→ Data:

Data refers to distinct pieces of information that are collected, store, and processed to provide inside knowledge. or support dicissus information. Data can be any represetion of facts, numbers, texts, symbols, images, audio, video or more that has some meaning. Data can be raw or processed, and it serves as the foundation for ~~ganterating~~ genrating information and knowledge.

→ Data Structure

A data structure is a storage that is used to store and organized data. It is a way of arranging data on a compuker. So that it can ~~atass~~ access and updated efficiency.

A data structure is used for orgaz-ing the data. It is also used for processing, retrieving and storaging data.

→ Information

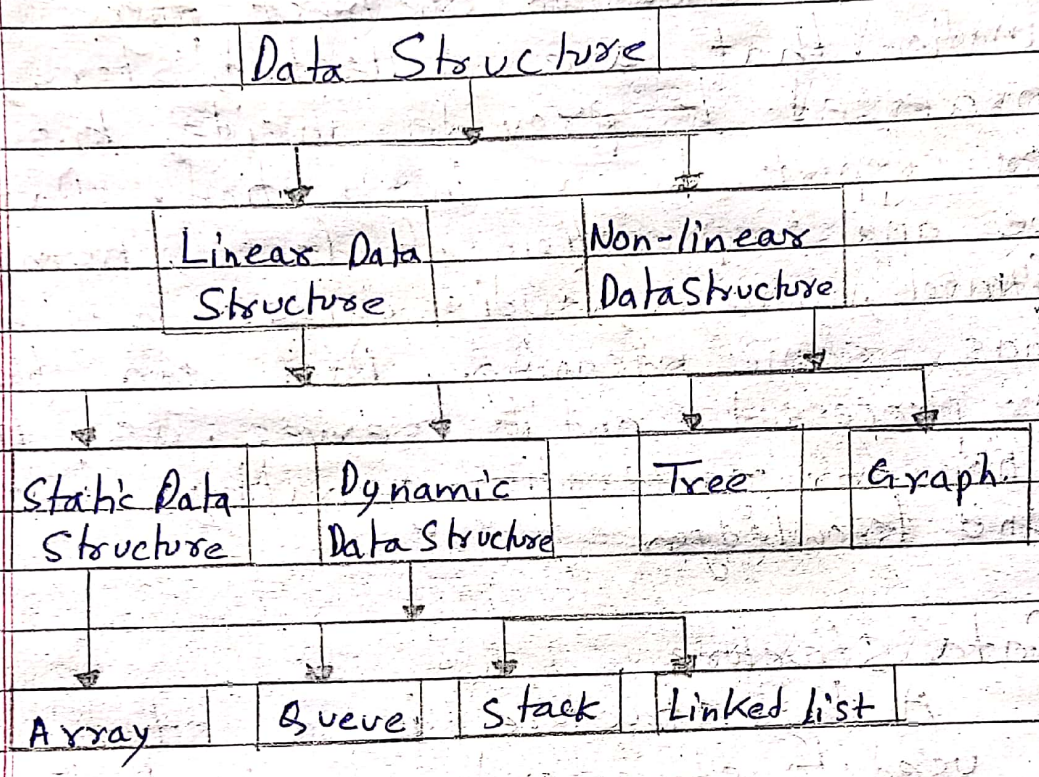
Information is known as facts provided or learned about something or some one.

→ Data type:

The ~~Type~~ Data type is basically a type of data that can be used in different computer program. It signifies the types like integer, floatetc, the space like integer will take 4- bytes.

Scanned with OKEN Scanner

→ Classification of Data Structure.



→ Linear Data Structure

Data Structure in which data element are arranged in squencely or linearly where each element is attracted to it's previous and next adjecent elements is called a linear Data Structure.

Example :-

1) Array
2) Stack
3) Queue
4) Linked list.

→

→ Static Data Structure.

Static Data Structure has a fix amount of memory size. It is easier to access the elements in a data in a static Data Structure. Example of this data structure is an array.

