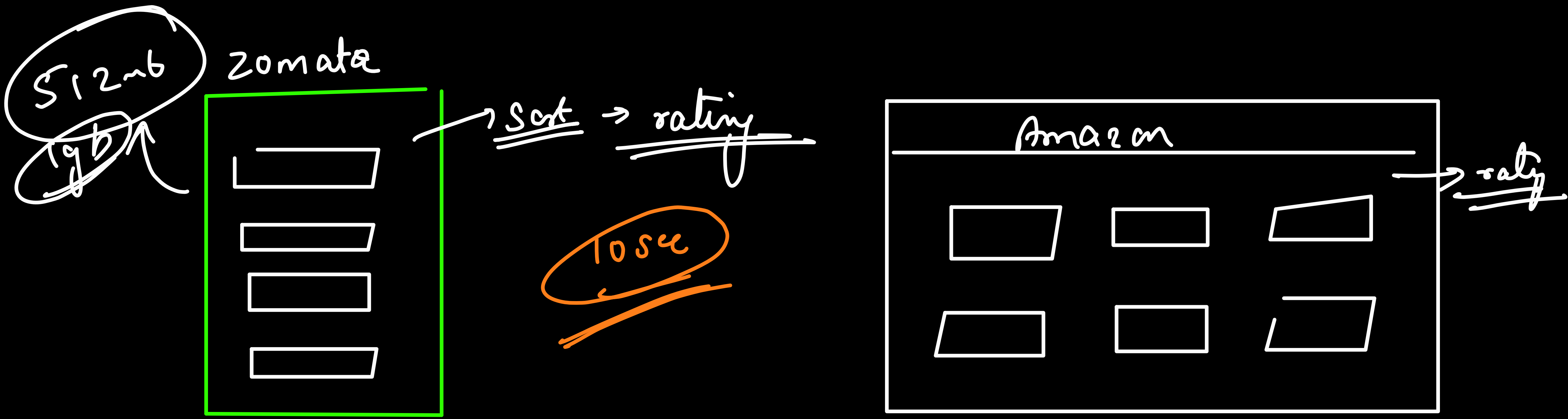
Time And Space Complexity Analysis

Algorithm Complexity Analysis



1) Efficiency of an algorithm in terms of time taken is imp.
2) Efficiency of an algorithm in terms of space/memory taken is imp.

