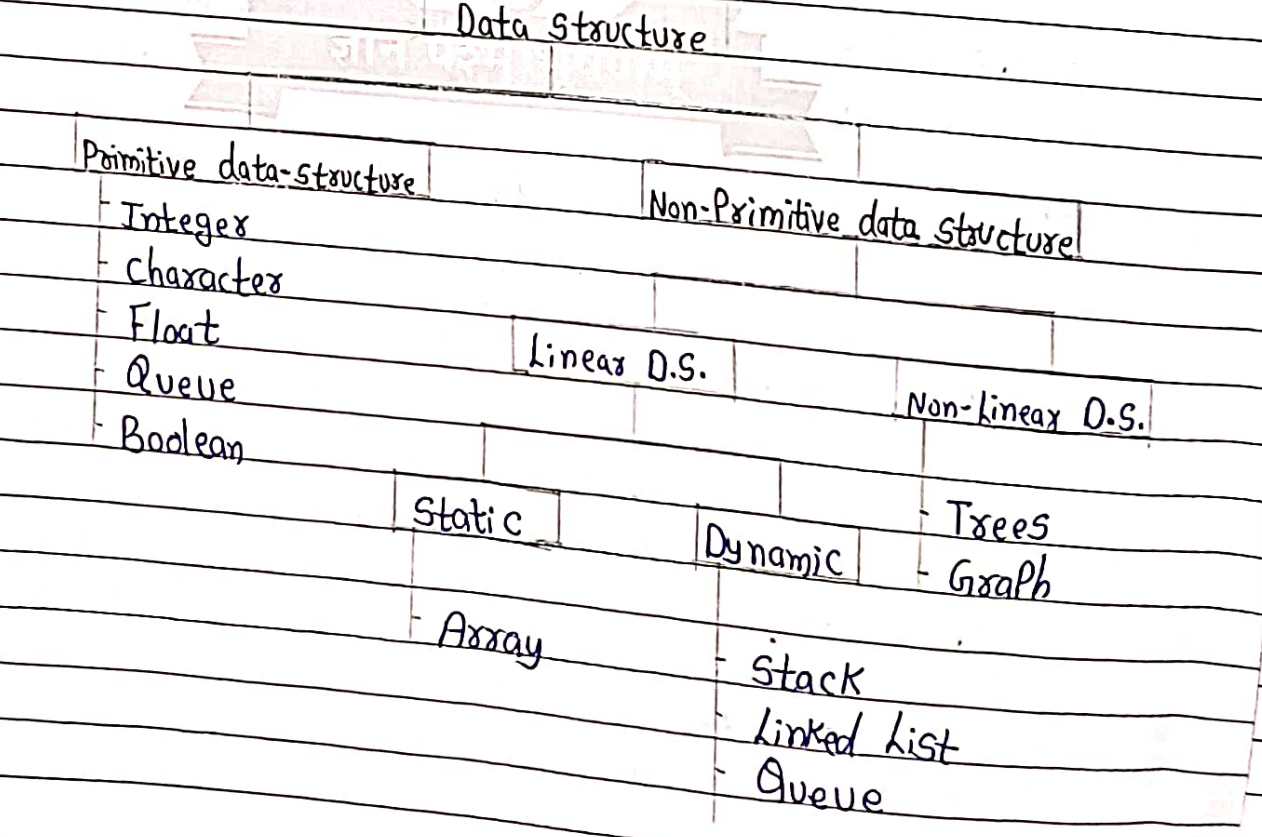
Que:-1 Write down classification of Data structure in detail.

Ans. → A data structure is a storage that is used to stor and organize data.

→ It is a way of arranging data on a compute so that it can be accessed and ~~uploade~~ updated efficiently.

→ Data Structure is way of organising all data item and relationship to each other.

* Types of data structure :-



(1) Primitive Data structure :-

- These are basic structure and are directly operated by machine instructions.

For example :- Integer, float, Character

(2) Non-Primitive Data Structure :-

- They are derive from the Primitive data structure. It is a collection of same data type or different data type Primitive data structure.

- These data structures can't be manipulated or operated directly by machine-level instructions.

For example :- Array, trees & Graph.

