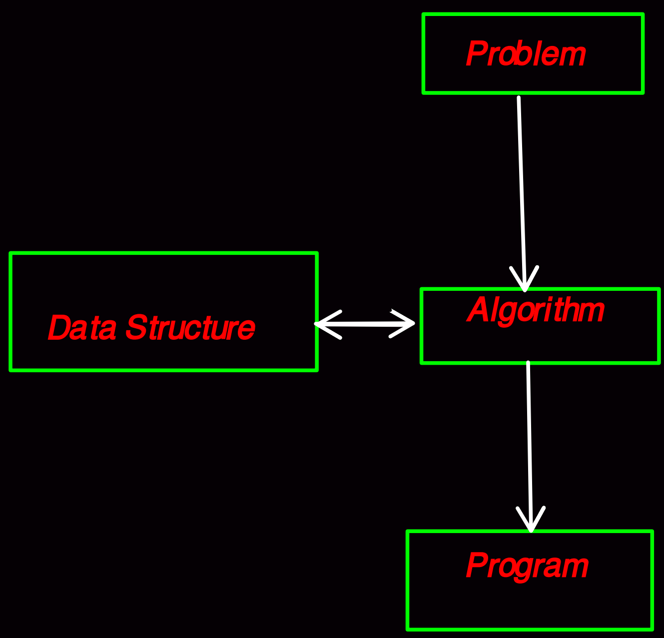
DATA STRUCTURE

Defination : Data Structure is logical and mathematical model of sorting and organizing data in a particular way that it can be required for designing and implemention of Algorithm.

Example: Array, Linked-List, Stack, Queues etc.



Data : Data are simply values or sets of values.

Data items: Refers to a single unit of values, items are devided into subitems.

Example: Md. Sohanur Rahman Hridoy

Record : Collection of various data item.

File : Collection of record of one type.

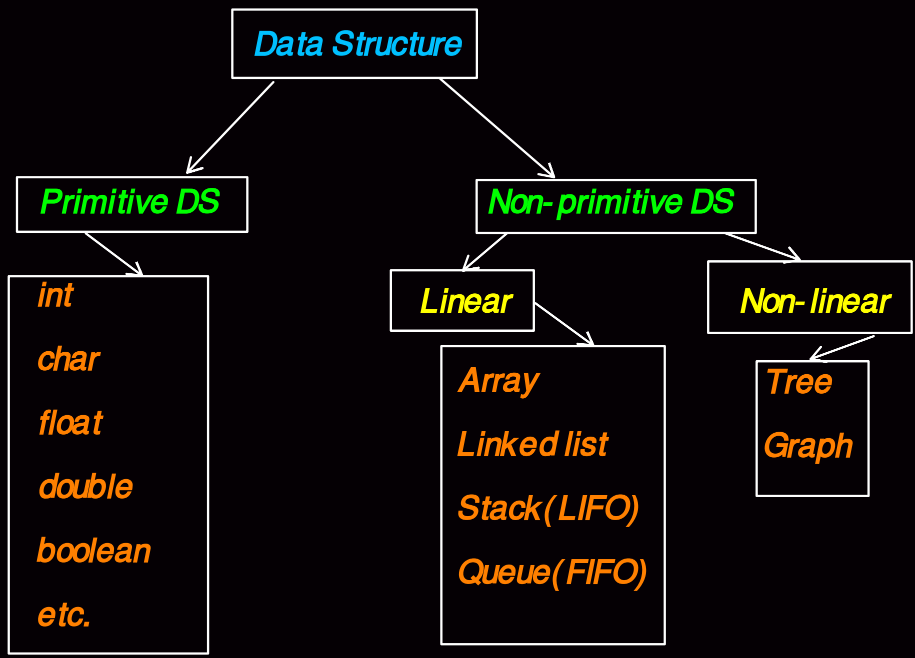
Entity: *An entity is something that has certain attributes or properties which may be assigned values.*

Field : *Single elementary unit, representing attrubute of an entity.*

Information: *Data with attributes, meaningfull data.*

Name	Age	Student ID
Md. Ozaer Hossain	*23*	*2102001*
Istiyak Ahmed Mahi	*22*	*2102002*
Zarif Tajul Arnob	*23*	*2102003*
Md. Sohanur Rahman Hridoy	*22*	*2102004*

Classification of Data Structures :



Primitive : Data Structure that directly operate upon machine instruction. Those are predefined operation & properties.