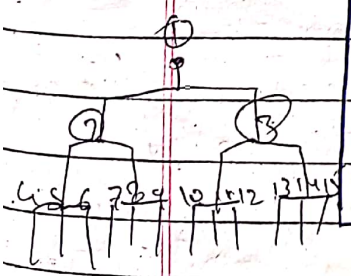
→ Dynamic Data Structure.

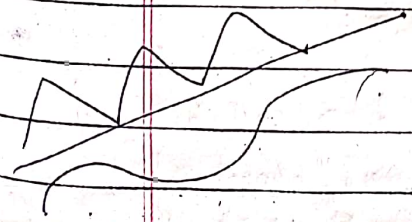
In ~~Dy~~ Dynamic Data Structure the size is not fix. It can be randomaly updated during the run time. which may be consider efficent concering the memory complete of code. Example:- Queue, Stack or linked list.

→ Non-linear Data Structure.

Data Structure where data element are not placed sequencly or linearly are called Non-linear Data Structure. In a Non Linear ~~dy~~ data Structure we can't travers all the element in a single run ~~of~~ only. Example:- tree or Graph.

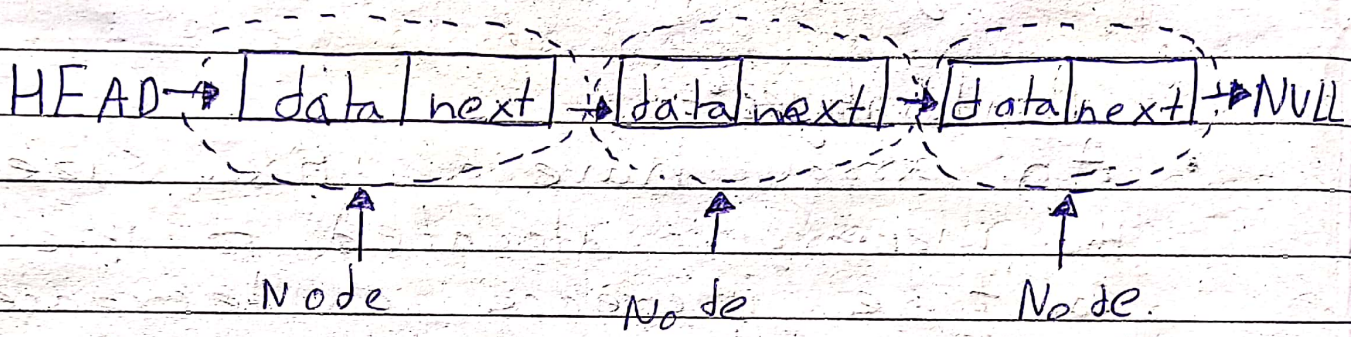
Note:-
Queue → FIFO (First in first out)
Stack → LIFO (Last in First out)





→ Linked list =

A linked list is linear data structure, in which the elements are not stored at contiguous memory location. The elements in linked list are linked using pointers as shown in the below image.



→ Stack

A stack is an ~~abs~~ abstract data type that holds an ~~or~~ ordered linear sequence of items. ~~In~~ In contrast to ~~acc~~ a queue, a stack is a last in first out (LIFO) structure. A real life example, it's a stack of ~~te~~ plates. ~~You~~ You can only take a plate from a top of a

Date 6/9/23
Page

stack and you can only add a plate to the top of the stack.

→ Queue

A queue is define as a ~~line~~ linear data structure that is open at both ends and the operation performed first in first out (FIFO) order. We define a queue to be list in which all addition to the list are made at one end & all deletion from the list are made at other end.

→ Tree

Tree hierarchical data structure with a root node and child node.

Each node have zero or more child node.

Binary tree are common type tree with which each having at most to children.

Example include :- binary tree, binary search tree, AVL tree, heap tree.

