Analysis Of algorithms — Part -1

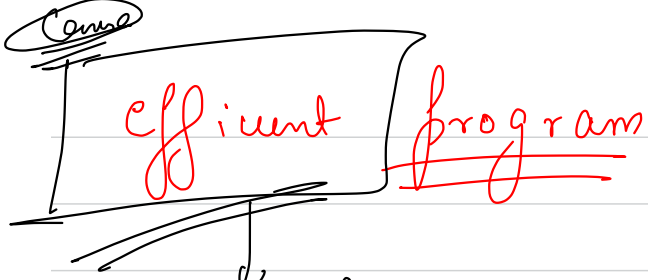
→ Computation → Hardware config

- Processor
- Ram
- SSD / HDD
- CPU
- ⋮

we want the systems to be as fast as possible

No

You want your software to be as efficient as possible.



In what sense

- in terms of time → less time
- in terms of space → less memory space

PUBG

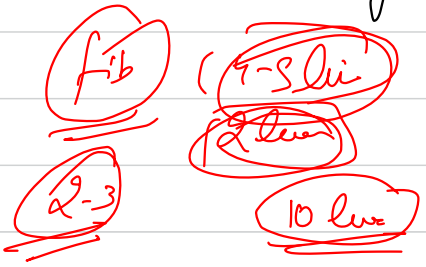
- how much memory this game will take
- heat up ↑
- throttle ↑
- lag while loading
- Bad ✗
- 3 GB
- your plays offline with no bots
- lag
- 6 GB

The game algorithms are also taking lot of time

How to analyse an algorithm ??

↳ Can we compare algorithms based on no. of line of code ??

NO



↳ Can we check the execution time of the algorithm??