→ CPU ✓

→ SSD / HDD

→ RAM ✓

→ GPU

⋮

Smooth

Experimental Analysis

User 1

User 2

intel
Core 2 Duo
inside

Algo

100 ms

113ms

450ms

380 ms

Algo 2

Multitasking

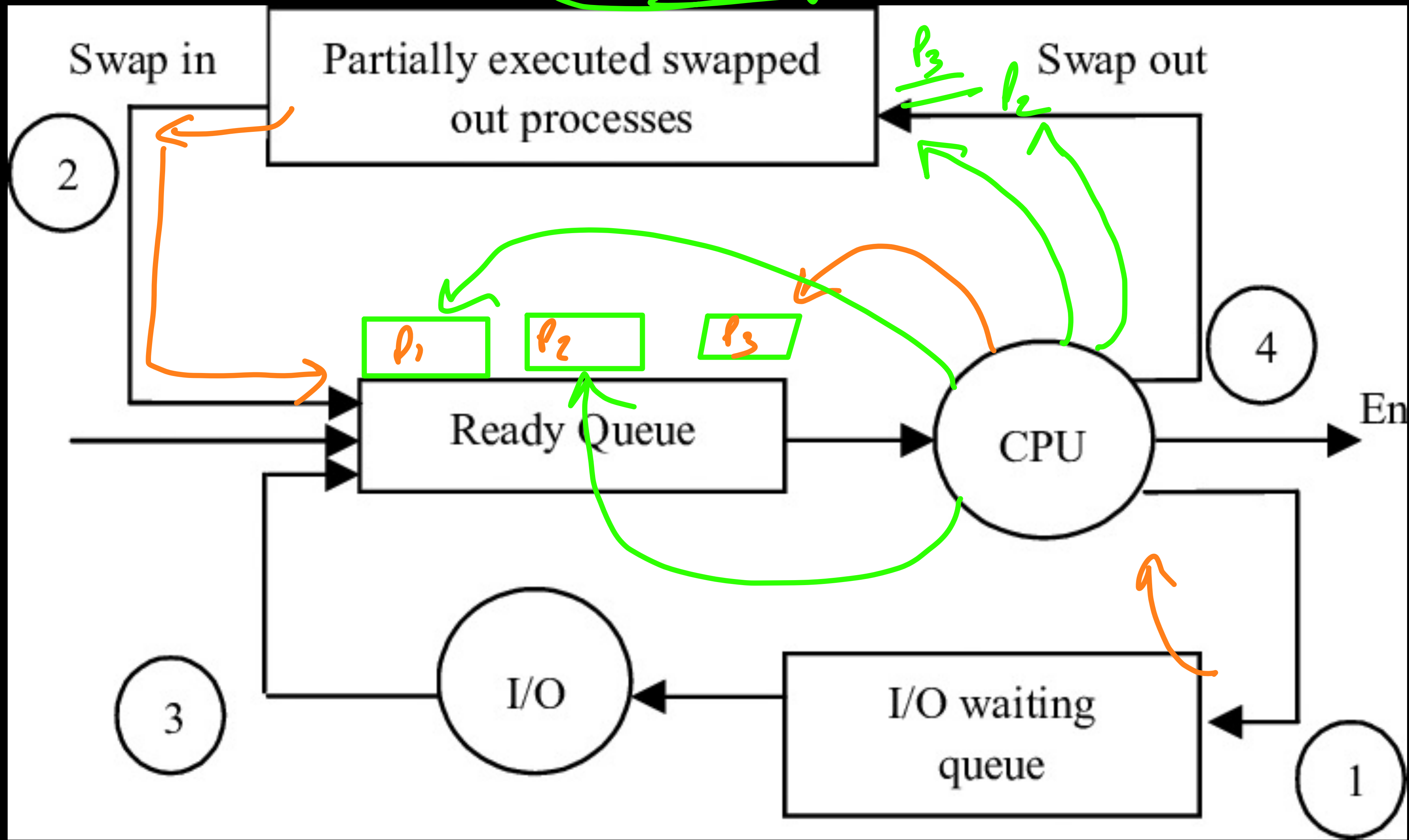
Single core

10 . 48 . 35

2 waiting area

1 sec

10^8 instructions

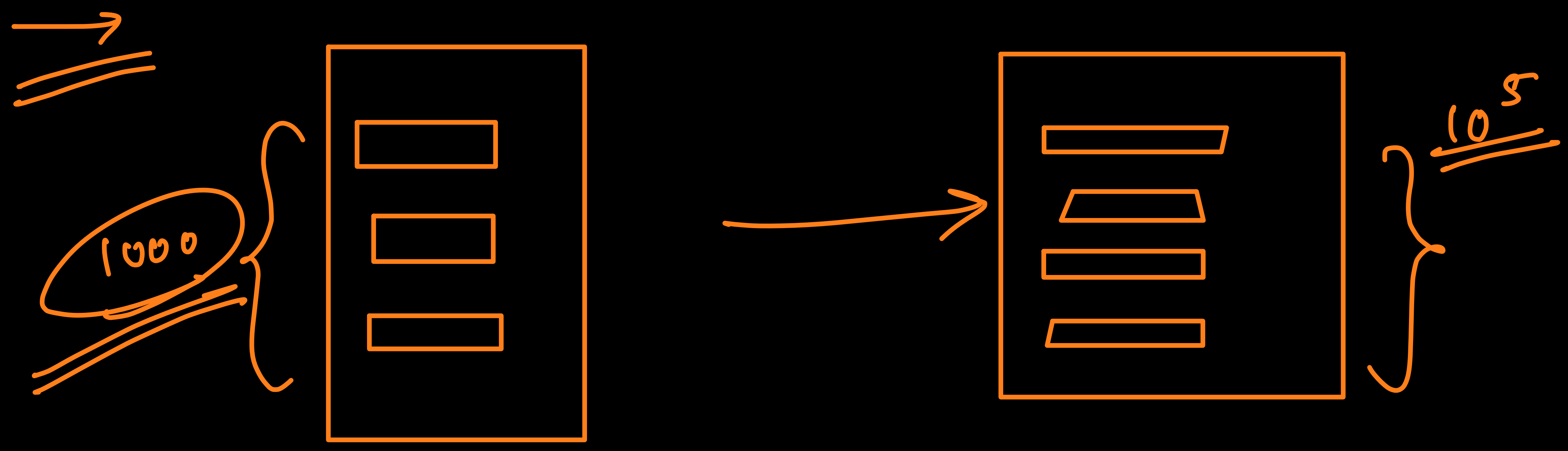


$P_3 \rightarrow$ 1000 ins