Non-Primitive : Derived from primitive and not directly operate upon machine.

Linear	Non-linear
01. All element arranged in linear order, where each element has the successor and predecessors except first & last element.	01. This DS doesn't form a sequence. Data element are arranged in heirarchical.
02. Single level involved.	02. Multilevel involved.
03. Data element traversed in single run.	03. Can't be traversed in single run.
04. Used in Software development.	04. Used in AI and DIP.

Data Structure Operations :

1. **Traversing :** Accessing each record exactly once.

2. **Searching :** Finding the location of the record with given key.

3. **Inserting :** Adding new record of DS.

4. **Deleting :** Removing a record from DS.

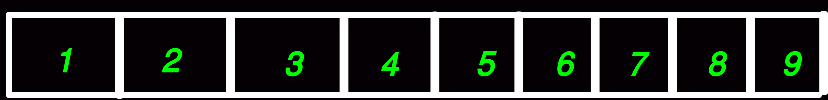
5. **Sorting :** Arranging the record in some order.

6. **Merging :** Combining two different sorted file into single sorted file

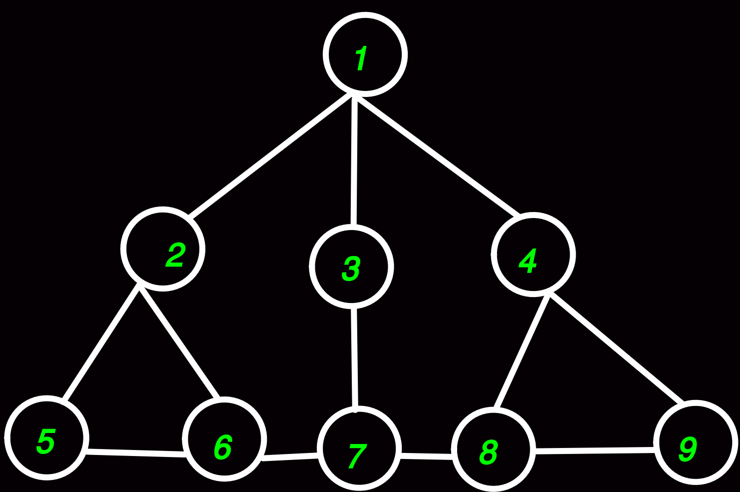
Abstract Data Type (ADT) :

ADT refers to a set of value associated with operation of function. With ADT we know what a specific data type can do but how it actually does that is hidden.

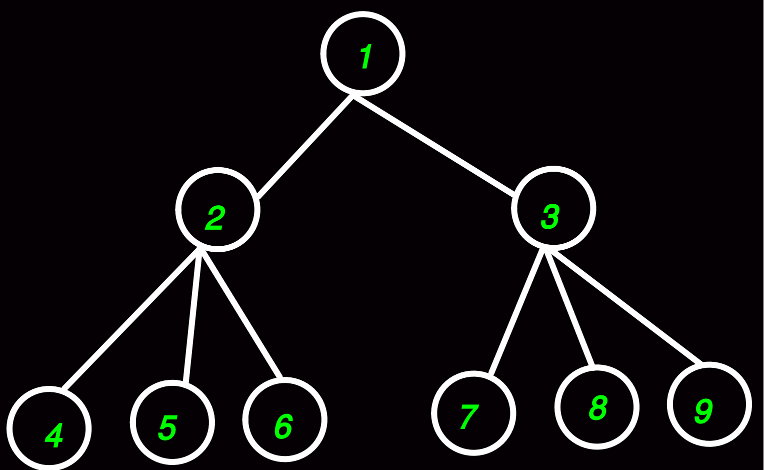
Example :



