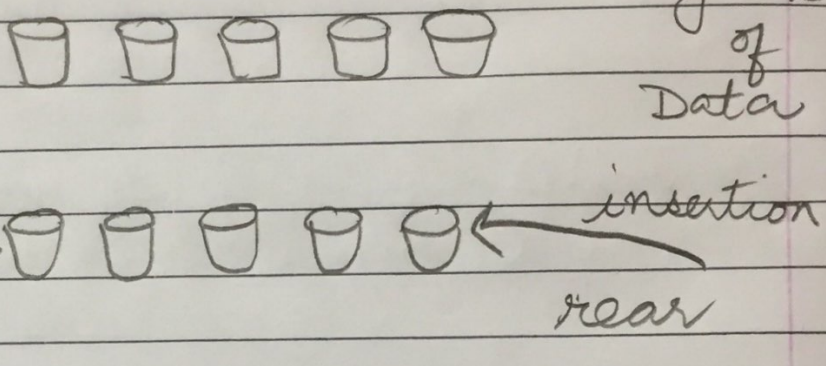
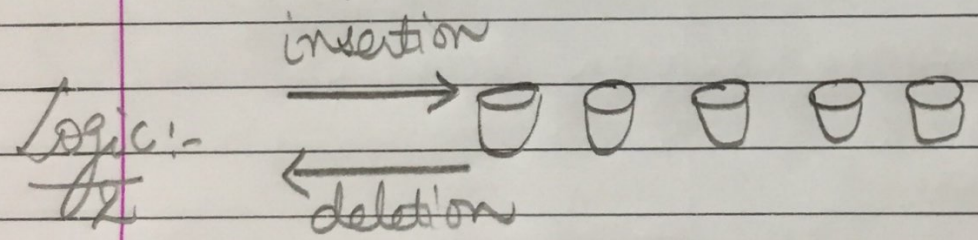
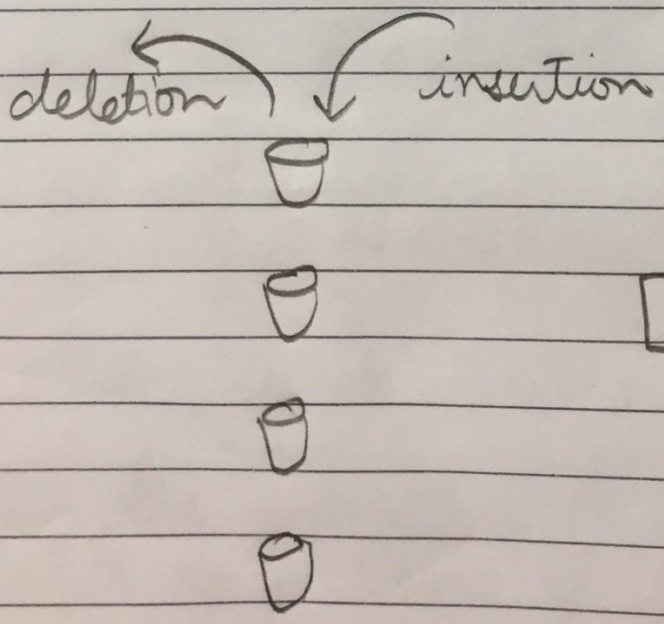
Data Structures

- Organizing data in a particular logic.
- Applying a logic :— on the organization of data
- like :-



[Queue]



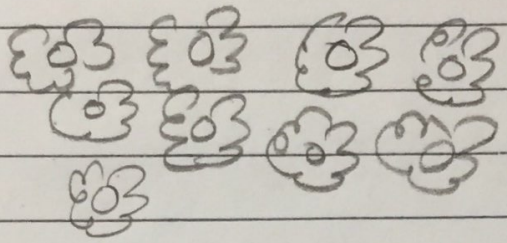


[Stack]

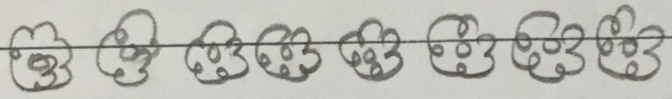
Data structure :—

The logical or mathematical model of a particular organization of data is called a data structure.