→ Graphs

Graphs consists of nodes (vertices) and adjust (connection) bet^n those node.

They can represent complex relation-ship, networks and various real world Scenery.

Graphs can be directed (edge have

direction) or Undirected (edges have no direction)

Example includes social network, road network and computer network.

→ Different types of trees:-

1) Binary tree:-

A tree in which each node has as ~~most~~ two children, which are reffer to as left child and right child.

The structure remap an inverted tree, with the single node at top called the root and branching downward into sub-tree, Binary tree are common use in computer science and have various application including data storage and ~~retrivet~~ retrieve, expression, evolution, network routing and game AI. They can also be use to implement various algorithm such as searching, sorting and graph algorithm.

→ Operation on binary tree

1) Inserting a binary element.
2) Removing a binary element
3) Searching a binary element
4) Delection for a binary element
5) Traversing a binary element.

Scanned with OKEN Scanner

Kaggle.

Operation on Data Structure.

1) Traversing:-
Traversing data structure means to visits a element store in it. It's visits a data in a systematic manner. This can be done by any type of data structure.

2) Searching:-
Searching meas to find the particular element in the given data structure. It is consider as sucessfull random requirement element is found. searching is the operation which we can be perform on data structure like array, linked list, tree, graph.

3) Insertion:-
It is the operation which we applied on all the data structure. Insertion mean to add an element is a given data structure. The operation of insertion is succesfull when the required is add to the required data structure. It is unsucessfull in some cases when the size of data structure is full and when there is no space in the data structure to add any additional elements.

Scanned with OKEN Scanner

The insertion has the same name. ~~has the~~ as an insertion data structure as a array, linked list, graph, & tree. In stack, these operation is called push. In the queue, this operation is called enqueue.

3) Delection:-

It is the operation in which applied all data structure. Delection mean to delete element to the given data structure. The operation of delection. If is successful when the required element is deleted. from the data structure. The delection have the same ~~m~~name as a delection in the data structure as an array, linked list, tree, etc. In Stack these operation is ecolled pop. In queue these ~~oper~~operation is called de queue.

4) Creation:-

It reserve memory for program element by declaring them. The creation of data structure can be done during:-

1) Compiled time
2) Runtime

~~We~~ You can ~~ca~~ use malloc()

function.

Selection:-

It's ~~like~~ select specific data from present data. You can select any condition specific data by giving ~~condition~~ on loop.

Update:-

It updatae the data structure. You can also specific data by giving some condition in a loop. like select approch.

Sort:-

Sorting data on a paxticular data. (assending and deassending)

Merge:-

Margeing data of two different order in a spefic data structure may assend or decesend.. We use merge sort to ~~merge~~ merge. sort data

Split data:-

Dividing data in to different sub-parts to make the process complete in less time.

VVV ΣMPP

Asympotic Notation.

Asymptotic Notation used to describe the running time of an algorithm. how much time an algorithm take with a given input and. There ~~are~~ three different notation. (O, θ, Ω) Big O, Big θ, Big Ω. Big θ is used when the running time is same for all cases, Big O for the worst case running time. Bing Ω for the best case running time.

1) Big O Notation:- represent

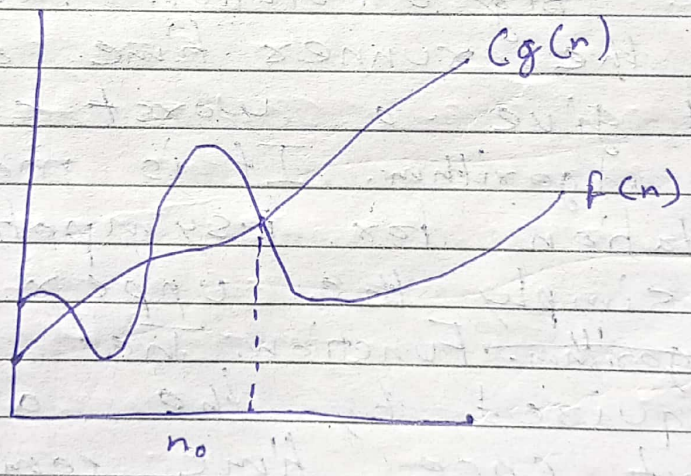
Big O Notation the upper bound of the runner time of algorithm. Therefore it give a worst case complexity of an algorithm. It is most widely used. Notation for asympotic anayliso. It simply the upper bound of an algorithm. Function. The max^m time required by the algorithm or the worst case time complexity. It return highest possible output value (O) for a given input.

