Monday.
15 January 2024

Session 1

Data Structures.

good program.

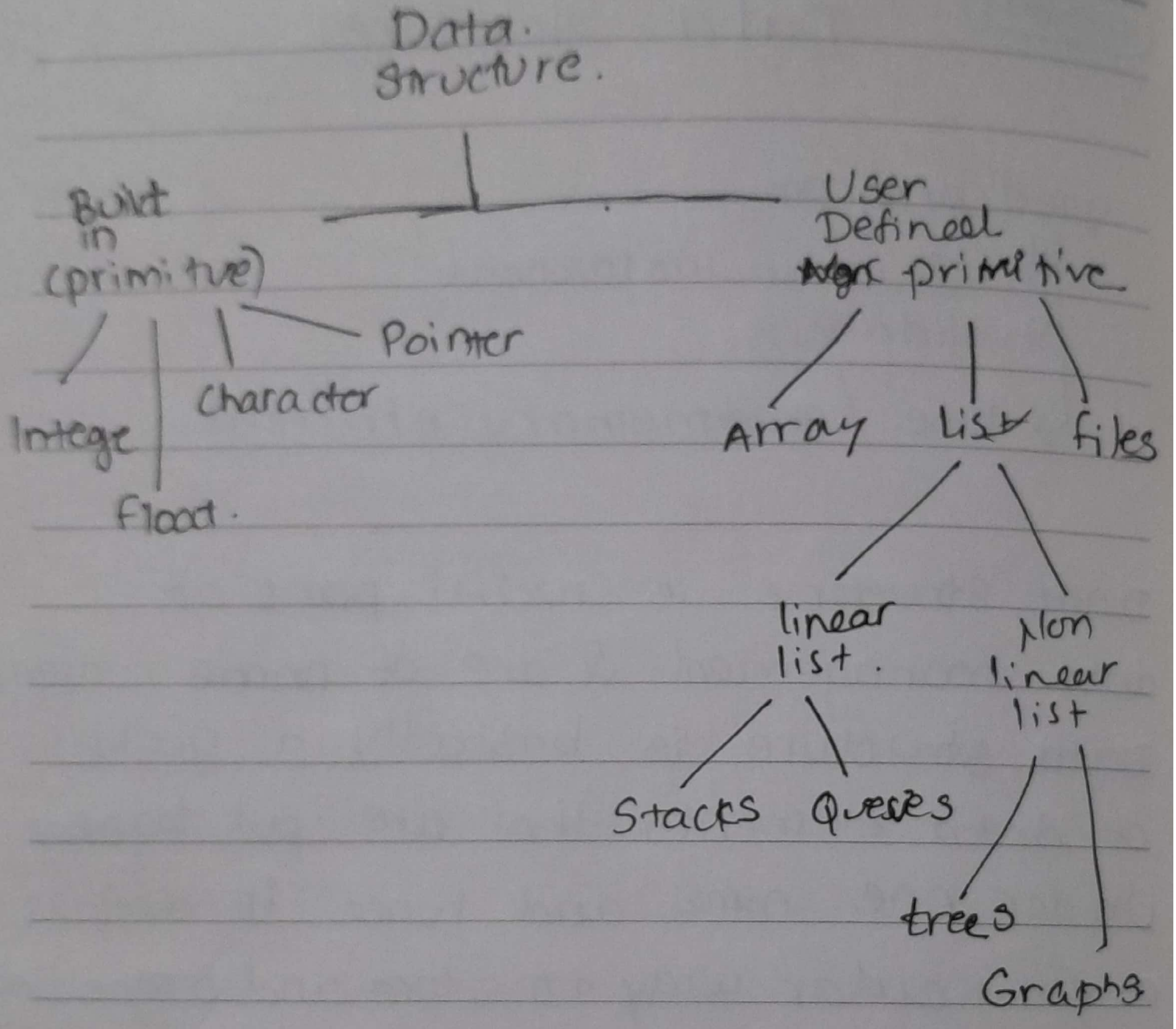
- Less no. of instructions.
- validations.
- time / ~~grad~~ memory efficient.

Data Structure is Crucial part of data management & are of prime concern Data structure is basically a group of data elements that are put together under one name and hence it defines a particular way to store and organizing the data in computer memory.

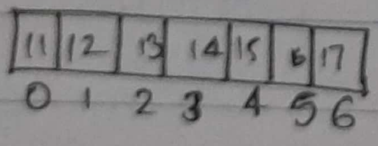
Requirements to write good program.