Data :—

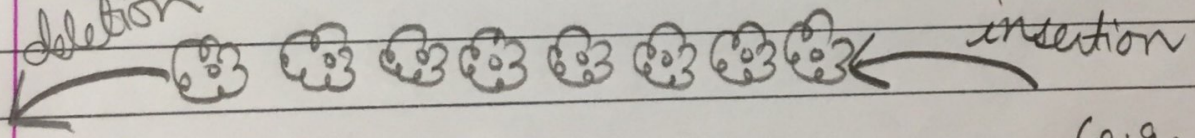


Organization of Data :— like :—



[Array (List)]

Applying the logical model :— like :—



(e.g. Queue)

Applying the logical model

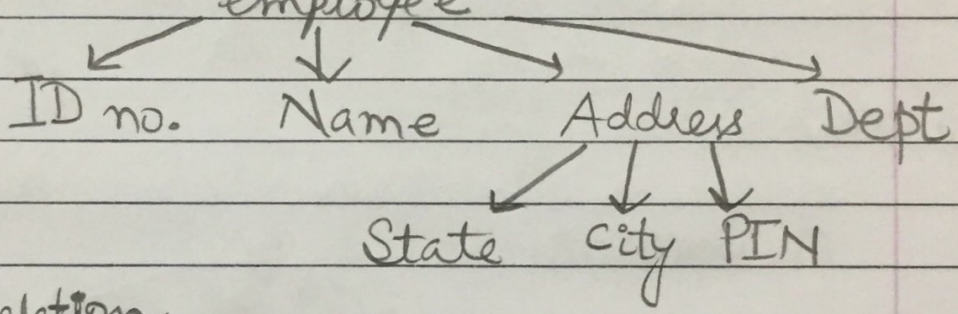
means → special rules for operations like insertion, deletion

Data Structures

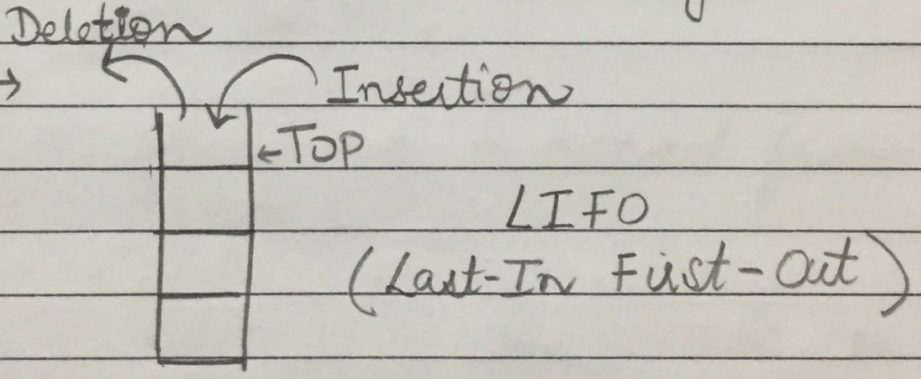
* logical/mathematical model of a particular organization of data.

* Various Data Structures :-

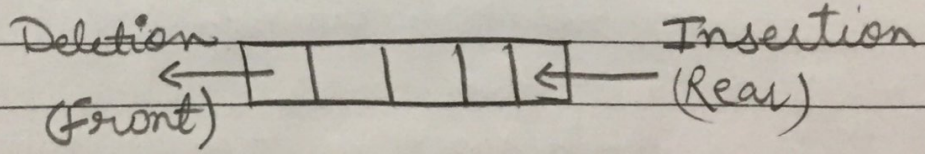
- Arrays → lists
- Linked Lists → node → info, pointer
- Trees → hierarchical relationship among data



- Stack →

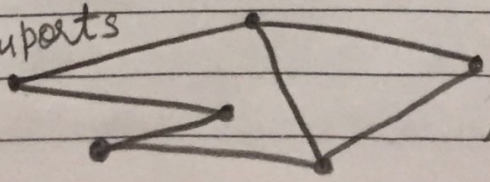


- Queue →



FIFO
(First-In Fust-Out)

- Graph →



Relationship b/w pairs (not hierarchical)

Data Structures

✓ Definition of Data Structures
✓ Various Data Structures
- arrays, linked lists, trees, graphs, stack (LIFO), queue (FIFO), etc.