300

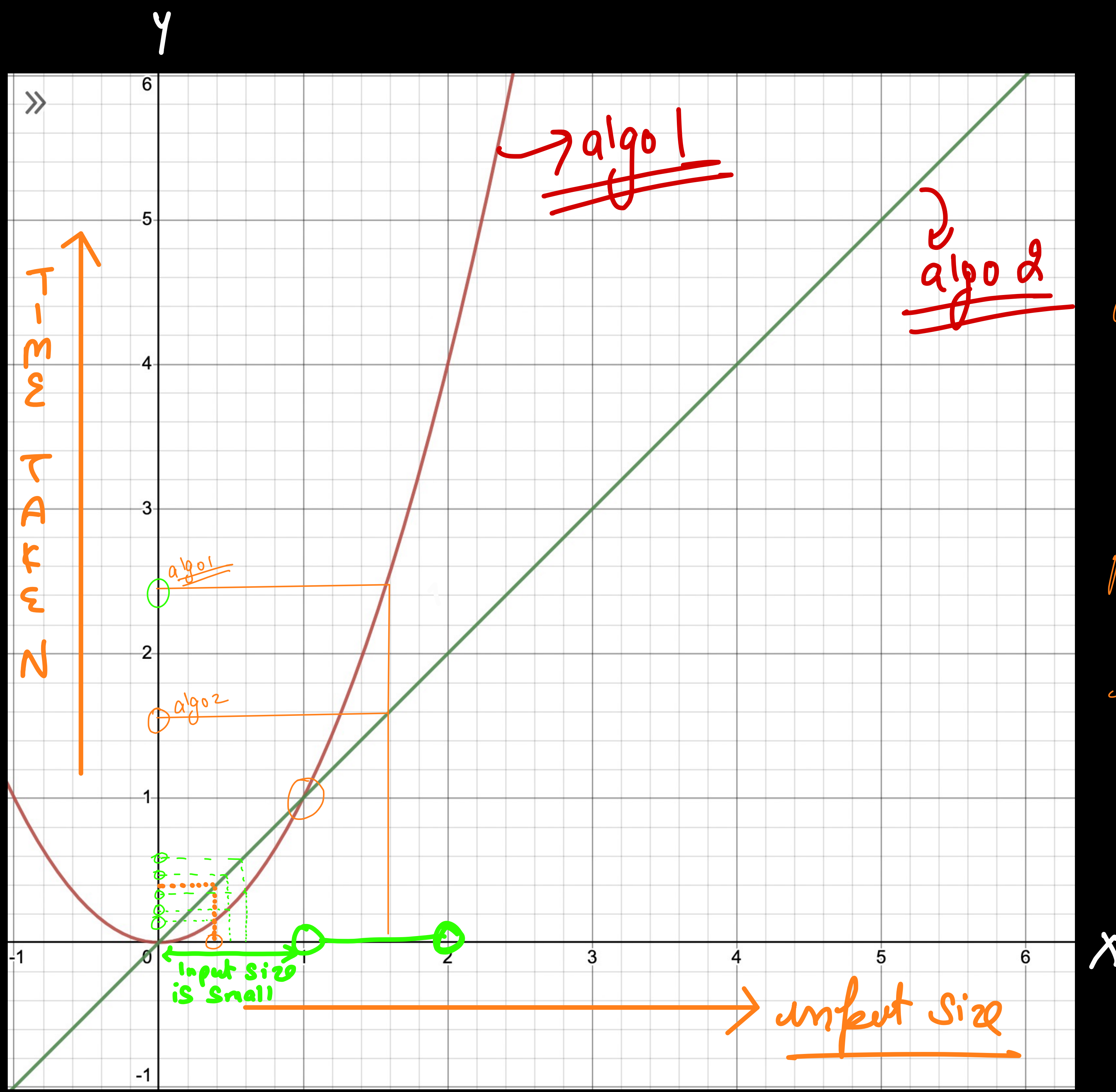
$P_2 \rightarrow$ 5000 ins

800 1000



→ Input (what, size, how input is given) directly affects the course of algo execution.

Rate of growth



always we care about how the algorithms are performing for large input size.

if runtime changes extremely high with a small change in input then growth is high.