↳ NO

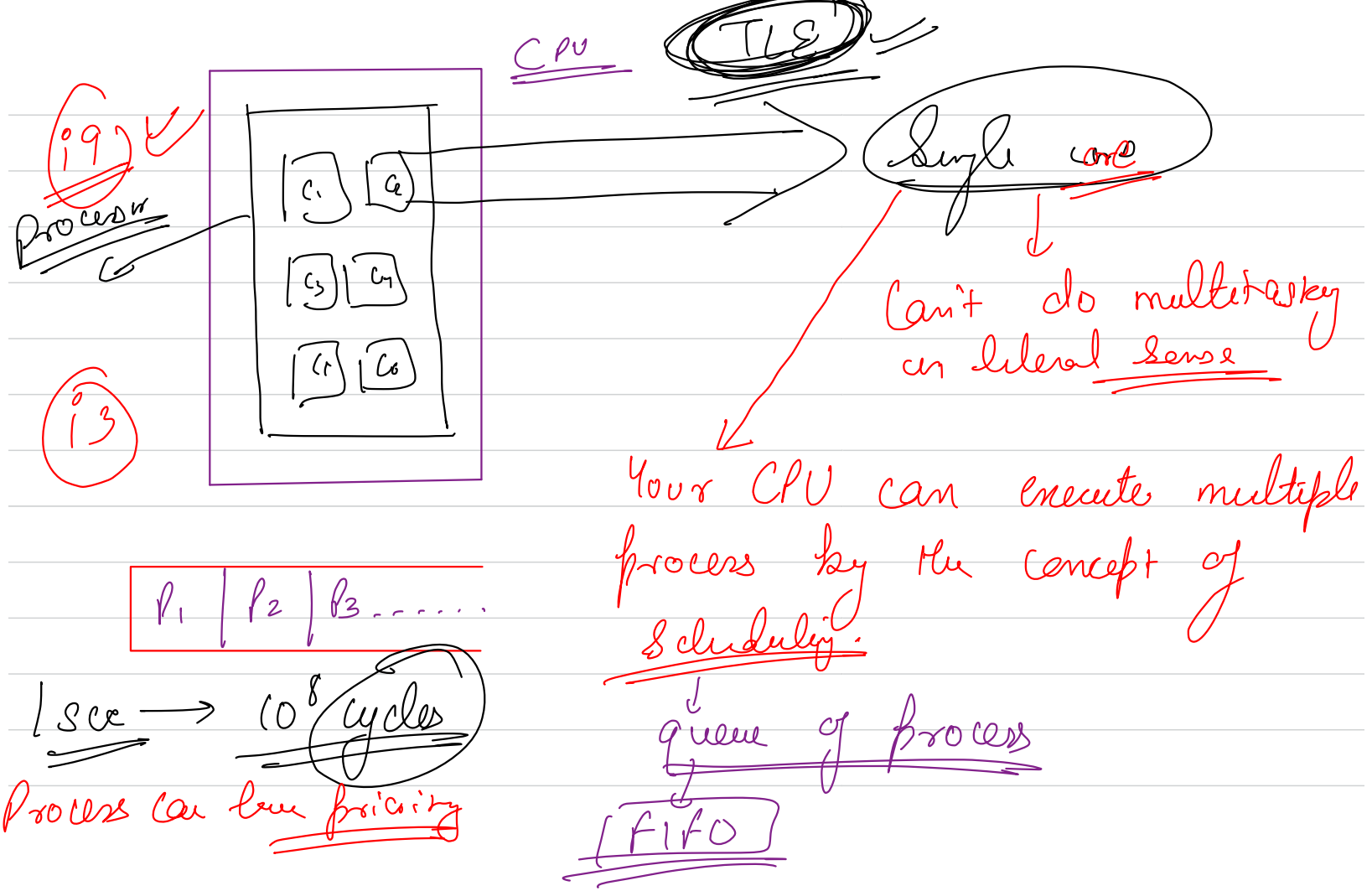
$t = t_2$ $t = t_1$

