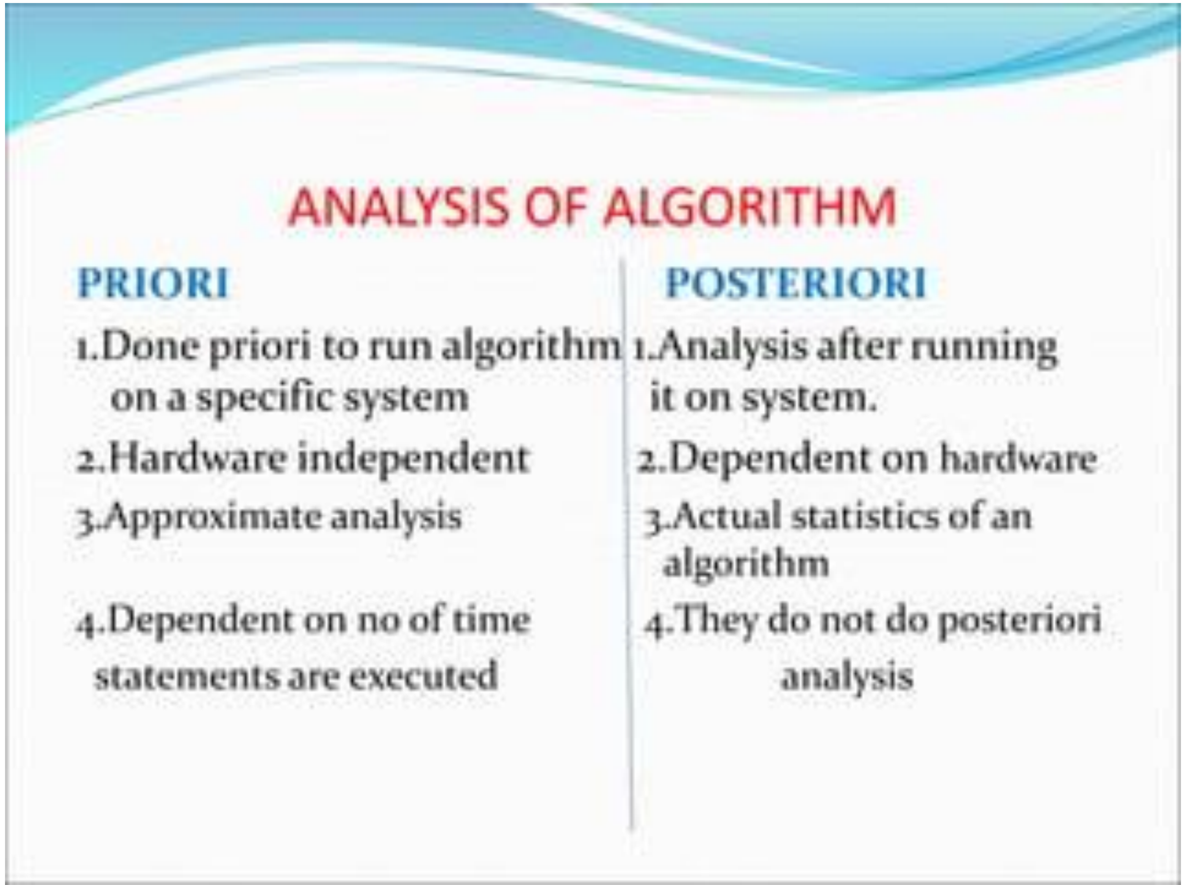
Algorithm	Program
Algorithms are written at design phase	Programs are written at Implementation Stage
Algorithms are programming syntax independent	Programs are programming syntax dependent
Algorithms are not dependant on operating system architecture and hardware	Programs are dependent on operating system architecture and hardware
Algorithms are analysed on efficiency in terms of completion time and the space used.	We just test the program to see if it will scale in production

A Posteriori

Knowledge is obtained through experience

A Priori

Knowledge is obtained by analyzing concepts independent of experience



Characteristics of Algorithm

- Unambiguous:

  Algorithm should be clear and unambiguous. Each of its steps (or phases), one and their inputs/outputs should be clear and must lead to only meaning.
- Input:

  An algorithm should have 0 or more well-defined inputs.
- Output:

  An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- Finiteness:

  Algorithms must terminate after a finite number of steps.
- Feasibility:

  Should be feasible with the available resources.
- Independent:

  An algorithm should have step-by-step directions, which should be independent of any programming code.