	C_1	C_2
1st	100	20
2nd	105	80
3rd	110	140

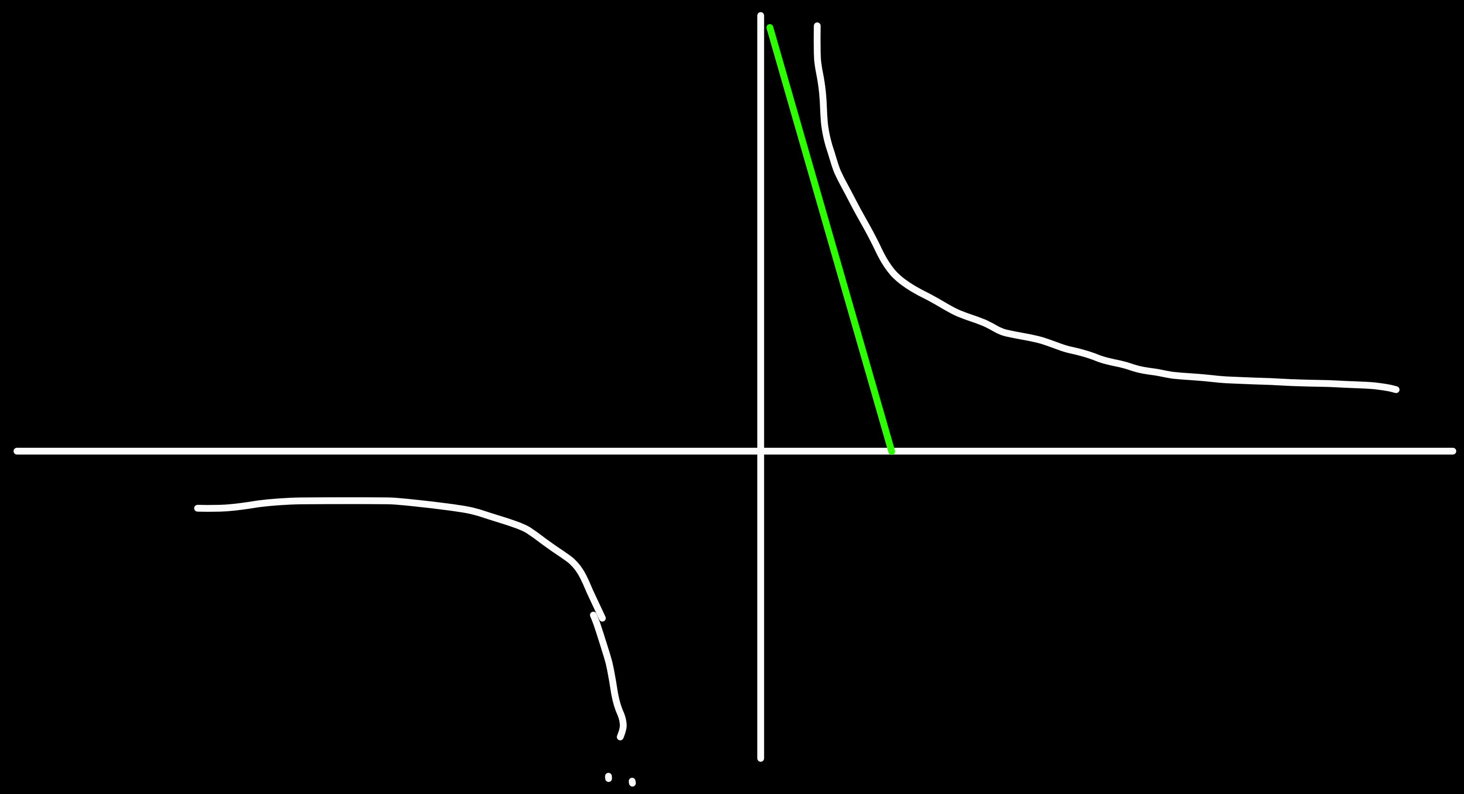
C_2

$$\frac{\Delta t \uparrow}{\Delta i \downarrow}$$

Rate at which running time increases as a func^{n} of input is called Rate of growth.

Asymptotic Analysis

Asymptote → It is a st. line that constantly approaches a given curve but doesn't meet at any ∞ dist.



→ Rate of growth of algo (running time w.r.t input size)

→ Behaviour of the rate at very large input value.