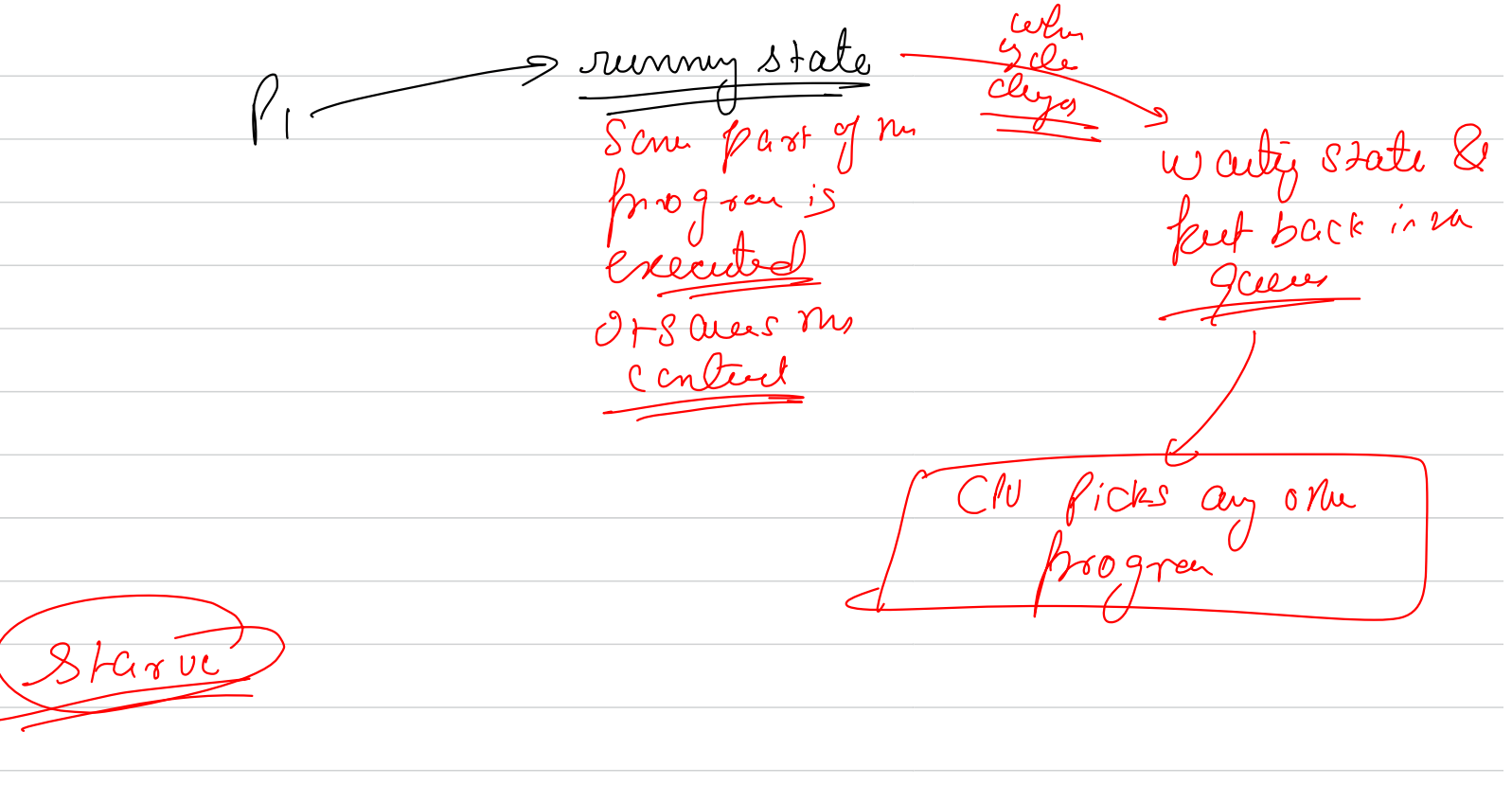
end time − start time ⇒ time executed $(t_2 - t_1)$