✳ **Operations on Data Structures :—**

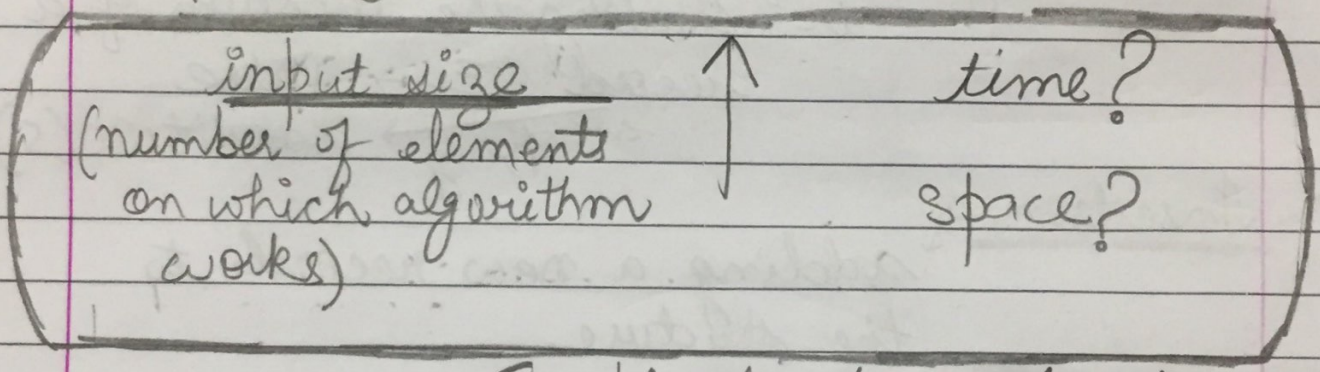
- **Traversing** → accessing each record
- **Searching** → finding a record
  i.e. finding the location of a record = → value satisfying → condition(s)
- **Insertion** → adding a new record to the structure.
- **Deletion** → removing a record from the structure
- **Sorting** → arranging the records in some logical order
  - alphabetical order
  - ascending order
  - descending order
- **Merging** → Combining the records in two different sorted files into a single sorted file.

Algorithms

* well-defined list of steps for solving a particular problem.
* efficiency → time
  → space

* **Complexity of an Algorithm**

The function which gives the running time and/or space in terms of the input size.