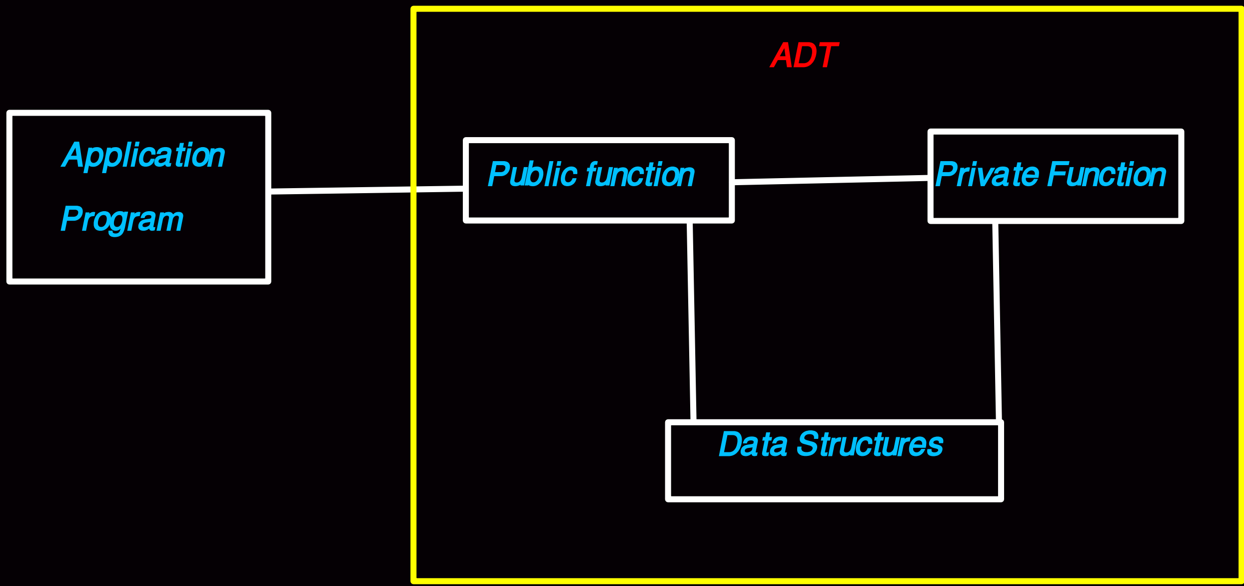
List



Graph



Tree



Example :

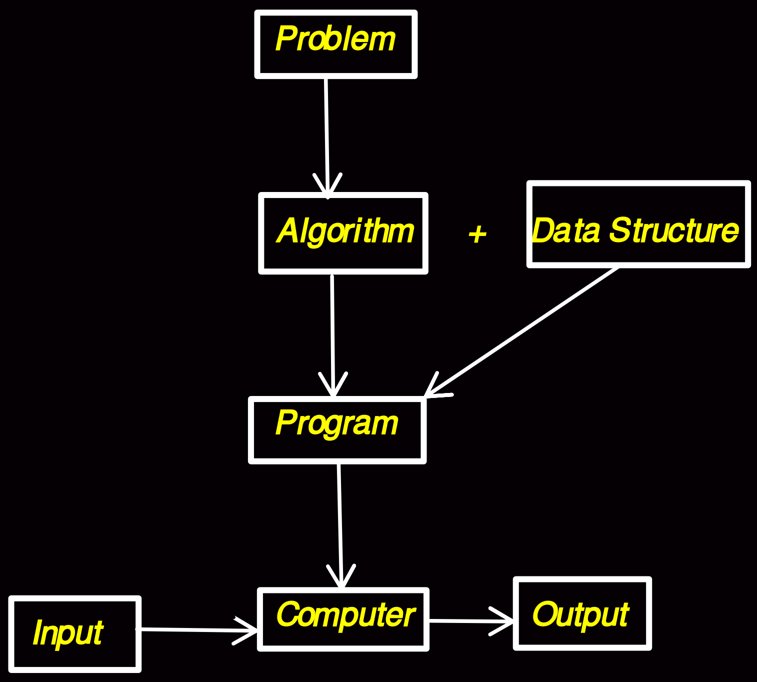
Array : sizeof()

Stack : push(), pop()

Queue : insert(), delete()

List : size(), insert()

DESIGN AND ANALYSIS OF ALGORITHMS



The word " Algorithm " comes from the name of parsian author Abu Ja' far Mohammad in 825 A. D.

Defination 1. 1 : An algorithm is a sequence of computational step that transform the input into output.

(Thomas H. Cormen)

Defination 1. 2 : An algorithm is a finite step of intruction that is followed , accomplishes a particular tast.