⇒ Linear Data structure :-

- The arrangement of data in the sequential manner is known as linear data structure.
- The data structure used for this purpose are Arrays, Linked list, Stacks and Queues.
- In this data structure one element is connected to only one anotherelement in a linear form.

- Data structure can also be classified as Static or Dynamic data structure.

⇒ Non-linear Data Structure :-

- When one element is connected to the 'n' number of elements is known as non-linear D.S.
- In this case, elements are arrage in random method.

For example :- Trees and Graph.

Que:2 What is Algorithm ? Explain its types with example.

- An algorithm is a process or set of rules required to perform calculations or some other Problem-solving operations especially by a computer. The formal definition of an algorithm is that it contains the finite set of instructions which are being carried in a specific order to perform the specific task.
- It is not the the complete Program or code ; it is just a solution (logic) of a problem, which can be represented either as an informal description using a Flowchart or Pseudocode.

* Types of Algorithm:

(1) Sequence Algorithm :-

- This algorithm Perform successively one by one without skipping any steps.

- No selection Procedule or condition branching exits in Sequence algorithm.

Example:-

//adding two numbers.

Step1 : Start
Step 2 : read a,b
Step3 : Sum = a+b
Step 4 : write Sum
Step 5 : Stop

(2) Selection Algorithm :-

- The Sequence type of algorithm are not sufficient to solve the Problem which involves decisions and conditions.

- In order to solve the Problem which involve decision making or option selection, we go for Selection type of algorithm.

Example :-

Algorithm Smaller (A, B)
A, B :- values to be checked.

Step 1 :- If (A < B)
Return A

Step 2 :- Else
Return B

(3) Iteration algorithm :-

- Iteration type of algorithm are used in solving problem which involves repeation of statement.
- In this type of algorithms, a particulars number of statements are repeated 'n' no. of times.

Example :-

Algorithm Print-n (n)

(1) I ← 1
(2) Repeat up to step until I <= N
(3) Print I
(4) I ++
(5) Return

Que:-3 What is Time Complexity?

Ans
- The time complexity of an algorithm is the amount of time or the number of steps needed by Program to complete it tasks.
- The time complexity of an algorithm is denoted by the big O notation.
- Here, big O notation is the asymptotic notation to represent the time complexity.

Example:-

```
#include <stdio.h>

Void PrintNumbers (int n)
{
  for (int i =1; i<=n; i++)
  {
    Printf ("%d", i);
  }
}
int main ()
  {
    int num =5;
    PrintNumbers(num);
    return 0;
  }
```

Que:- 4 What is Space Complexity?

Ans. - The amount of space occupied by an algorithm is known as space complexity.

- An algorithm is said to be efficient if its occupies less space and require the minimum amount of time to complete its execution.

Example :-

```
#include <stdio.h>

void createArray(int n)
{
  int* arr = (int*)malloc(n*sizeof(int));
  for(int i=0; i<n; i++)
  {
    arr[i] = i+1;
  }
  free(arr);
}
int main(){
  int num = 5;
  createArray(num);
  return 0;

}.
```

Que:-5 Write down Asymptotic Notation in detail.

Ans

- The commonly used asymptotic notation used for calculating the ~~xx~~ running time complexity of an algorithm is given below:

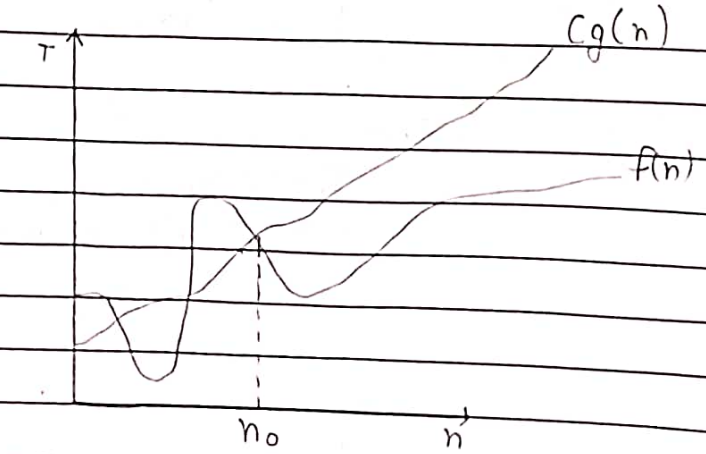
1] Big oh notation (O)

2] Omega Notation (Ω)

3] Theta Notation (Θ)

1] Big oh Notation (O) :- (worst case)

- Big ~~oh~~ O notation is an asymptotic that measures the Performance of an algorithm by simply Providing the order of growth of the function.