Complexity of an algorithm

* **Time - Space Tradeoff**

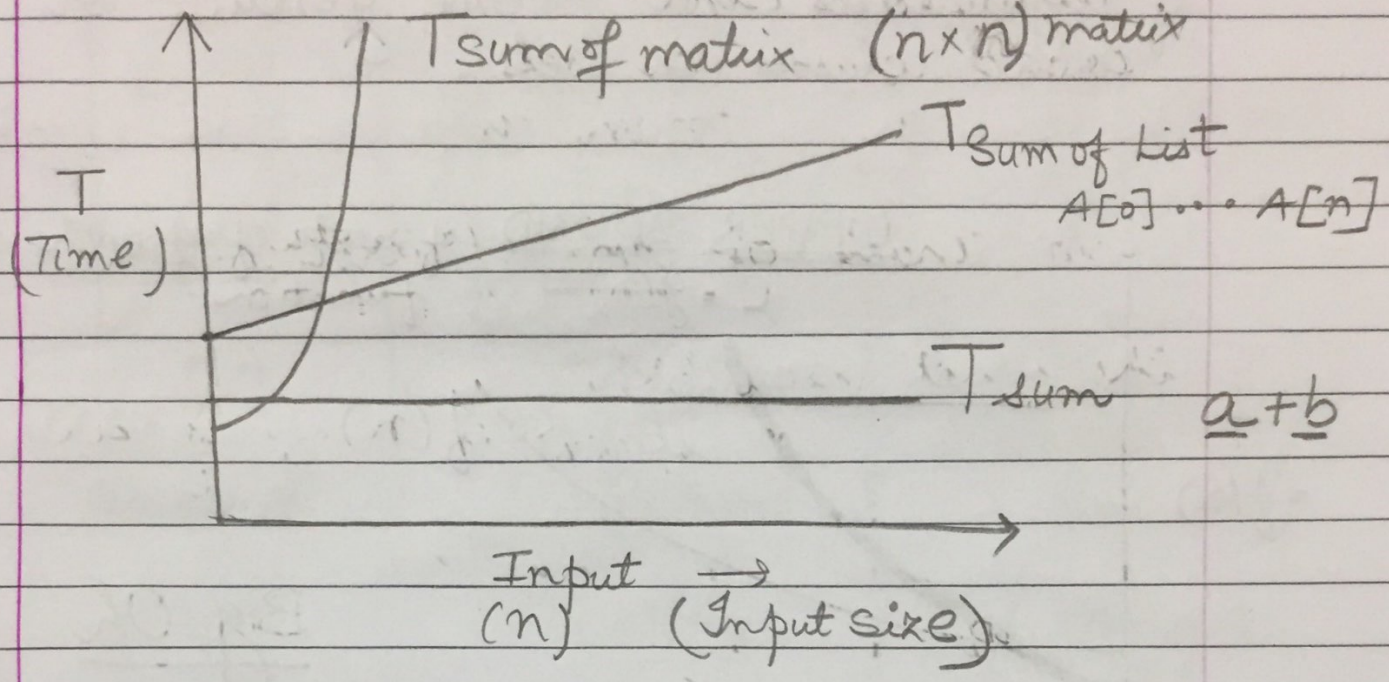
- by increasing the amount of space for storing the data, one may be able to reduce the time needed for processing the data.

space ↑ , time ↓

or

space ↓ , time ↑

Time Complexity → Rate of growth of time taken with respect to the input



Time? $n \uparrow \Rightarrow$ time-complexity

T_{sum} ⟶ (constant)

$T_{sum\ of\ list}$ ↗ (linear)

$T_{sum\ of\ Matrix}$ ↗ (quadratic)