$y = n^2$ $n \to 10$ $n \to 10^5$

$y = 100$

$y = 10^{10}$

$y = n \times \sqrt{n}$ $n \to 100$ $n \to 10^6$

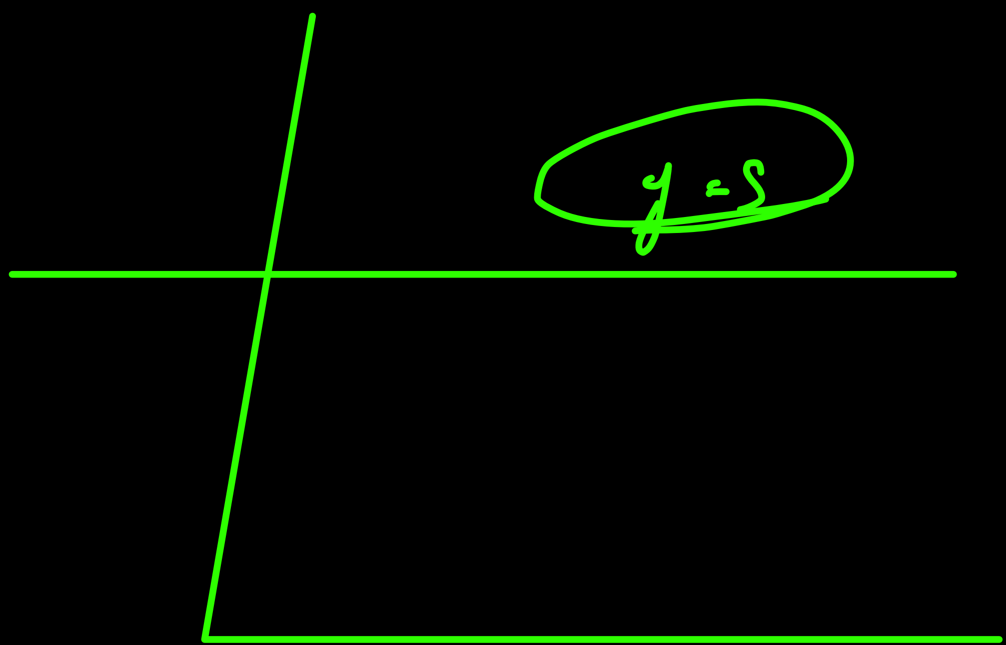
$100 \times 10 \to 1000$

$10^6 \times \sqrt{10^6} \to 10^9$

$y = n \log n$ $\qquad\qquad$ $y = n$

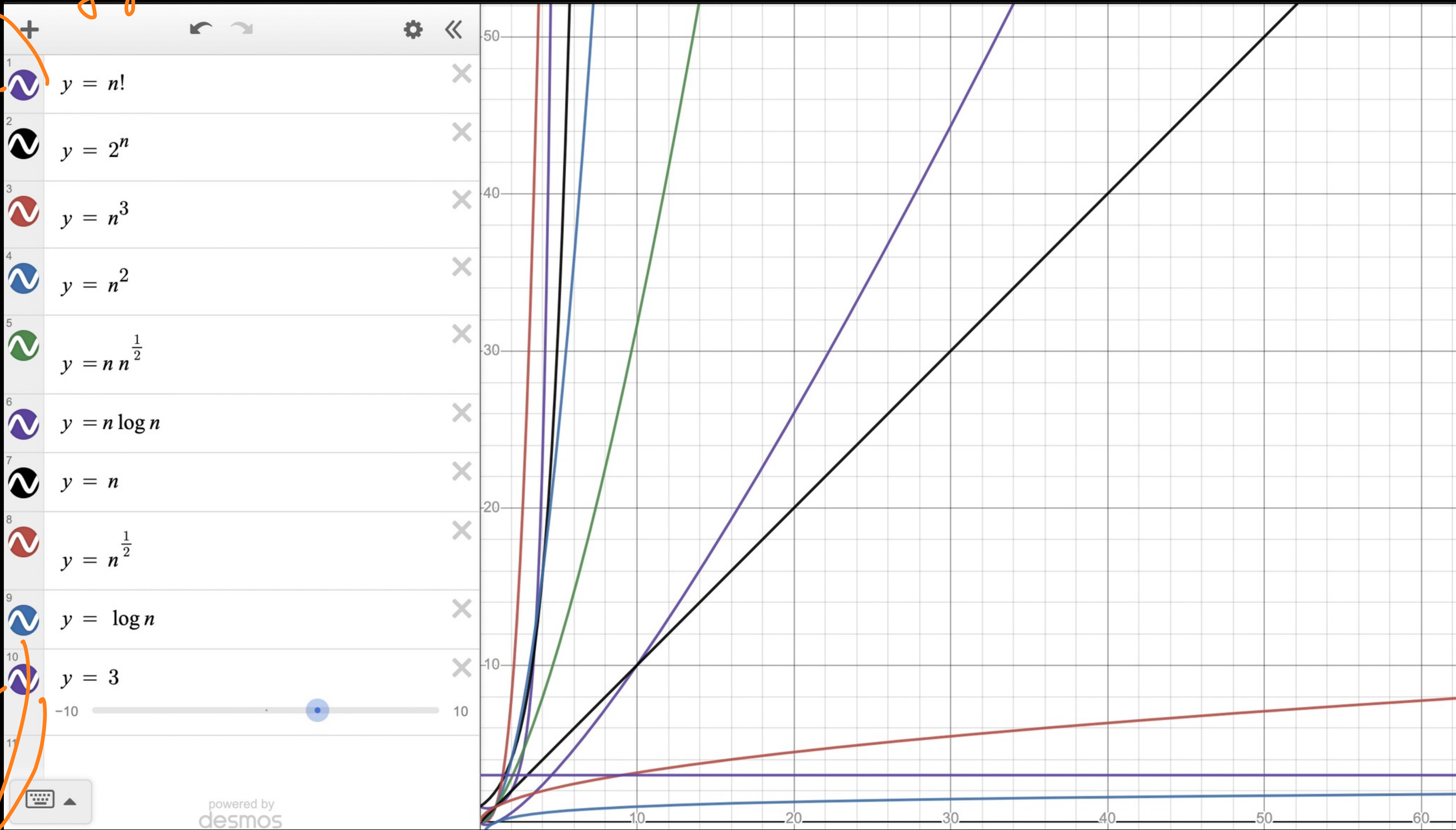
$n \to 10^3$ $\qquad$ $y = 10^3 \times 10 \to 10^4$ $\qquad$ $y = 10^3$

