# Time & Space complexity

→ to compare 2 algorithm
→ doesn't depend on time run [not standarilized form]

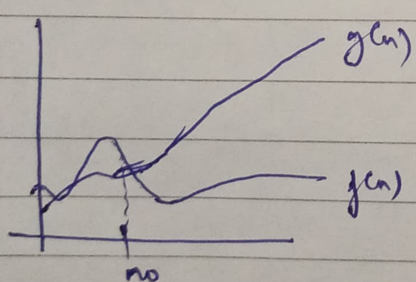
so, asymptotic notations are used.

→ Big O [worst time complexity]
→ omega
→ theta

## Big O

n → input size
f(n) → function
g(n) → another function [after no g(n) > f(n)]

∴ we say g(n) is giving Big O notation




Linear search: for loop till finding the desired object

O(n)
↓
worst case scenario.

upper bound of an algo.

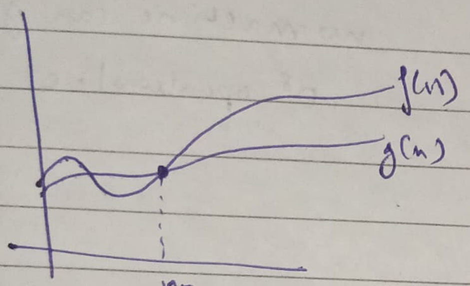
$f(n) = an^2 + bn^4 + c$ $\quad$ $f(n) = \frac{an}{4} + bn$ $\quad$ $f(n) = 100$

$O(n^4)$ $\quad$ $O(n)$ $\quad$ $O(1)$

Time complexity evaluation

- ignore constant
- Biggest degree should be concerned.

## Omega Notation $\Omega$




- lower bound.
- searching of element in array $\Omega(1)$

$f(n) = an^2 + bn$

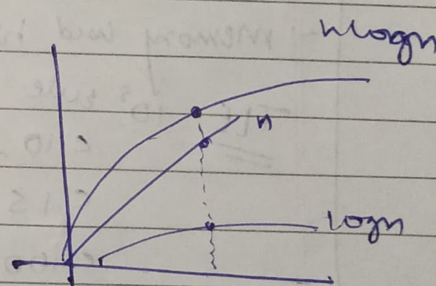
$\downarrow$

$\Omega(n)$

$f(n) = n\log n + bn$

$n\log n > n$

$\Omega(n) = n$




- find smallest degree

## Metha notation ($\Theta$)

- exact bound
- average case complexity of algorithm

average $= \frac{O}{2} = \Theta$

$\Theta(n)$

insertion sort : $O(n^2)$ $\Omega(n)$ $\Theta(k)$

Time limit exceeded TLE

→ above $10^8$ u can't go.

→ $n < 10^4$ → $O(n^2)$
→ $n \le 10^6$ → $O(n \log n)$
→ $n \le 10^8$ → $O(n)$
→ $n > 10^8$ → $O(\log n)$ or $O(1)$

→ $n \le 12$ → $O(n!)$
→ $n \le 25$ → $O(2^n)$
→ $n \le 100$ → $O(n^4)$
→ $n \le 500$ → $O(n^3)$

no machine can go above $10^8$ operations/sec

Recurrence Problems

Space complexity

→ memory used to solve.

TLE $10^8$ rule

| | |
|---|---|
| $< 10 \ldots 11$ | $O(n!)$ or $O(n^6)$ |
| $< 15 \ldots 18$ | $O(2^n * n^2)$ |
| $< 100$ | $O(n^4)$ |
| $< 400$ | $O(n^3)$ |
| $< 2000$ | $O(n^2 * \log n)$ |
| $< 10^4$ | $O(n^2)$ |
| $< 10^6$ | $O(n \log n)$ |
| $< 10^8$ | $O(n)$, $O(\log n)$ |

if constraint says $10^4$ my max T.C. should be $O(n^2)$ can use it any of the above them but cannot $O(n \log n)$, $O(n)$, $O(\log n)$

Space Complexity

count of variable doesn't matter