uage

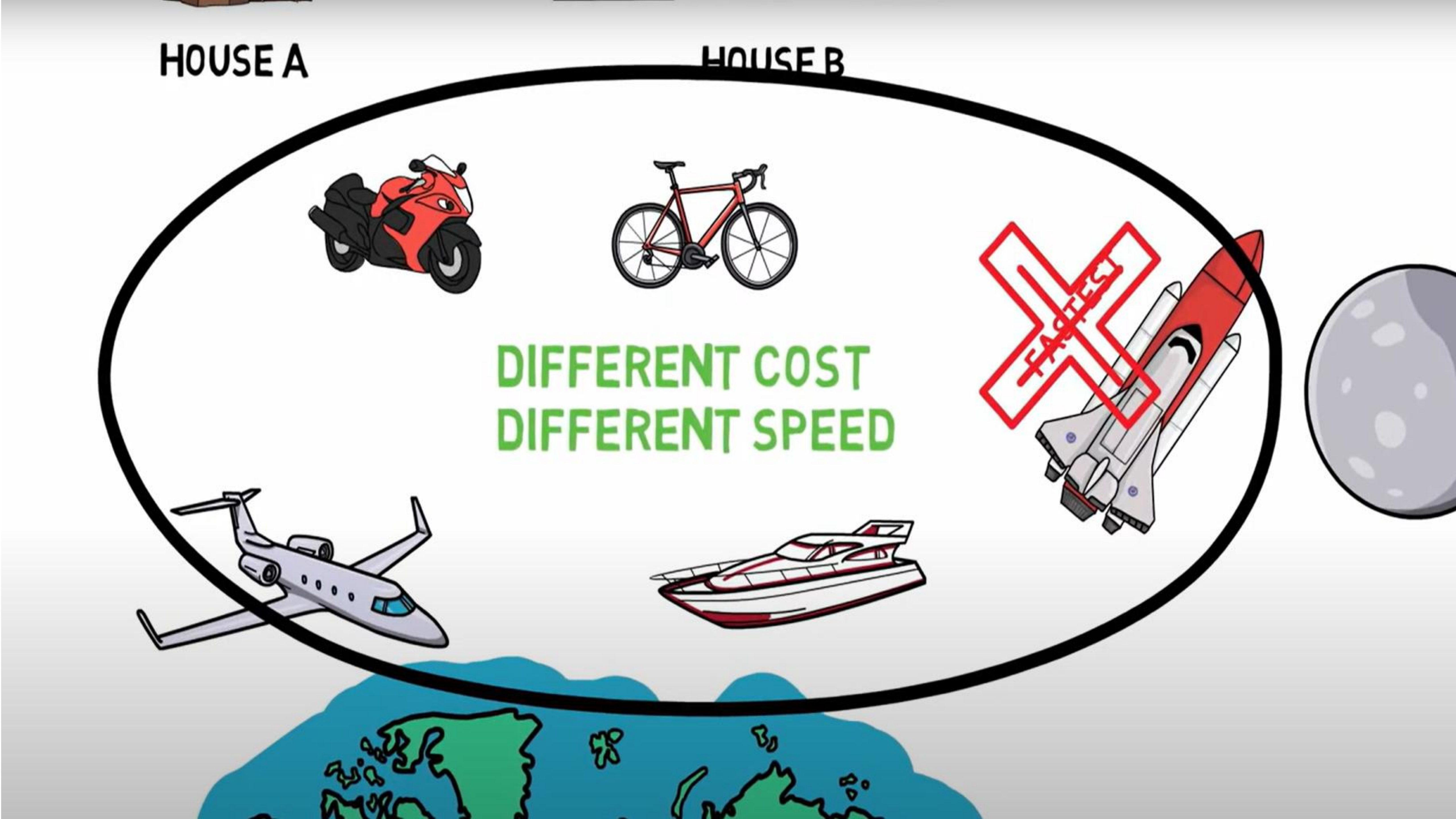
ed

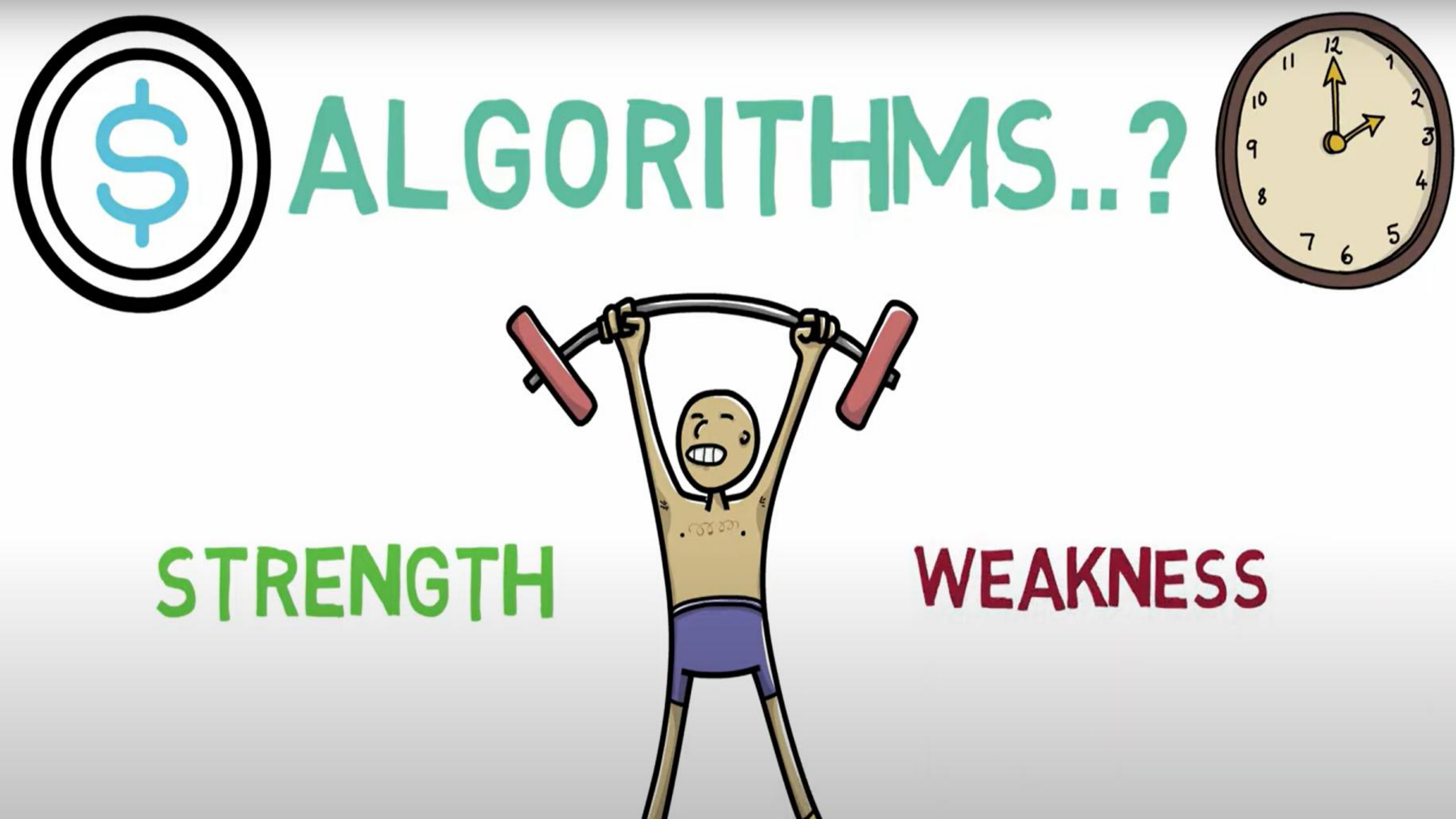
