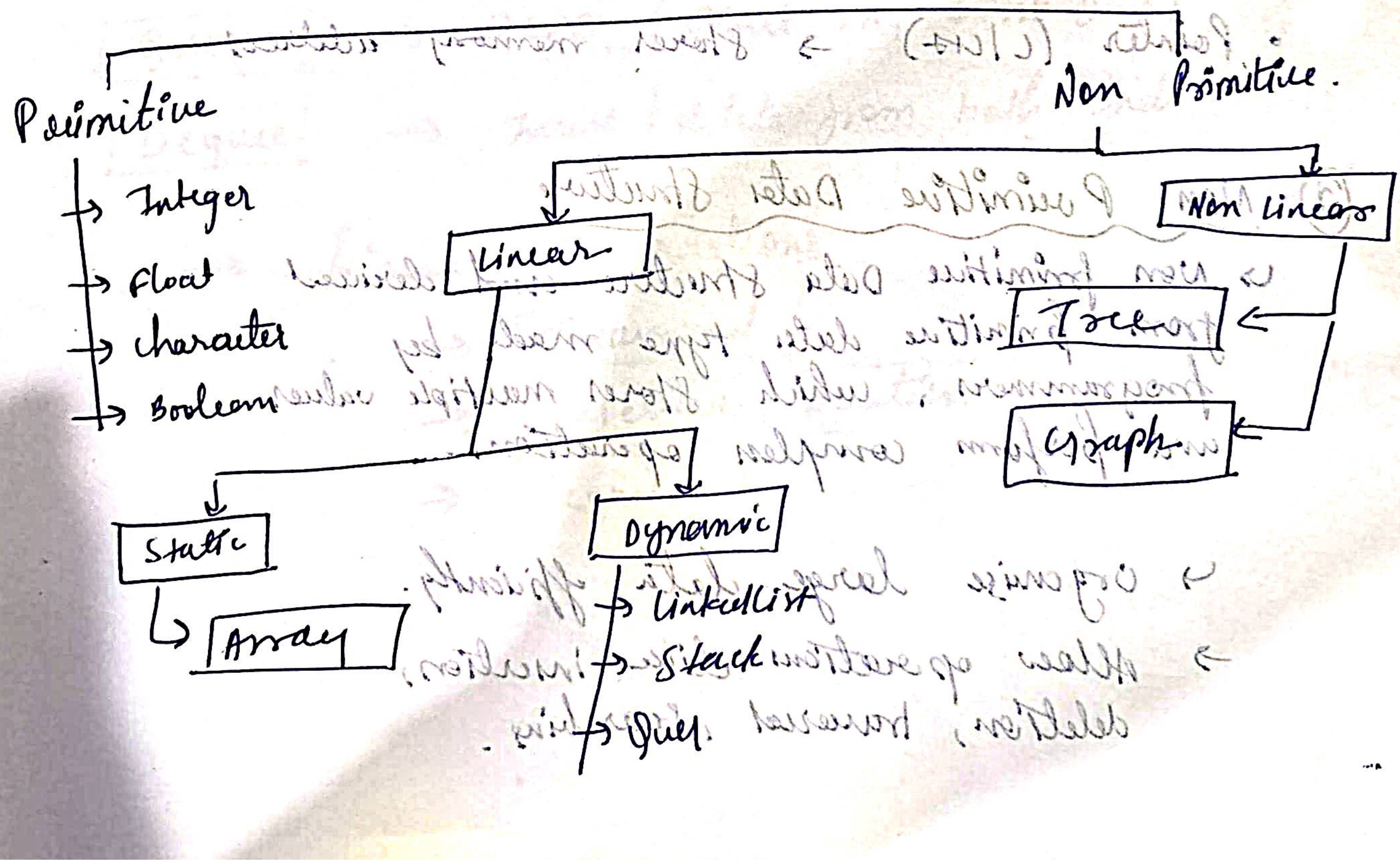
why do we need Data Structure ?

→ Organise data efficiently

→ Perform operations like Search, insert, delete quickly.

→ Optimize memory usage

Types of Data Structure



① Primitive Data Structure ?

→ This are most basic types of data available in programming language. There are directly supported by compilers.

↳ Storing single values.

↳ Fast to use

• Integer	→	Store whole number
• Float	→	Stor real number (decimal)
• String	→	Store english words.
• character	→	Store a single character
• Booleam	→	Store true or false
• Pointer (c/c++)	→	Stores memory address

② Non-Primitive Data Structure

↳ Non primitive Data Structure and derived from primitive data type made by programmers, which stores multiple values and perform complex operations.

↳ Organise large data efficiently.

→ Allow operations like insertion, deletion, traversal, searching.

→ Non Primitive Data Structures are divided into two different types.

① → Linear Data Structure

② → Non Linear Data Structure

~~① Array~~

① Linear Data Structure

→ Elements are arranged sequentiall (one after another)

Array → Fixed-size, same-type elements in contigious memory.

Linked list → Dynamic nodes where each nodes points to the next.

Stack → LIFO (Last In, First Out).

Queue → FIFO (First In, First Out)

Deque → Insert / delete from both ends

→ Linear DS Operations

→ Traversal

→ Insertion / Deletion

→ Search

(B.) Non - Linear Data Structure

→ Elements are arranged hierarchial or with complex relationships. (Not - one - after - another)

- Trees → Hierarchial structure with parent-child nodes.
- Binary Tree → Max two children per node.
- Binary Search Tree → Stored binary Tree
- Heap → Complete binary Tree.
- Trie → Tree for storing String with Prefix sharing.
- Graph → Nodes connected with edges.

| → Non linear DS operations →

- Traversal.
- Search.
- Shortest Path
- Topological Sorting.

(c) Hash Based. Data Structure.

↳ Hash Table / Hash Map → Stores Key- value pair

↳ Set → store unique data elements

What is an Algorithm ?

→ An Algorithm is a step by step well defined procedure to solve a specific problem in a finite number of steps.

Why Are Algorithm Important ?

→ Efficient use of time and memory.

→ Helps solve real-world problems.

Some Algorithm Names.

(1) Breute force Algorithm.

↳ Try all possible combinations blindly untill the correct one is found.

Examples

↳ Generating Permutations.

↳ Password cracking.

They are usually very slow.

often has $O(n!)$ or $O(2^n)$ time complexity.

② Divide and Conquer :

→ Divide the problems into smaller, ~~question~~ sub problems, solve them recursively, and combine result.

Example.

→ Merge Sort : $O(n \log n)$

→ Quick short $O(n \log n)$ average

→ Binary search $O(\log n)$

③ Recursion.

→ A function call itself to solve subproblems

④ Bit Manipulation.

→ Use bitwise operator to solve problem efficiently.

⑤ Dynamic Programming (DP)

→ Break problems into overlapping subproblems, solve each only once, store the result.

⑥ Greedy Algorithm

→ Divide the problems into smaller subproblems,

At each step, choose the "locally optimal" choice hoping it leads to a global optimum.