→ Experimental Analysis

→ Why was the time of execution change without even changing the input or code ??

Multitasking ??


CPU
TLB
i9
Processor
C1
C2
C3
C4
C5
C6
i3
Single core
Can't do multitasking in literal sense
Your CPU can execute multiple process by the concept of scheduling.
queue of process
FIFO
P1 | P2 | P3 .....
1 sec → 10^8 cycles
Process can have priority


P1
running state
when cycle elapse
Some part of the program is executed & saves its context
waiting state & put back in the queue
CPU picks another program
Starve

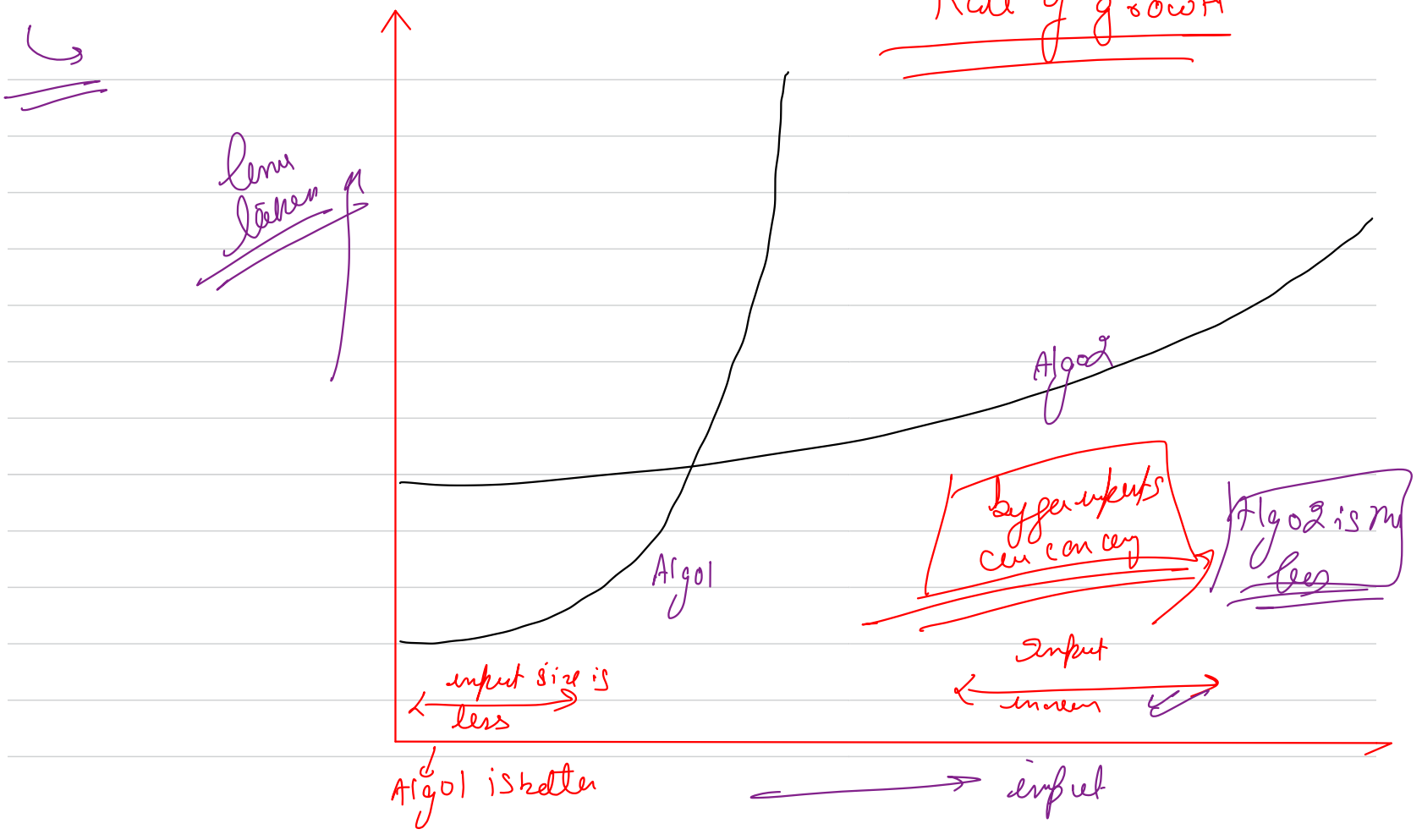
That's why we can't do experimental analysis.