Time complexity

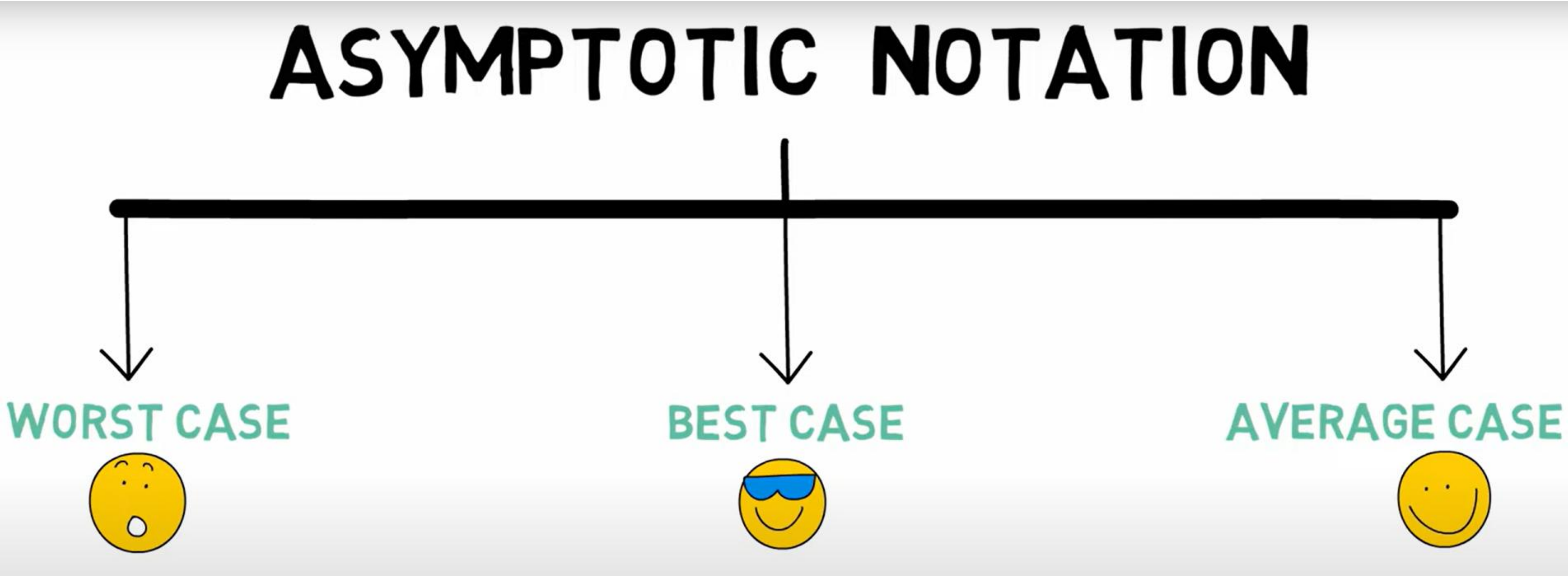
SEARCHING, SORTING, ETC....