The above expression can be describe as a function f(n) belongs to the set O g(n). if there exsit a positive constant C such that it lies beth O and C g(n) for

sufficiently large $n(0)$. For any value of n the running time of an algorithm doesn't cross the time provided by a $O(g(n))$.

$O(g(n)) = \{ f(n)$ there exist positive constant C and n_0 such that $0 \leq f(n) \leq C g(n)$ for all $n \geq n_0 \}$

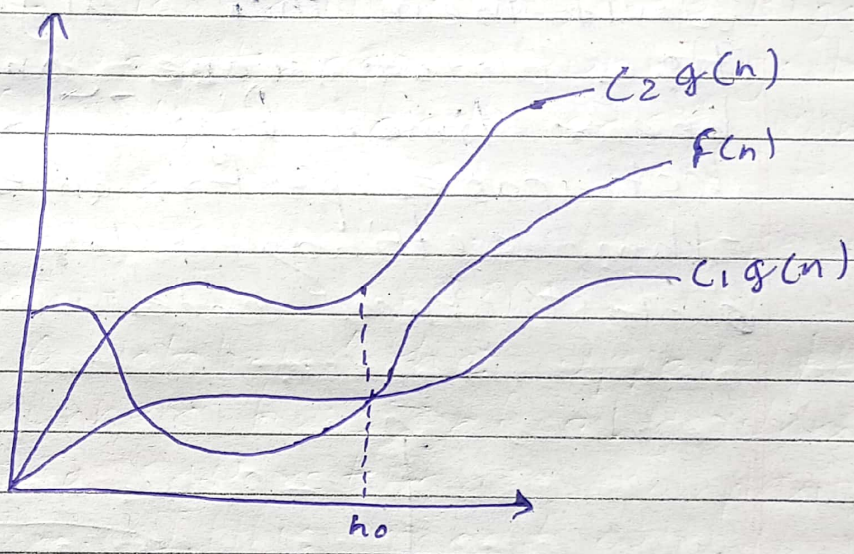
Since it gives the worst case running time of an algorithm it is widely used to anaylise an algorithm as we are alway's intrest the worst case scenario.



$f(n) = O(g(n))$

Big θ

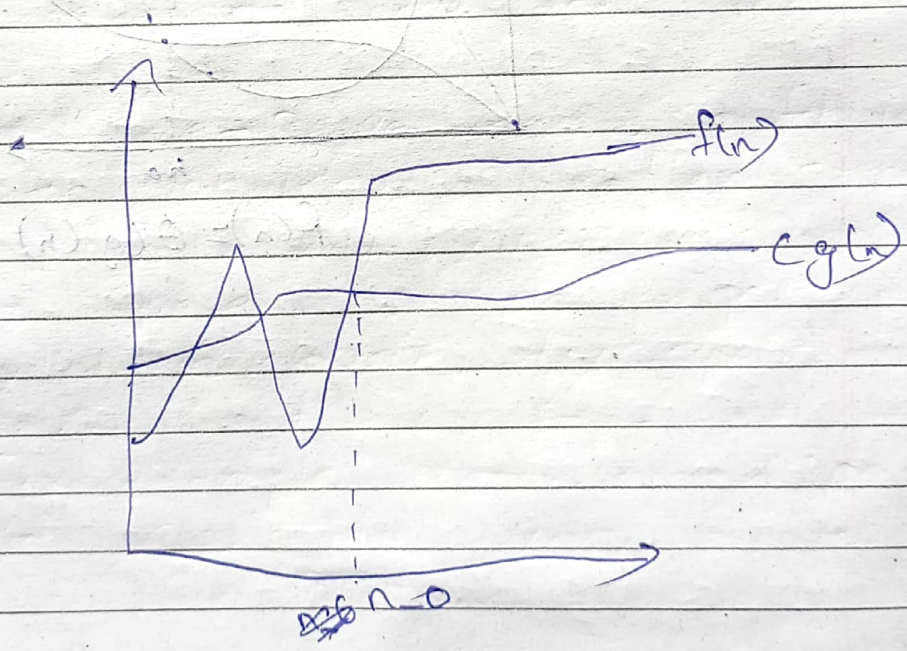
θ notation enclosed the function from above and below. Since it represent the upper and lower bound of the running time of an alg^n, It is used for analysis the average case of an algorithmo.