$n \to 10^6$ $\qquad$ $y = 10^6 \times 20 \to 2 \times 10^7$ $\qquad$ $y = \underline{\underline{10^6}}$



high rate of growt

worst



best

slow rate of growth

Algo1

$y \Rightarrow 3x^2 + 8$

$x \to 10^6$

$y = 3 \times (10^6)^2 + 8$ ↳ lower degree term

$3 \times 10^{12} + 8$ ↳ neglygible

avoidable

Algo 2

$y = 10x^5 + 3x^3 + 2x + 10$

const

$y = 10 \times (10^6)^5 + \underbrace{3 \times (10^6)^3 + 2 \times 10^6 + 10}_{\text{neglygible}}$

negligable

lower degree terms

→ we avoid all constants and lower degree terms.

→ we only concern ourselves with the highest degree term.

algo 1 $\quad\longleftrightarrow\quad$ algo 2

$y \approx x^2 \longleftrightarrow y \approx x^5$

```
let x = 20;
let y = 3000;
let count = 3;
for(let i = x; i ≤ y; i++) {
    console.log(i*2);
    count += i;
}

console.log("END");
```

$\rightarrow O(n)$

Let $(y-x)$ be n

$\underbrace{1 + 1 + 1 + 1 + 4n + 1}$

$4n + 5 \rightarrow f(n)$

$n \rightarrow 10^7$

$4 \times 10^7 + 5 \rightarrow$ negligible

approx $\rightarrow n$

$O(n)$

$y = n \rightarrow$ linear st. line

curve

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
$\rightarrow 4n+5$	9	13	17	21	25	29
$5n$	5	10	15	20	25	30