HOUSE A

HOUSE B


HOUSE A
HOUSE B
DIFFERENT COST
DIFFERENT SPEED
FASTEST

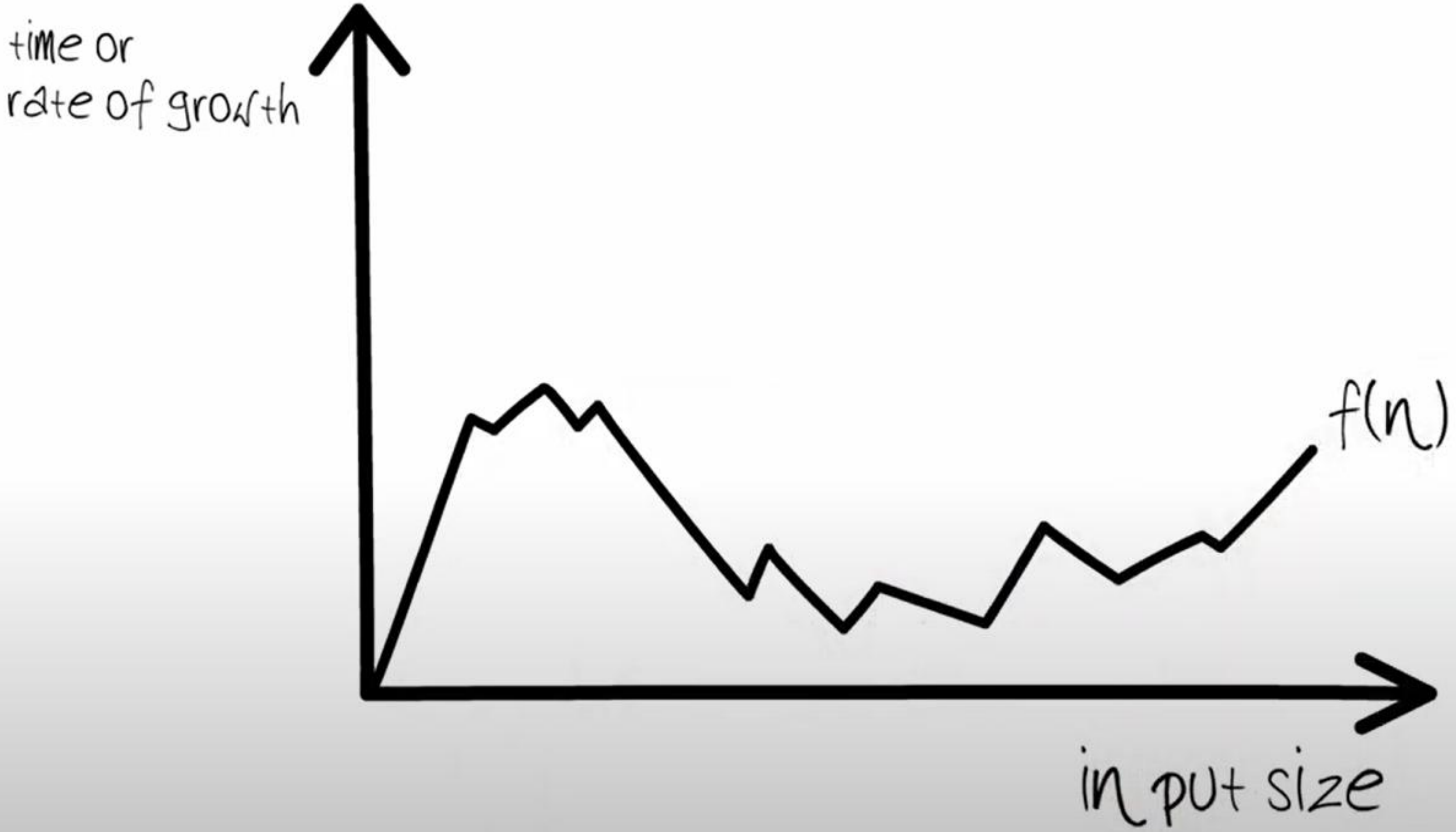

ALGORITHMS..?
STRENGTH
WEAKNESS