1 sec → 10^8 cycles

10 sec → 10^9 cycles

2 sec → 2×10^8 cycles

⋮


Rate of growth
less faster
Algo2
Algo1
bigger inputs can concay
Algo2 is my fav
input size is less
Algo1 is better
input increases
input

What is Rate of growth ??

The rate at which running time of algo increases as a function of input.

Assume,

$f(n) = n^2$

$g(n) = 10n$

$n = 10$

$f(n) = 100$

$g(n) = 100$

$n = 20$

$f(n) \Rightarrow 400$

$g(n) = 200$

$=$

$n = 30$

$f(n) = 900$

$g(n) = 300$

Rate of growth of a curve depends just on the highest degree term. All the lower degree term for bigger input could be insignificant

↳ So if we want to analyse algos, for bigger input, we need to extract the highest degree term from the polynomial curve representing time vs input of algo

If algo-x → $f(n) = 2n^2 + 3n$ → n^2

n → input size

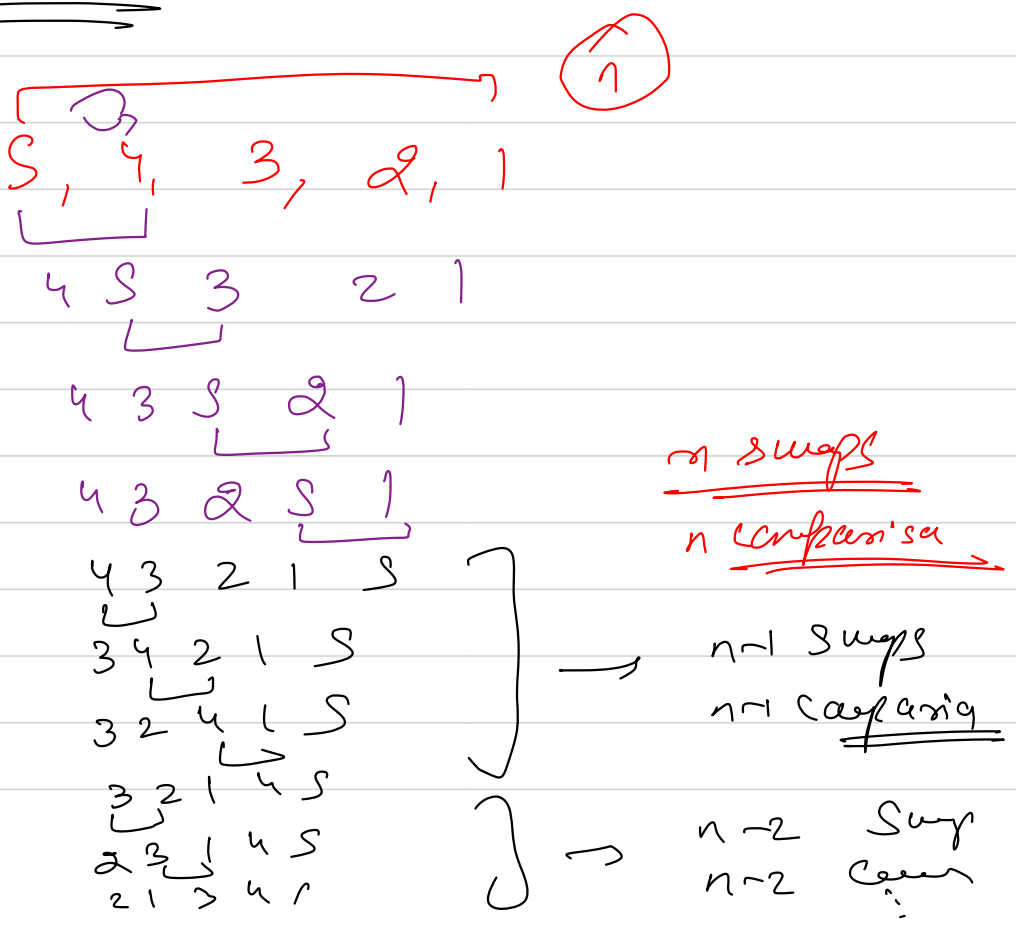
algo-y → $f(n) = 3n^2 - 2n + 3$ → n^2

runtime will be same for algo x & algo y

This type of analysis where we compare rate of growth w.r.t input is called

ASYMPTOTIC ANALYSIS

Bubble Sort


1
5, 4, 3, 2, 1
4 5 3 2 1
4 3 5 2 1
4 3 2 5 1
4 3 2 1 5
3 4 2 1 5
3 2 4 1 5
3 2 1 4 5
2 3 1 4 5
2 1 3 4 5
n swaps
n comparisons
n-1 swaps
n-1 comparisons
n-2 swaps
n-2 comparisons

$n + n + (n-1) + (n-1) + (n-2) + (n-2) - - - - - - - - 1$

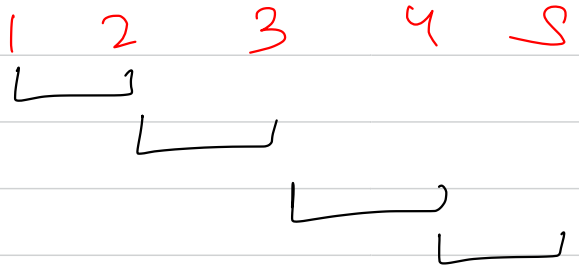
$2(n + n-1 + n-2 - - - - - - - 1)$

$\frac{n(n+1)}{2} \rightarrow n^2 + n$

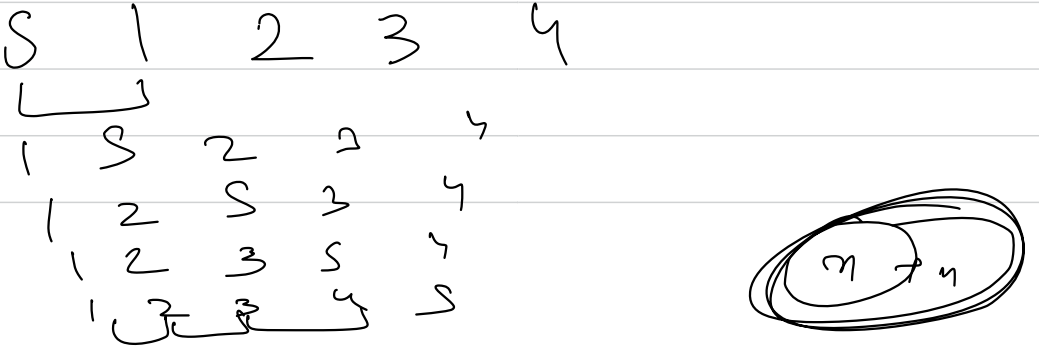
$n \rightarrow$ size of list

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

→ Rate of growth depends on n^2



$f(n) = n$ → Rate of growth is n



Based on type of input the rate of growth can change:

3 types of analysis →

Worst Case
Of defines i/p for which algo takes long time
Here algo is the slowest