$\forall\ n \geq 6$

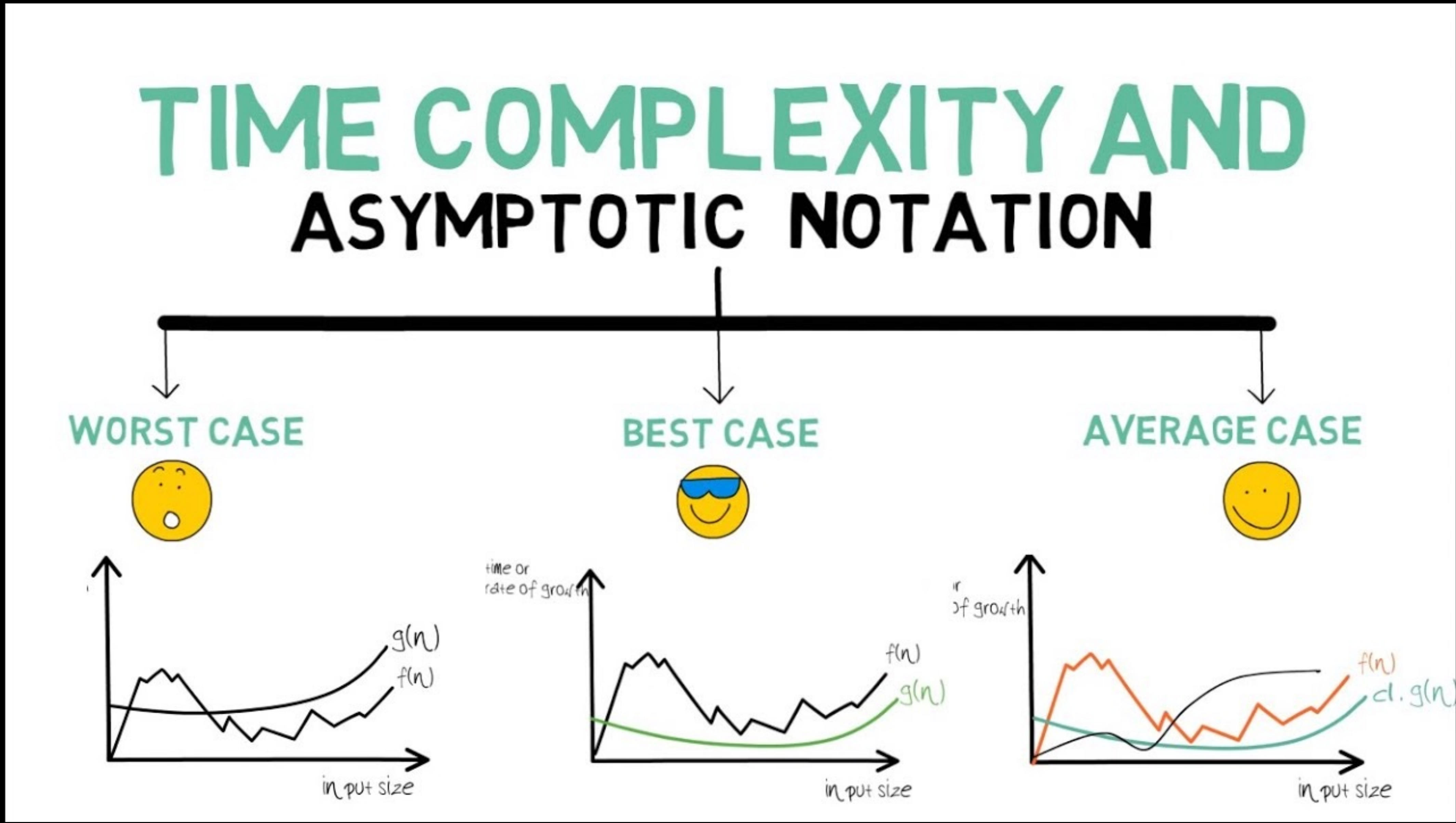
$f(n) \leq c\,g(n)$

$4n+5 \leq 5n$

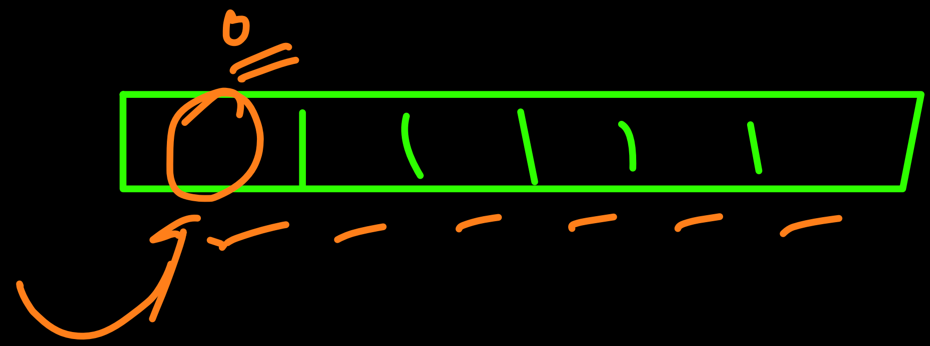
$c \rightarrow 5$

$g(n) = n$

Notations



TIME COMPLEXITY AND
ASYMPTOTIC NOTATION
WORST CASE
BEST CASE
AVERAGE CASE
g(n)
f(n)
input size
time or
rate of growth
f(n)
g(n)
input size
f(n)
c1.g(n)
input size



X x

for representing the rate of growth in terms of highest degree terms, we use these notations

Best Case → Ω	→	Big omega
Avg Case → Θ	→	Big theta
Worst Case → O	→	Big O.