ASYMPTOTIC NOTATION
WORST CASE
BEST CASE
AVERAGE CASE

WORST CASE

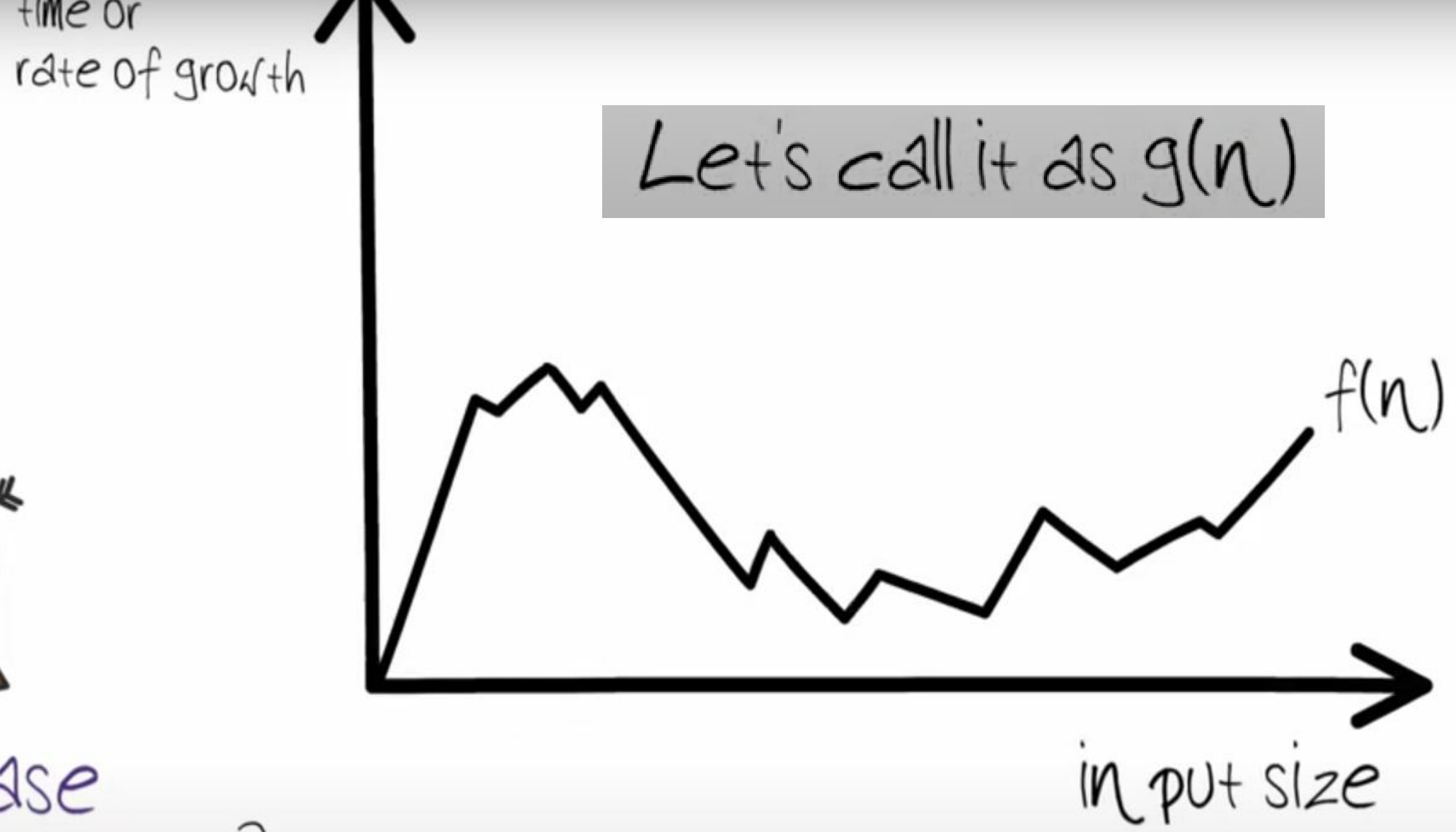
Big Oh Notation O

$f(n)$ = Our Algorithm



MATHS.....