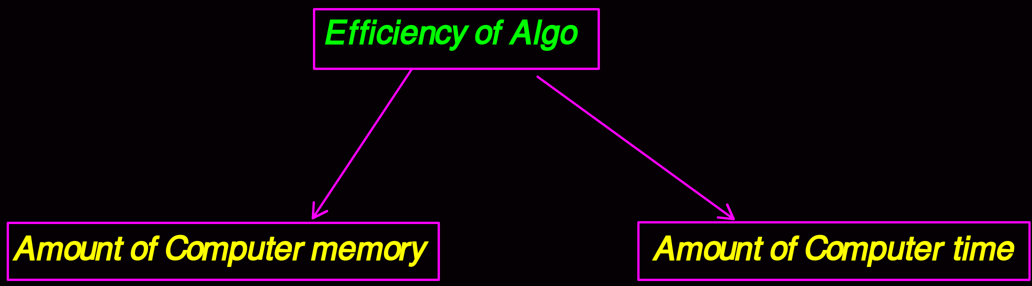
(Sartaj Sahni)

Criteria for Algorithm :

1. Input : Zero or more.
2. Output : one.
3. Definiteness : Clear & unambiguous.
4. Finiteness : Finite number of steps.
5. Effectiveness : Performance in terms of time & space and solve the desired problem.

Analyzing Algorithm :

Analyzing of Algorithm is required to dectate the Correctness and Measurement the efficiency of Algorihm.



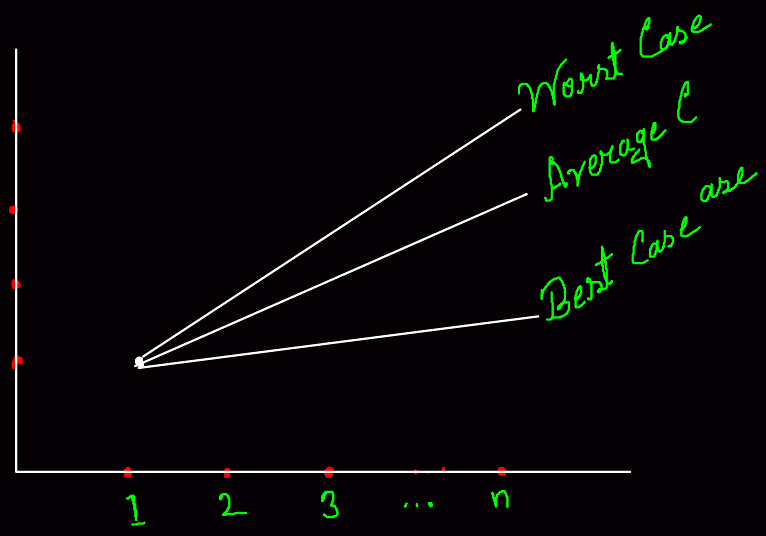
There are three types of Analysis :

1. **Worst Case :** *Maximum number of steps taken on any instance of size ' n' .*

2. **Best Case :** *Minimum number of steps taken on any instance of size ' n' .*

3. **Average Case :** *Any average number of steps taken on any instance of size ' n' .*

No. of steps



n(no. of elements)

Example :

Bubble Sort :

Best	Avg.	Worst
O(n)	O(n2)	O(n2)

Space : O(1)

Complexity of Algorithm :

Time Complexity : Amnount of computer time need to run to completion .

Rules :

1. Single Statement -

$c = a + b \longrightarrow O(1)$

2. Loop -

$$\left.\begin{array}{l} \text{for } i \leftarrow 1 \text{ to } n \\ \quad s \leftarrow s + a[i] \end{array}\right\} O(n)$$

P1

3. Nested Loop -

$$\left.\begin{array}{l} \text{for } i \leftarrow 1 \text{ to } n \;\}\, n \\ \quad \text{for } j \leftarrow 1 \text{ to } n \;\}\, n \cdot n = n^2 \\ \quad\quad s \leftarrow s + a[i] \;\}\, n \cdot n \cdot 1 = n^2 \end{array}\right\} O(n^2)$$

P2

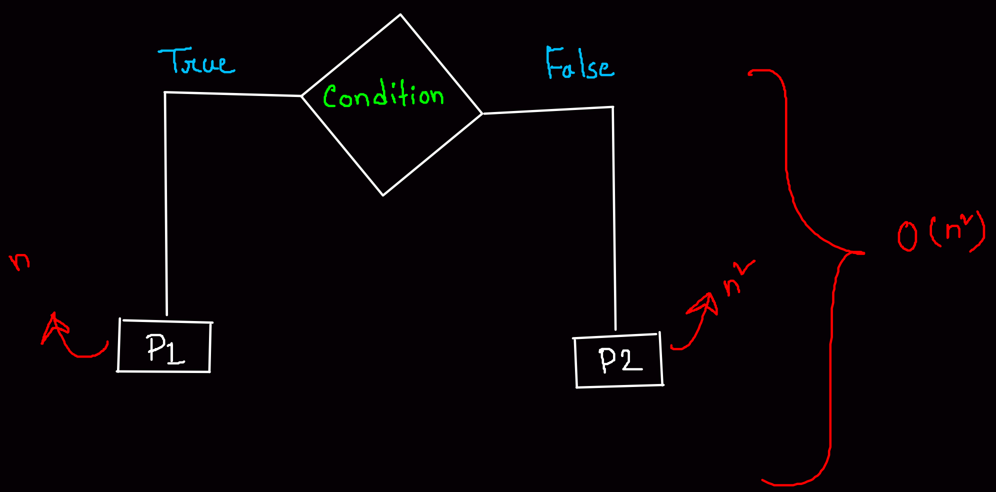
4. Consecutive Statement :