Imp

```
for(i=0;i<n;i++)
   ...
   ...
   ...
}
```

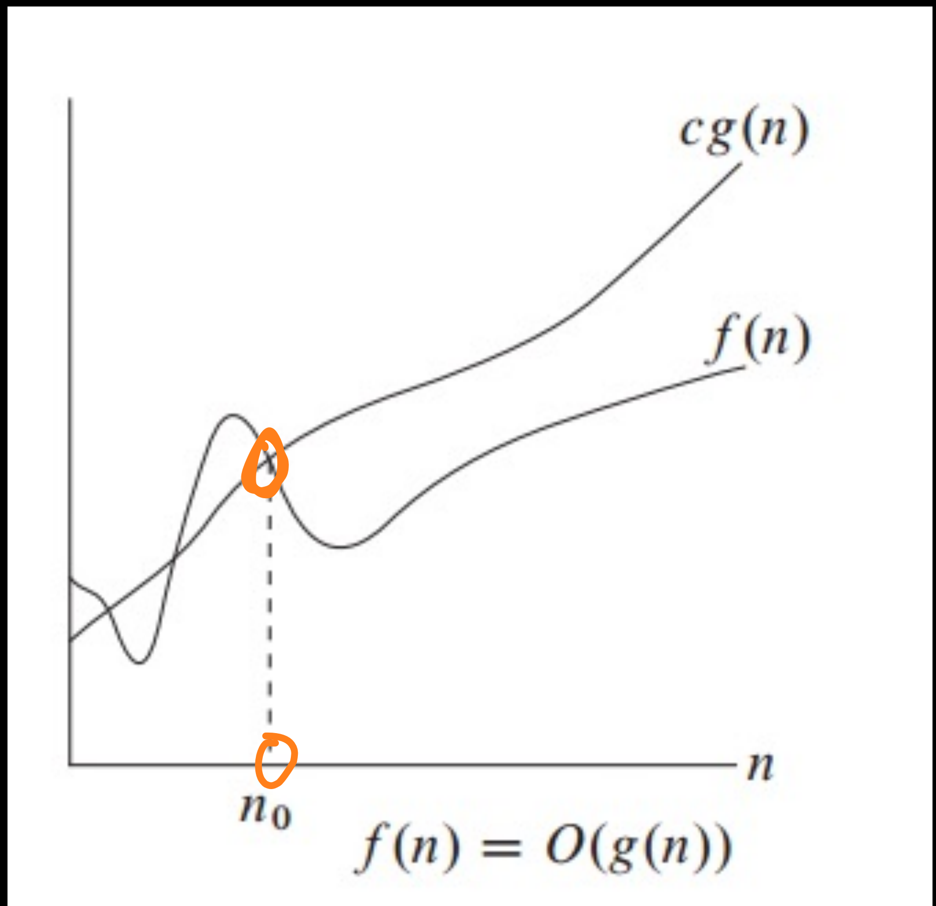
→ $O(n)$

Worst ← Big O Notation

This notation gives Tight Upper Bound of the given function.

Ex ⇒ $f(n) = 2n^2 + 3$ → $O(n^2)$

$n \to 1 \to f(1) \to 5$
$n \to 2 \quad f(2) \to 11$



Big O of the func^n $f(n)$ means,

there is some function $g(n)$ such that

$\forall\, n > n_0 \qquad 0 \le f(n) \le C * g(n)$ $\qquad$ C → constant

$0 \le f(n) \le 5n^2$ $\qquad$ $C = 3$

		$n=1$	$n=10$	$n=100$
$2n^2+3$	$\longrightarrow$	5	203	20003
$3n^2$	$\longrightarrow$	3	300	30000

$\underline{\underline{C=3}}$

Ex → $f(n) = 3n+8$

$\downarrow$

$g(n) = n$

$C \rightarrow 4$

$n_0 = 8$

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$
$3n+8$	11	14	17	20	23	26	29	32
$4n$	4	8	12	16	20	24	28	32

→

$$0 \le f(n) \le c\,g(n) \quad \forall\, n > n_0$$

$$3n+8 \le 4n \rightarrow O(n)$$

2

Ex $\rightarrow f(n) = 2n^3 - 2n^2$

$g(n) \rightarrow n^3$

$c \rightarrow 2$

$O(n^3)$ $f(n) \leq cg(n)$

	$n=1$	$n=2$
$2n^3-2n^2$	0	$16-8=8$
n^3	1	8

$n \geq 1$

2

for(i=0; i<n ; i++)

for(j=0; j<n; j++)

$$\begin{array}{lcl} & & j \\ i=0 & \rightarrow & n \\ i=1 & \rightarrow & n \\ i=2 & \rightarrow & n \\ \vdots & & \vdots \\ i=n-1 & \rightarrow & n \\ \hline & & n^2 \end{array}$$