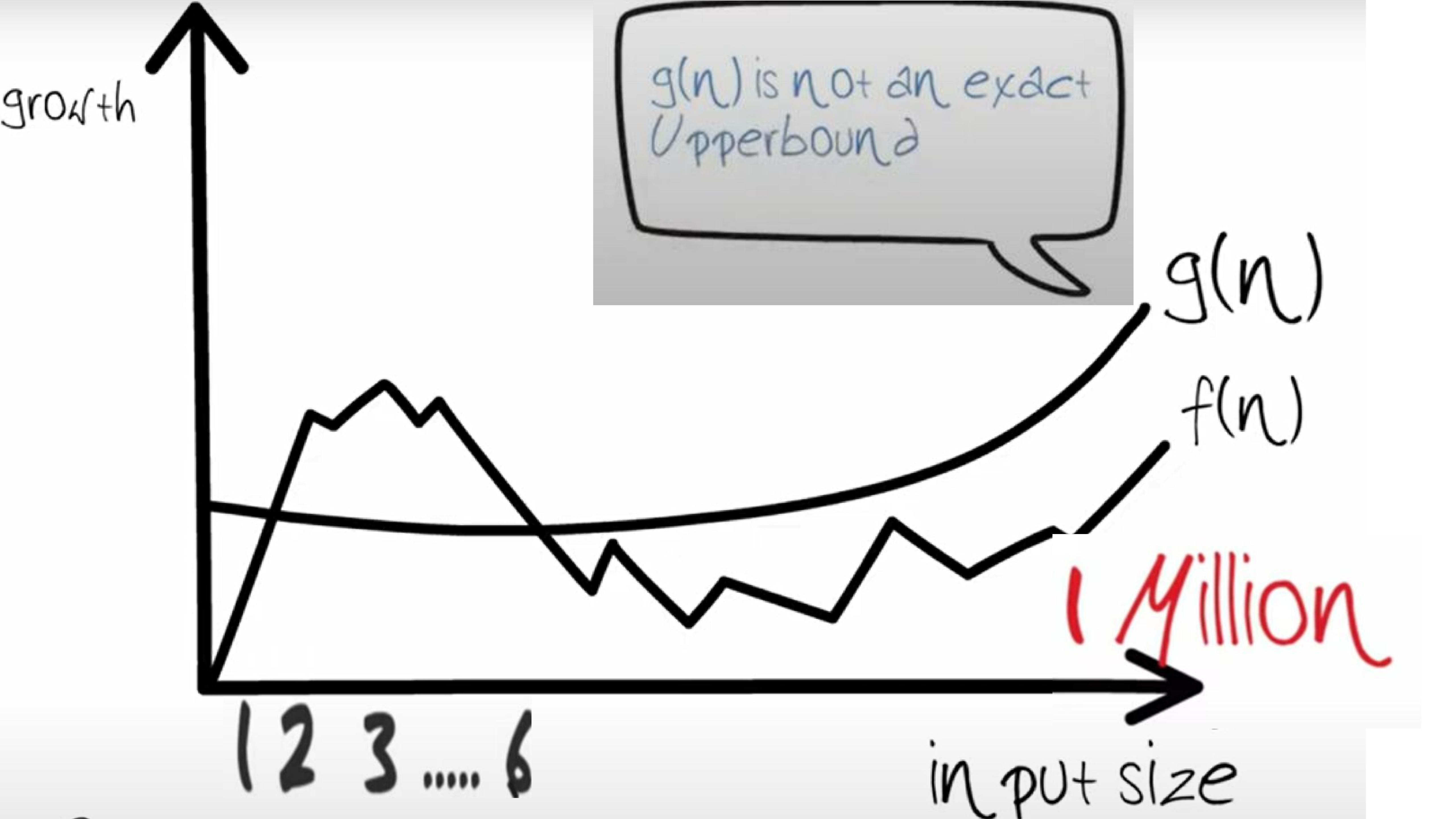
Let's call it as g(n)