* Broader view to study time complexity

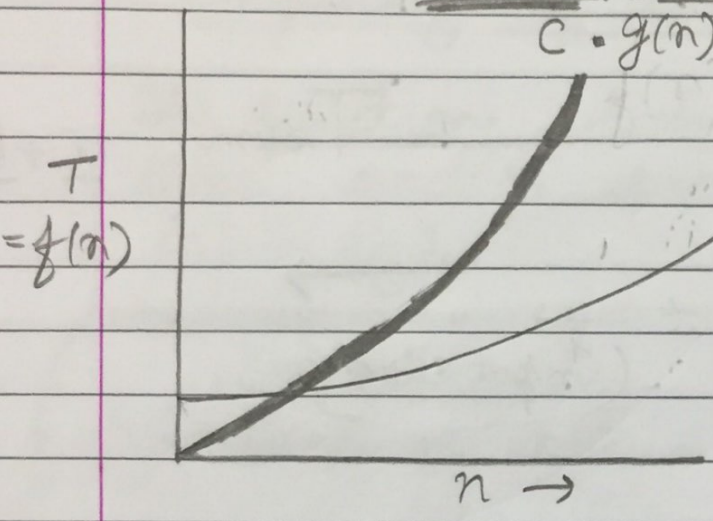
* $T = f(n)$ $n \rightarrow$ input size

? category / Behavior

* Asymptotic Bounds / Notations

Asymptotic Notation

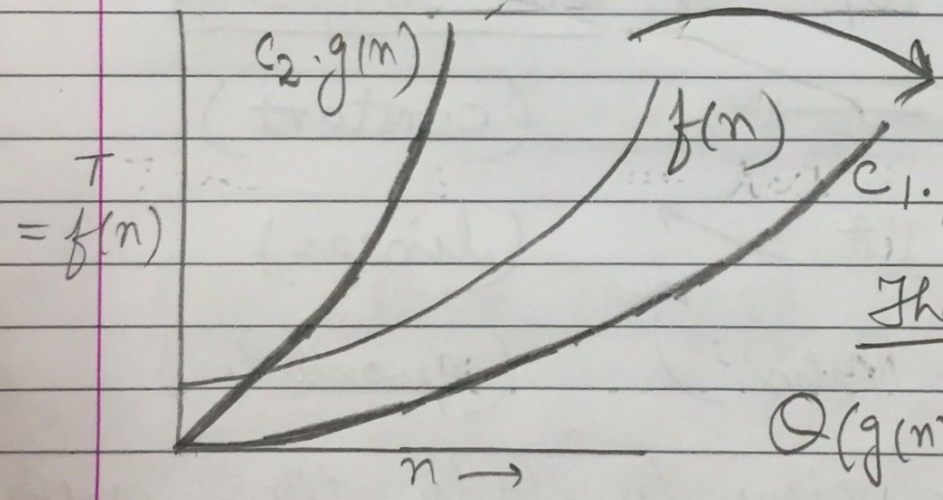
* is a way to classify the running time of algorithms into some generic or broad classes or sets.



UPPER BOUND of rate of growth of a function.

Big Oh 'O'

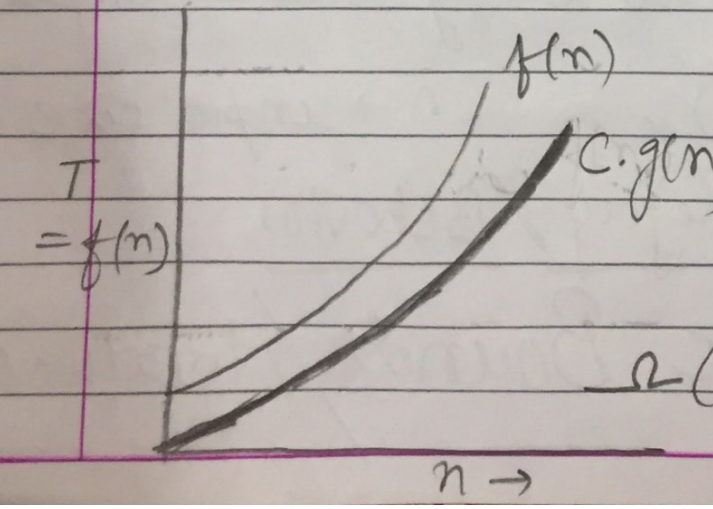
$O(g(n)) = \{ f(n) \leq c \cdot g(n) \}$