$\boxed{P1} \rightarrow n$

$\boxed{P2} \rightarrow n^2$ $\quad \therefore n^2 + n$

$O(n^2)$

5. if else -



6. While Loop -

```
while (n>0)
{
    i <- i+1
    n <- n/2
}
```
$O(\log n)$

```
while (n>0)
{
    i <- i+1
    n <- n-1
}
```
$O(n)$

7. Recursion :

```
fact(n)
    if(n ≤ 1)
        return 1
    else
        return n * fact(n-1)
```
if / return 1 : 1

return n * fact(n-1) : n

$O(n)$

```
fib (n)
    if n < 1
        return 1
    else
        return fib(n-1) + fib(n-2)
```
T(n) : fib (n)

if / return 1 : 1

$\underbrace{fib(n-1)}_{T(n-1)} + \underbrace{fib(n-2)}_{T(n-2)}$

$\therefore T_n = T_{n-2} + T_{n-1} \cancel{+1} \cancel{+1}$

$= T_{n-1} + T_{n-2}$

Some well-known searching and sorting algorithms :

(a) Linear search : O(n)

(b) Binary Search : O(log n)

(c) Bubble sort : O(n2)

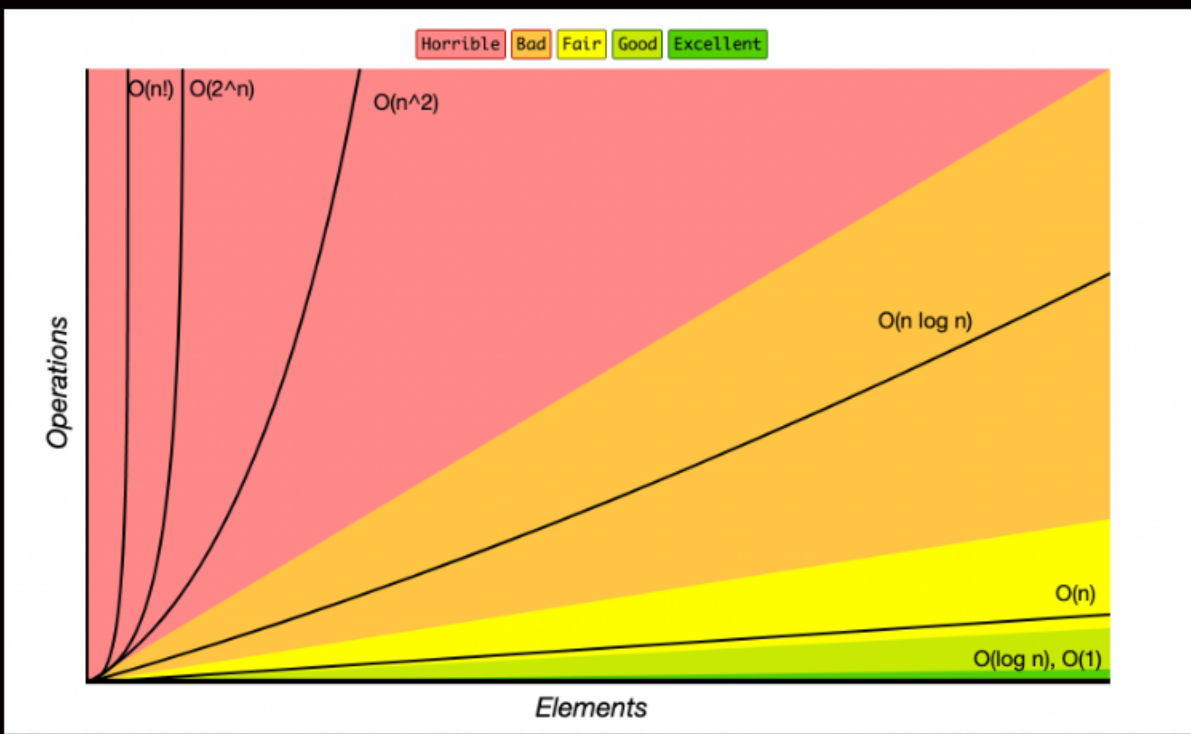
(d) Merge-sort : O(n log n)