$f(n) = \theta(g(n))$

Big Ω

The big Ω notation gives you a lower bound of the running time of an algorithm. So big Ω(n) means the algorithms run at least at the n time but could actually take a lot longer. The big O notation gives you a upper bound so O and would mean the algorithms run in it's worst case in an or linear time. Big Ω represent the best case performance for an algorithm, setting a lower bound on how fast the code can perform



f n = Omega(g(n))

	Big O	Big Ω	Big Θ
1	It is like (≤) rate of grow of an algorithm is ~~less~~ (≤) to a specific value.	It is like (≥) rate of grow in (≥) to a specific value.	It is like (=) meaning the rate of growth is (=) specific to a value.
2	The upper of an algorithm represent by big O notation. Only the above function bound by big O. Asymopotic is given big O notation.	The algorithm lower bound is represented by Ω. The Asymopatic ~~nata~~ notation is given by Ω notation.	The bounding of function from above and below is represent by Θ notation. The exact sympothic notation behaviour is done by this Θ notation.
3	Big O is upper bound.	Big Ω is lower bound	Big Θ is ~~right~~ bound.
4	It is define as a upper bound. And upper bound on an algorithm is the most amount of time required (the worst case performance)	It is define as lower bound on an algorithm is the best amount of time require (the most efficient way, in other word best case)	It is define as tightest bound. and tightest the best of all best case time that the algorithm can take.
5	Mathmatically $0 \leq f(n) \leq c(g(n))$ for all n ~~greater~~ $\geq n_0$.	Mathmatically $\Omega 0 \leq c(g(n)) \leq f(n)$, for all $n \geq n_0$	Mathmatically θ is $0 \leq c_2 g(n) \leq f(n) \leq c_1 g(n)$ for $n \geq n_0$.

Difference

Scanned with OKEN Scanner

The time-Space trade off.

A time Space trade off also known as time memory trade off algorithms Space-time in computer science is a case where algorithms or program trade's increase space uses where it decrease time.

A time-Space trade off is a way of soluing a problem or calculation in less time ~~my~~ by using more storage. ~~memory~~ Space or by soluing a problems in very little Space by spending a long time.

Types of trade off

1) Compressed/ Uncompressed Data :- A space time off can be applied to the problem of data storage. If data is stored uncompressed, it takes more space less time. If the data stored compressed, it take less space but more time to run the decompression algorithm.

2) Re ranging (store images :-
Storing only the source and ranking it as an image every time the page is requested would be trading time for space more time & use but less space.
↳ Storing the images would be trading space for time more space used. but less time.

3) Small code (with loop) large code (without loop)
small code occupied less space in memory but it requires hight computation time which is required. for jumping back to the beging of the look at the end of ittreation.
Large code or loop unrolling can be trade for higher program speed.

Scanned with OKEN Scanner

- It occupied more space in memory but required lesso computation time.