TIGHT BOUND

Thota Θ

$\Theta(g(n)) = \{ c_1 g(n) \leq f(n) \leq c_2 \cdot g(n) \}$

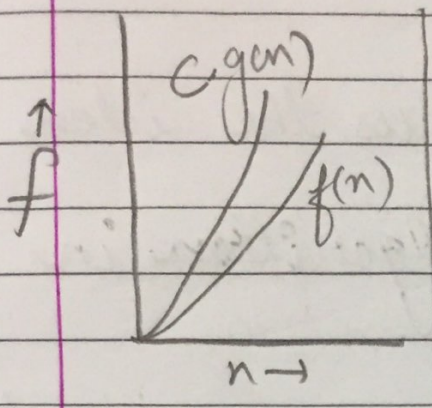


LOWER BOUND

Omega Ω

$\Omega(g(n)) = \{ c \cdot g(n) \leq f(n) \}$

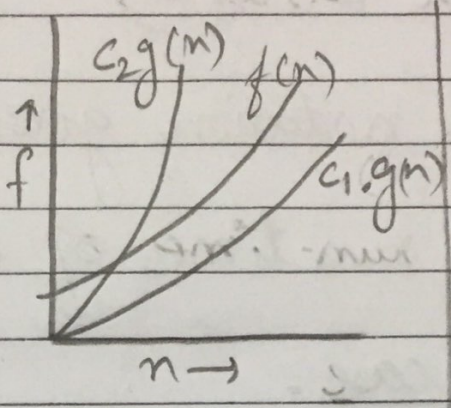
Asymptotic Notations



Upper Bound

Big oh

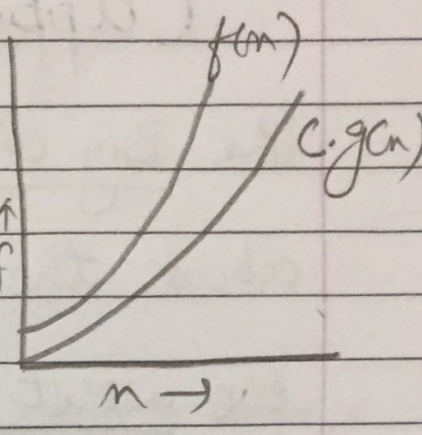
O



Tight Bound

Theta

Θ



Lower Bound

Omega

Ω

Worst case

(max?)

Predicted Behavior

(tight bound)

Best case

(min?)

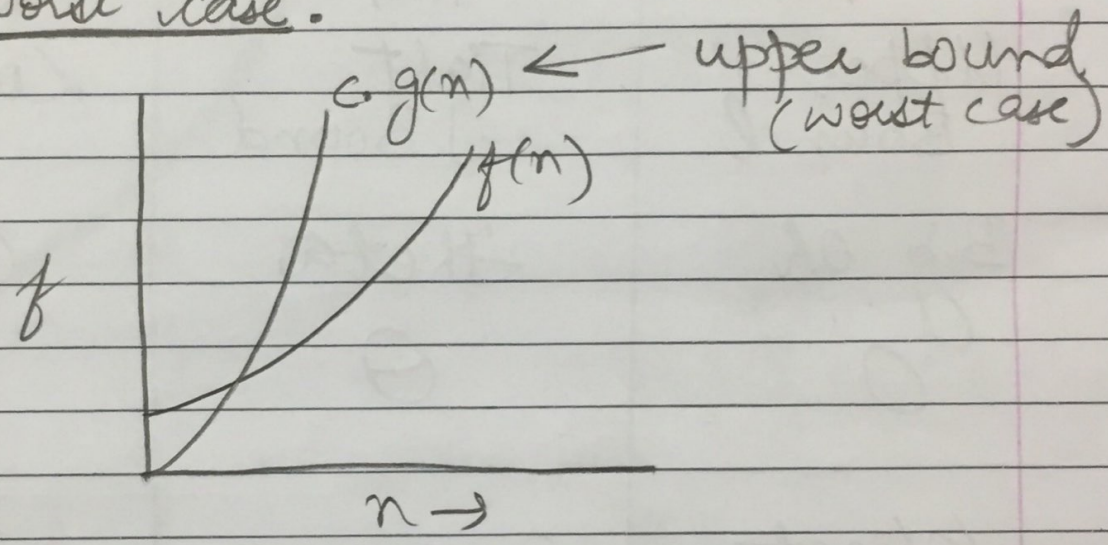
is studied to choose an algorithm

Big Oh (Upper Bound) } WORST CASE?

Big Oh Notation

(Upper Bound)

The Big Oh notation gives us the idea about the run-time of algorithm in the worst case.