Growth of Function:

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

Asymptotic Notation : Asymptotic notations are mathematical tool to represent complexity in terms of time and space .

1. Big - Oh(O) - - - Upper Bound
2. Big Omega(Ω) - - - Lower Bound
3. Theta (θ) - - - Average Bound
4. Small - oh (o) - - -
5. Small - omega (ω) - - -



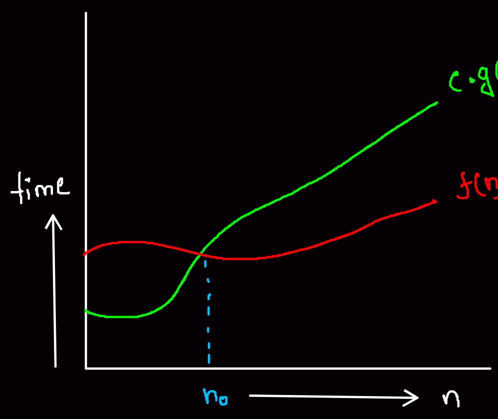
Big - Oh Notation :

Big - Oh notation is used to describe asymptotic upper bound.

Mathematically,

A function $f(n)$ is said to be $O(g(n))$ iff there exist constant c & n_0 such that,

$$0 \leq f(n) \leq c \cdot g(n) \quad \text{for all, } n \geq n_0$$
$(\forall)$



Big - Omega Notation :

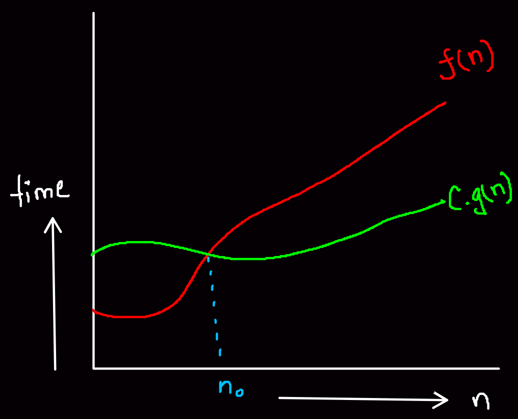
Big - Omega notation provides aysmptotic lower bound.

Mathematically,

A function $f(n)$ is said to be $\Omega(g(n))$ iff there exists constant c & n_0

such that,

$$0 \leq c g(n) \leq f(n) \quad \forall \; n \geq n_0$$



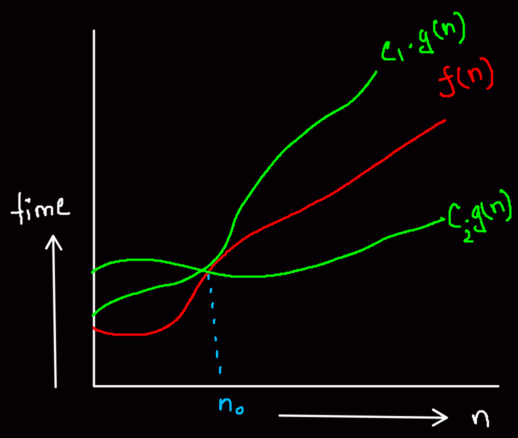
Big - Theta Notation :

Big - Theta notation is used when the function f (n) is bounded both from above and below by the function g(n) .

Mathematically ,

A function $f(n)$ is said to be $\Theta(g(n))$ iff $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$ and there excists constant c_1, c_2 & n_0 such that,

$$\left.\begin{array}{l} 0 \leq f(n) \leq c_1 g(n) \\ 0 \leq c_2 g(n) \leq f(n) \end{array}\right\} \quad \forall\ n \geq n_0$$



Meaarging the both equation we got,

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

The equation simply means there exist positive constants c_1 & c_2 such that $f(n)$ is sanwitched between $c_2 \cdot g(n)$ and $c_1 \cdot g(n)$.