3) Lookup Table / recalculation:-

In lookup table, an implementation table can include entire table which reduces computing time but increase the amount of memory needed.

It can recalculated i.e computable entries as needed increasing computing time but reducing memory requirement.

Operation of Stack:-

A stack is a data structure which is used to store data in a linear fashion. It is an abstract data type in which data is inserted and deleted from a single end which is the stacks top. It follow the Last in first out (LIFO) principle i.e. the data which is incerted most recently in the stack will be deleted first from the stack.

Operation of stack:

Expression parsing.

The way to write arithmetic expression is known as a Notation. An arithmetic expression can be written in three equivalent notations i.e without changing the essence or output of an expression these Notation are:-

1) Infix Notation:

We write expression in infix notation. i.e a-b+c, where operators are use in betⁿ operands. It is easy for us human to read, write and speak in infix notation. but the same doesn't goes well in computing devices. An Algorithmic infix notation could be difficult and costly in terms of time & space consumption.

2) Pre-fix Notation:-

In this Notation, operator is pre-fix to operands. i.e. operator is written is for example, +ab. this is equivalent pre-fix notation is also known as polish notation.

3) Post-fix Notation:-

This notation style is know as reversed polished notation in this notation style the operator is

briefly

post-fix to the operands. i.e. the operater is written after the operands. for example ab+. This is equivalent to it's infix notation a+b.

The following table ~~brefity~~ briefly try to show the difference in all the three notation

SN	Infix	Prefix	Postfix
1	a+b	+ab	ab+
2	(a+b)-c	-+abc	abc-+
3	a*(b+c)	*a+bc	abc+*
4	a/b+c/d	+/ab/cd	ab/cd/+
5	(a+b)*(c+d)	*+ab+cd	ab+cd+*
6	((a+b)*c)-d	-*+abcd	ab+c*d-

What is the analysis of the Algorithm.

Analysis of an algorithm is the process of finding the computational complexity of any algorithm. By computational complexity, we are referring to the amount of time taken, space, and any other resources needed to execute (run) the algorithm.

The goal of algorithm analysis is to compare different algorithms that are used to solve the ~~same~~ same problem. This is done to evaluate which method solves a certain problem more quickly, efficiently, and memory wise.

Usually, the time complexity is determined by the size of the algorithm's input to the number of step taken to execute it. Also, the space complexity is mostly determined by the amount of storage needed by the algorithm to execute.

What is algorithm? Full Explaination

→ ~~The step by step description of any~~

→ An algorithm is a well defined list of steps for ~~sto~~ solving a particular problem.

Example of algorithm.

A algorithm for add two numbers.

Step 1: Begin
Step 2: Intput two numbers a & b
steps: sum of a & b
step 4: Display output of addition
Step 5: Exit.

Property of algorithm:-

Input

Algorithm Complexity (performance):-

Algorithm complexity refers to the analysis of an algorithm performance in term of time and space required as a function of input size.

i) Space Complexity.

Total amount of memory required by an algorithm for it complete execution.

a) constant Space complexity:-

Example:-
```
int sqrt(int n)
{
    return(n*n);
}
```

b) linear space complexity:-

Example:-
```
int add(int arr[], int n)
{
    int sum=0, i;
    for(i=0; i<=n; i++)
    sum = sum + arr[i];
    return sum;
}
```

ii) Time complexity:-

Total amount of time required by an algorithm to it's complete execution called time complexity.

a) constant time complexity:-

Example:-
```
int sqrt(int n)
{
    return n*n;
}
```


Scanned with OKEN Scanner

n*n = 1 unit of time
return = 1 unit of time.

b) linear time complexity.

Example :-

```
int add(int arr[ ], int n)
{
    int sum=0, i;
    for(i=0; i<=n; i++)
    sum=sum + arr[i];
    return sum;
}
```

Application of stack.

1) Memory Management.
2) Function call
3) ~~Experer~~ Bank tracking.
4) Browser History
5) Task management OS.