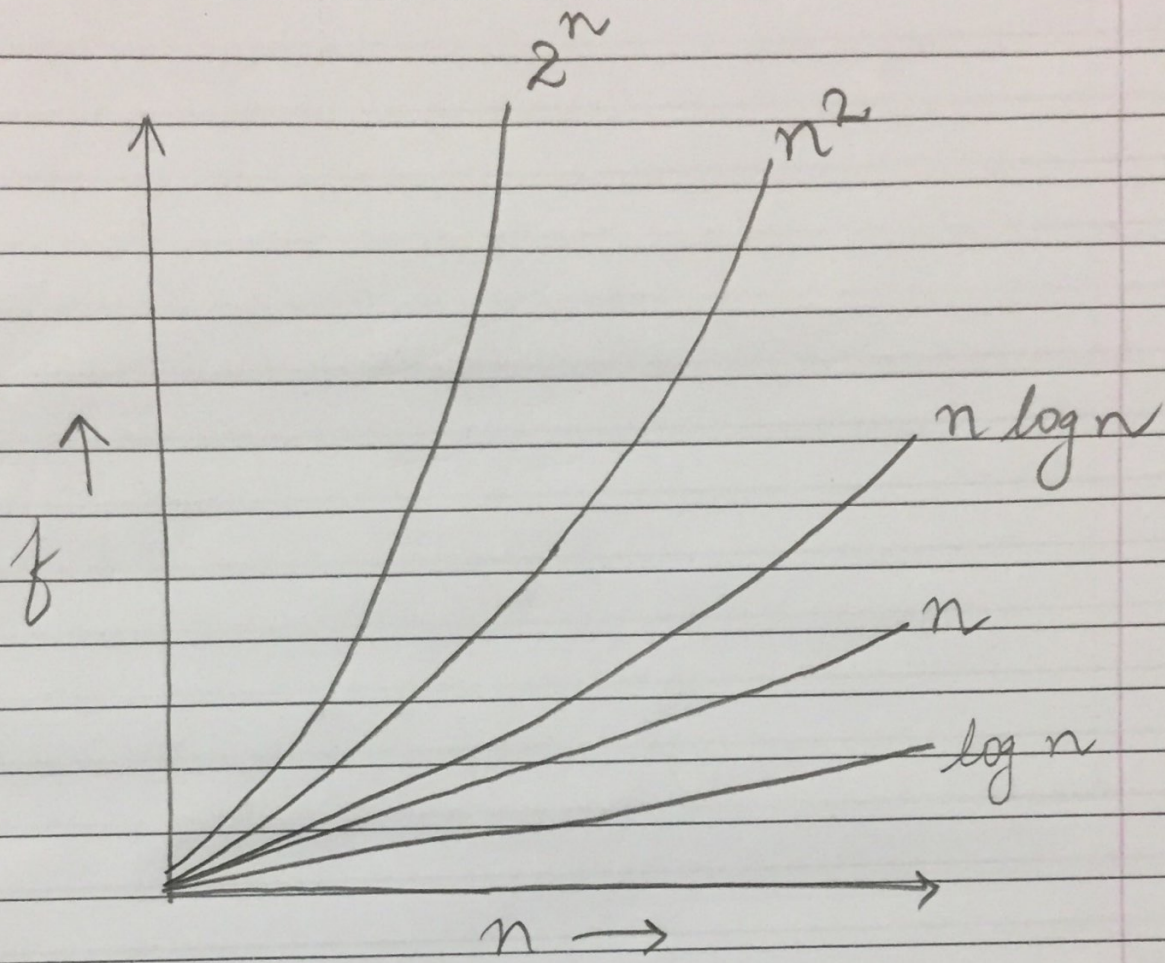
Time complexity = rate of growth of time with input size

Upper Bound (Big Oh):-

Rate of growth of f(n) will never be higher than c. g(n)

i.e. the worst case

Plot of functions



Complexity:-

$\log n < n < n\log n < n^2 < 2^n$

exponential (algorithms of exponential complexity are not feasible)

polynomial

⇒ The algorithms of small complexity such as $\log n$, n, $n\log n$, n^2, n^3, etc. are feasible.