- Analysis of problem to determine basic operation.
- Quantify resource Resources constraints.
- Select appropriate data structure.

Memory + Time = ~~Journey's~~ resourses.



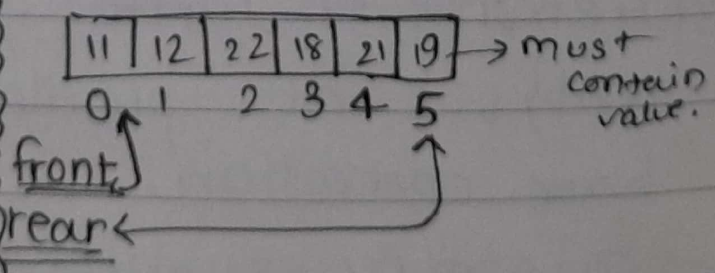
Array. Linear.



int array [7];

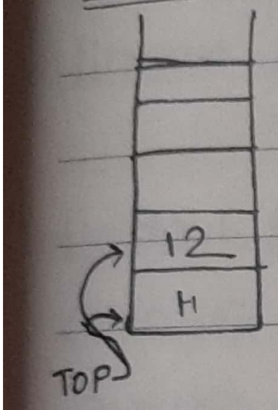
Queue: FIFO

first in first out.



last in firstout.
LIFO

Stack:



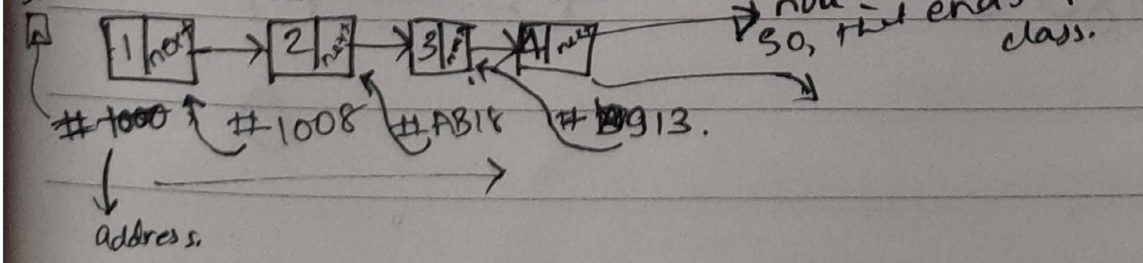
push → insert.
POP. → pop → delete.
peek → show the last element (display not pop)

2 actual structures → Array
↳ linked list.

others are implemented using array & linked list

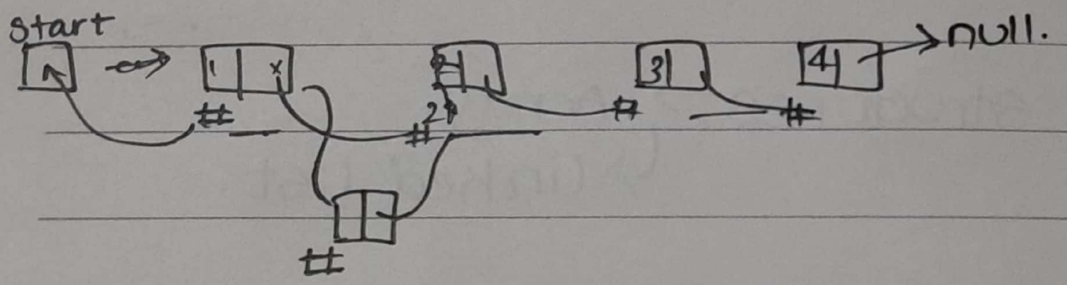
(ADT). Abstract ~~data~~ data. types

linked list

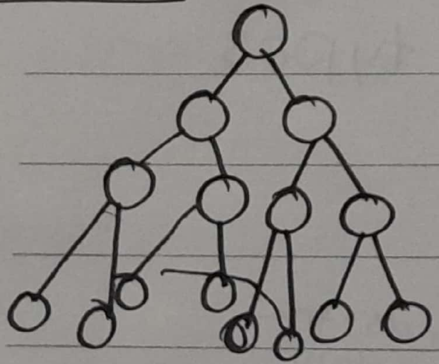


In linked list there are pointers which consumes lot of space whereas not in array.