$f(n) \leq Cg(n)$
$n \geq n_0$
$C > 0, n_0 \geq 1$

$\boxed{F(n) = O(g(n))}$

$Cg(n)$ = term
$f(n)$ = function
n = Input Size
C = (Positive num)
n_0 = Intants

→ This measures the Performance of an algorithm by Simply Providing the order of growth of the function.

→ This notation Provides an upperbound on a function which ensure that functions never grows faster than the upperbound / worst case.

Example:-

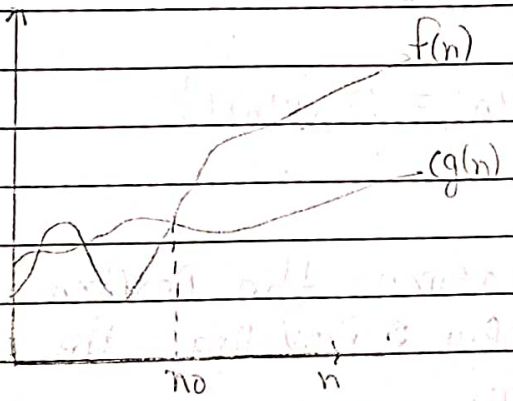
If $f(n)$ and $g(n)$ are two functions defined for positive integers.

then,

$f(n) = O(g(n))$ as $f(n)$ is big oh of $g(n)$ or $f(n)$ is on order of $g(n)$ if there exists ~~cont~~ constants C and n_0 such that,

$f(n) \leq Cg(n)$ for all $n \geq n_0$

2) Omega notation :- (Ω) (Best case)



$f(n) \geq c(g(n)$

$n \geq n_0$

$c > 0, n_0 \geq 1$

$\boxed{f(n) = \Omega(g(n))}$

→ It basically described best case scenarios whis is oppositto to big o notation

→ It is formal way to represent lower limit bound to an algorithm running time.

→ It measures the best amount of time an algorithm can take to complete or best case time complexity.

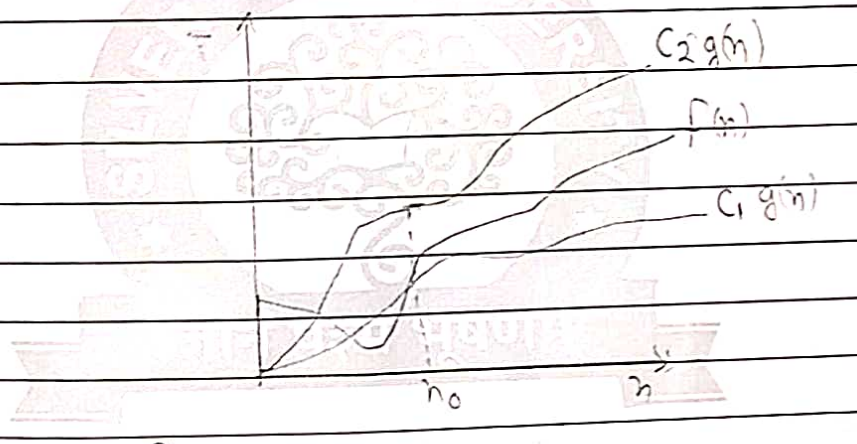
Example :-

If f(n) and g(n) are two functions defined for positive integers.
then,

$f(n) = \Omega\, g(n)$ as f(n) is omega of g(n) or f(n) on the order of g(n) if there exists constants c and n_0 such that

$f(n) >= cg(n)$ for all $n \geq n_0$ and $c > 0$

3] Theta notation (Θ) :- (Average Case)



$c_1 g(n) \leq f(n) \leq c_2 g(n)$

$n \geq n_0$

$c > 0, n_0 \geq 1$

$\boxed{f(n) = \Theta(g(n))}$

- The theta notation mainly describes Average case notation
- It represent realistic time complexity of an algorithm
- Big theta is mainly use where, the value of worst case and best case is same.