Avg Case
Predicts an approximation for any random input

Best Case
Here the algo is fastest & takes least run time

3 notations → general mumbo jumbo

Worst → Big O → $O(n^2)$

Avg → Big theta → $\Theta(n^2)$

Best → Big omega → $\Omega(n)$

Rate of growth

2^{2^n} → very bad

$n!$

2^n

n^3

n^2

$n \log n$

n

$\sqrt{n}$

$\log n$

$\log \log n$

constant → The Best

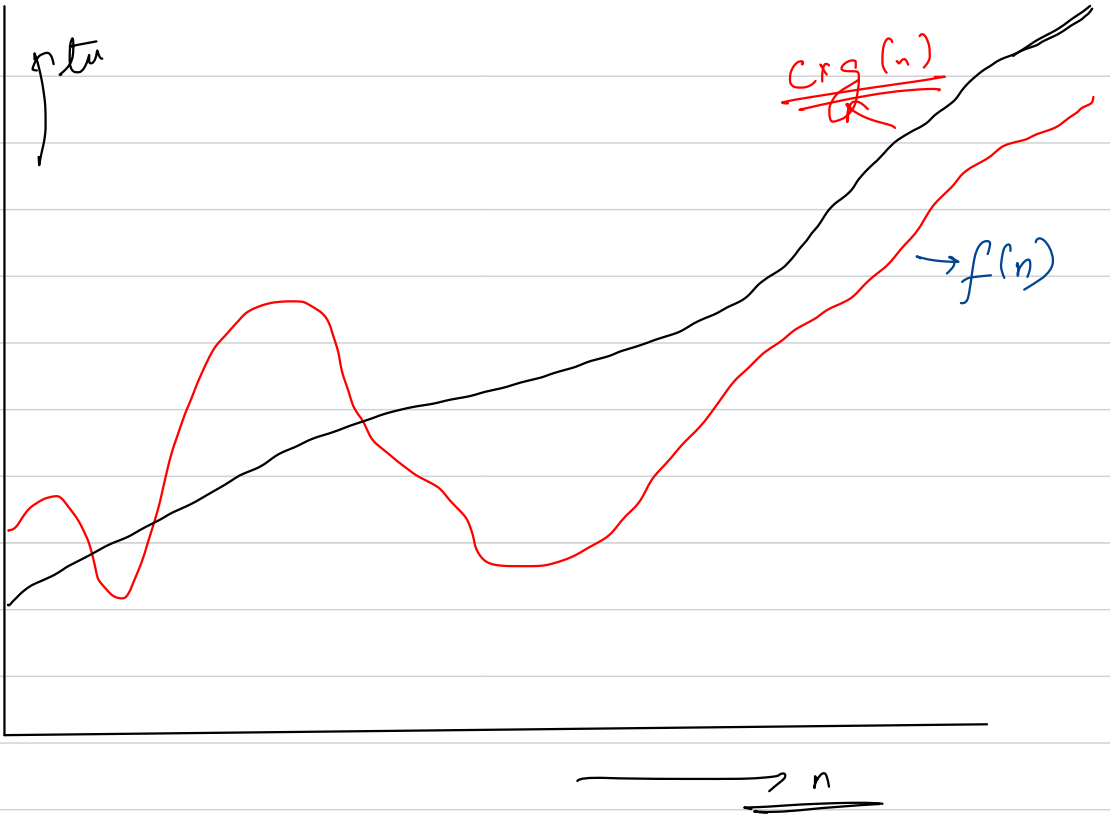
$f(n) = O$

Big O → highest upper bound

$f(n) = n^4 + 100n^2 + 10n + 50$

$g(n) = n^4$ larger n

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



Big omega

tightest lower bound.

$f(n) = n^4$

$g(n) = n^4$

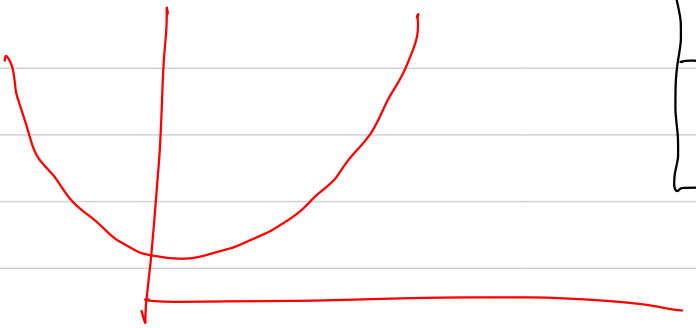
$\Omega(n^4)$

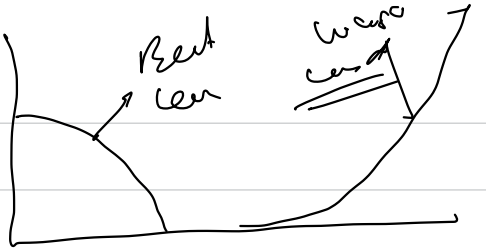
$f(n) = \begin{cases} n^2 \rightarrow n \\ n \end{cases}$

$\rightarrow$ faster

$\rightarrow$ fewer

$\Omega(n)$





$f(n) = n^2$

small n → n^2

Big n → n^2