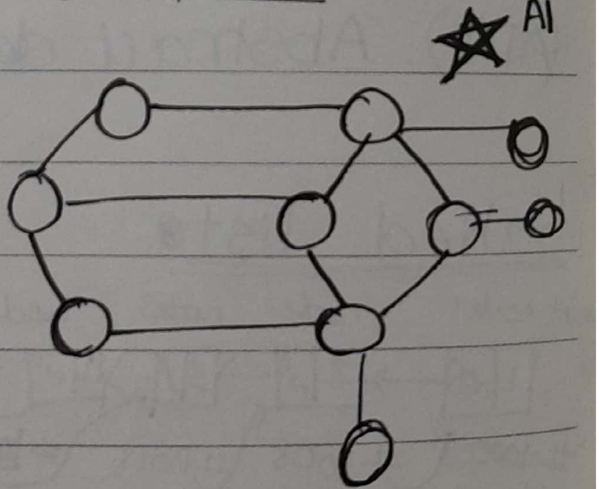
In linked list if one is deleted in just put next address it will be done



Trees.



Graphs:



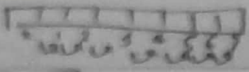
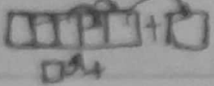
linear & Non-linear Data structures.

linear data structures include arrays, linked list, stacks and queues. They can be represented in memory in two different ways one way is to have linear relationship between elements by means of sequential memory location. and second way is to have linear relation between elements by means of links.

SESSION 2

- Primitive and non primitive
- Data type and Data structure

Data Structure Operations

- Traversing  from location to next to next location
  identify the index no.
- Insertion  shift by 1 then add □
- Deletion
- Searching
- Sorting
- Merging.

- array & pointer are linked to each other.

  int $\underbrace{\text{array}}$[]

Algorithm.

Set of rules.

- Unambiguous
- Input & output
- Finiteness.
- Feasibility.
- Independent. of any languages.

Algorithm Analysis

- Priori Analysis
- Posterior Analysis
- Algorithmic complexity

Time Complexity.

- first try to get the average case then find upper or lower.

★ Asymtotic Notations → O, Ω, Θ

used for calculating time complexity of an algorithm.

O → upper bound → function never grows faster than → worst situation time

Ω → lower bound → measure best amount of time an algorithm can possibly take to complete or best case-time complexity.

Θ → Average bound → average case of algorithm → fluctuate worst case & best case's between. $\frac{(n+1)}{2}$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^2\log n < n^3 < \ldots < 2^n < 3^n < \ldots n^n$

smaller ← . . . . → larger

11	12	13	14	15
0	1	2	3	4

$\Omega(1)$

$O(n)$

$\Theta(\frac{n+1}{2})$

$f(n) \Rightarrow$ algorithm.

$O(g(n)) \Rightarrow$ upperbound of $g(n)$.

$C \Rightarrow$ constant

Algorithm efficiency

Big O:.

The function $f(n) = O(g(n))$ if $\exists$ +ve constant C & n_0

Such that $f(n) \leq C * g(n)$ $\forall n \geq n_0$

$f(n) = 2n+3.$

$2n+3 \leq 2n+3n \quad | \quad \leq 2n^2 + 8n^2$

$5n \quad | \quad \leq 10n^2$

$2+3 \leq 5$

$\underline{\underline{=5 \leq 5}} \qquad \underline{\underline{5 \leq 10}}$

$f(n) = O(n).$

Omega (Ω)

The function $f(n) = \Omega(g(n))$ iff $\exists$ +ve constant c & n_0 such that $f(n) \geq c * g(n)$. $\forall\ n \geq n_0$.

$2n+3 \geq 5n$

$4n$

$3n$

$5 \geq 7$ ✗

$5 \geq 6$ ✗

$5 \geq 5$ ✓

$5 \geq 4$ ✓

Theta (θ)

The function

$f(n) = \theta(g(n))$ iff $\exists$ +ve constant C_1, C_2 & n_0

such that $C_1 * g(n) \leq f(n) \leq C_2 * g(n)$.

$$5n \leq 2n+3 \leq 5n.$$

$\Omega(n)$ $\qquad\qquad$ $O(n)$

$$\Theta(n).$$

$f(n) = 2n^2 + 3n + 4.$

$$2n^2 + 3n^2 + 4n^2 \leq 9n^2 \Rightarrow O(n^2)$$

$f(n) = 2n^2 + 3n + 4$

$$2n^2 + 3n + 4 \geq 8n^2 \Rightarrow \Omega(n^2)$$

$$9 \geq 8$$

$~~f~~f(n) = 2n^2 + 3n + 4$

$$8n^2 \leq 2n^2 + 3n + 4 \leq 9n2 \Rightarrow \Theta(n2).$$

$f(n) = n^2 \log n + n$