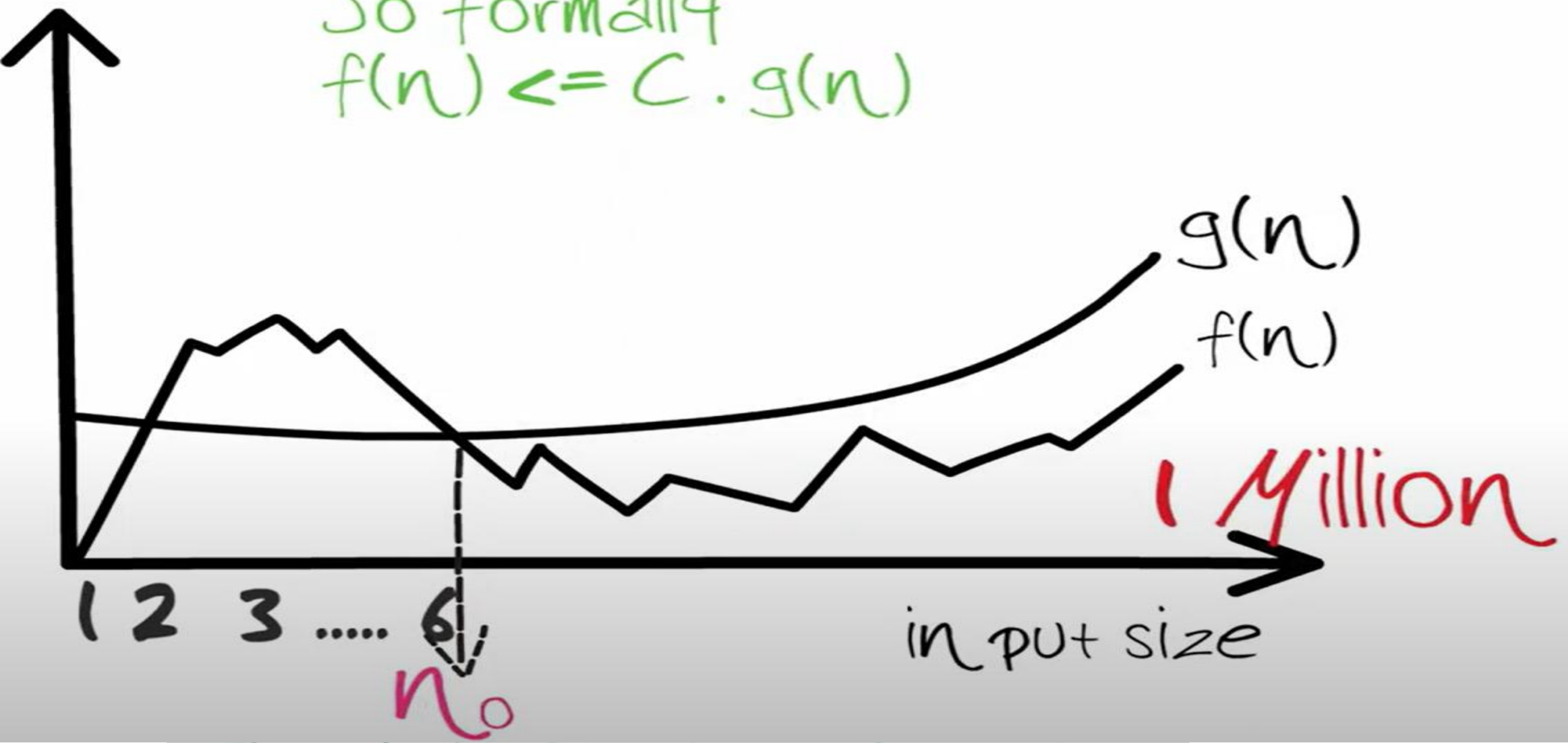
worst case

We just need to find an upper bound

We just need to find a function which is slightly greater than our f(n) function

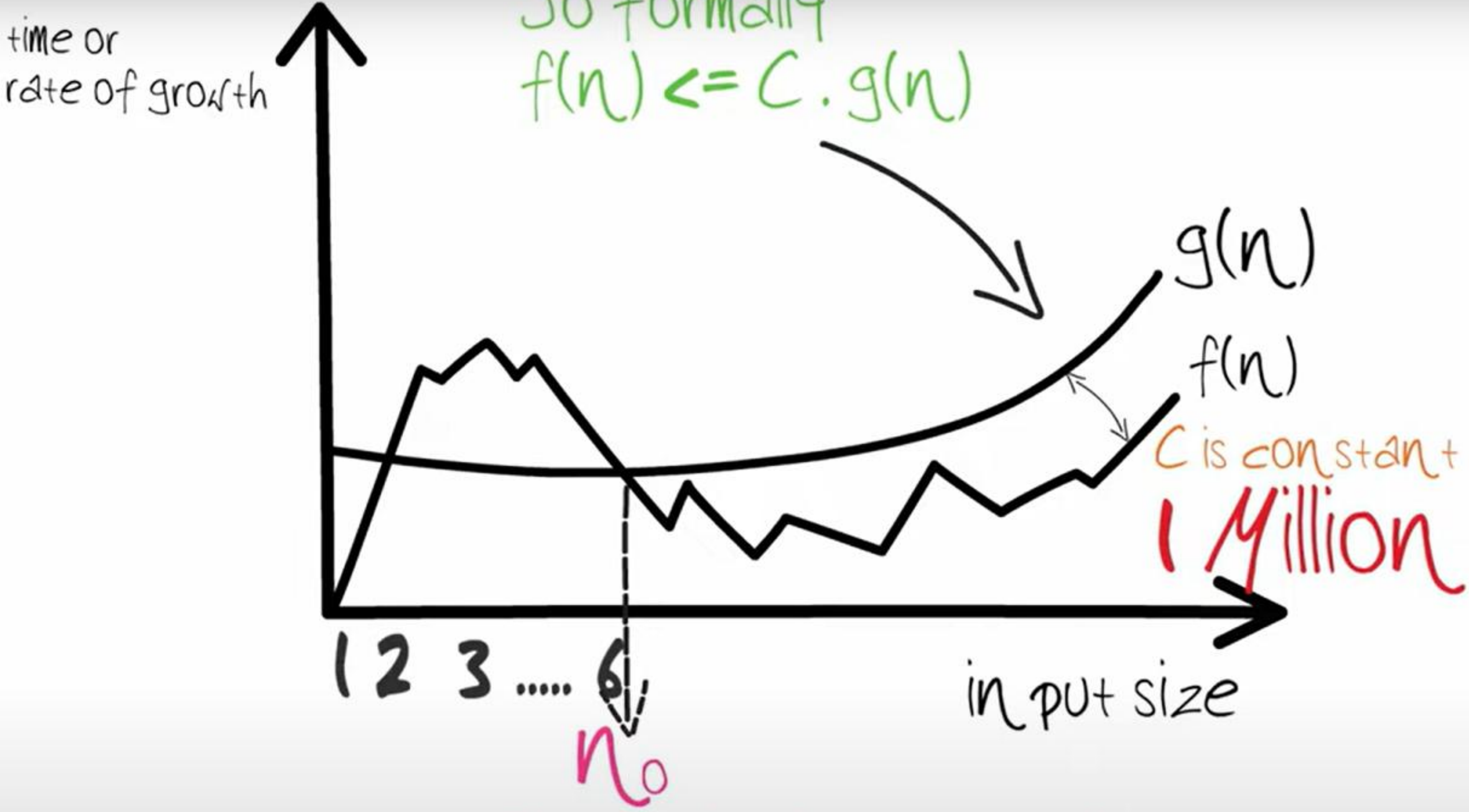

g(n) is not an exact Upperbound
growth
g(n)
f(n)
1 Million
1 2 3 6
in put size

So formally
$f(n) <= C \cdot g(n)$

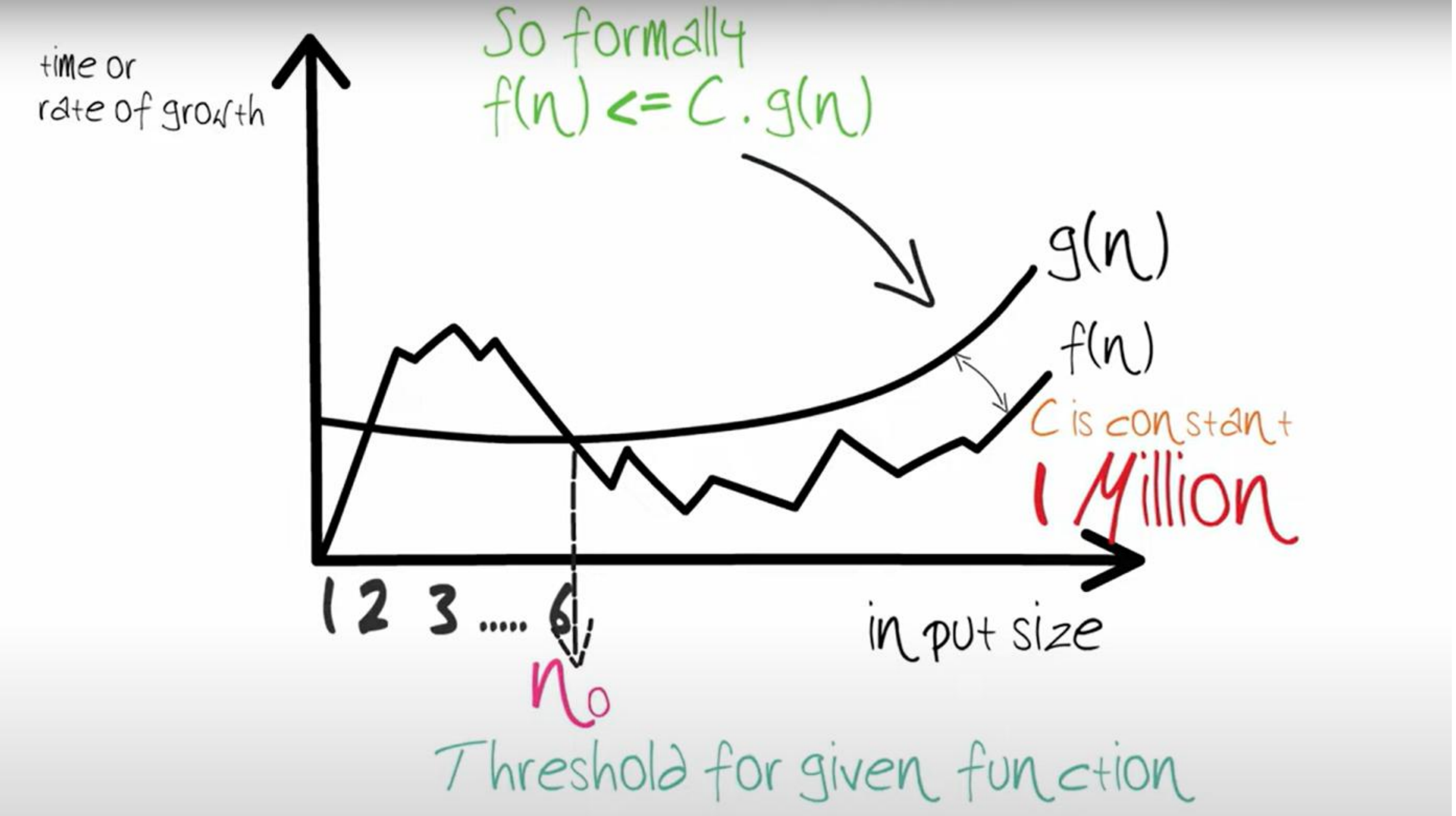


Threshold for given function

So formally
$f(n) <= C \cdot g(n)$



Threshold for given function



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

means $c \cdot g(n)$ is an upper bound on $f(n)$.

C and n_0 are constant

Find upper bound for $f(n) = n^2 + 1$

$$\boxed{f(n) <= c.g(n)}$$

$$n^2 + 1 <= c.n^2$$

$$n^2 + 1 <= 2.n^2 \quad \text{,for } n >= 1$$

$$\boxed{n^2 + 1 = O(n^2)}$$ for c = 2 and n0 >= 1

This is Big Oh Notation

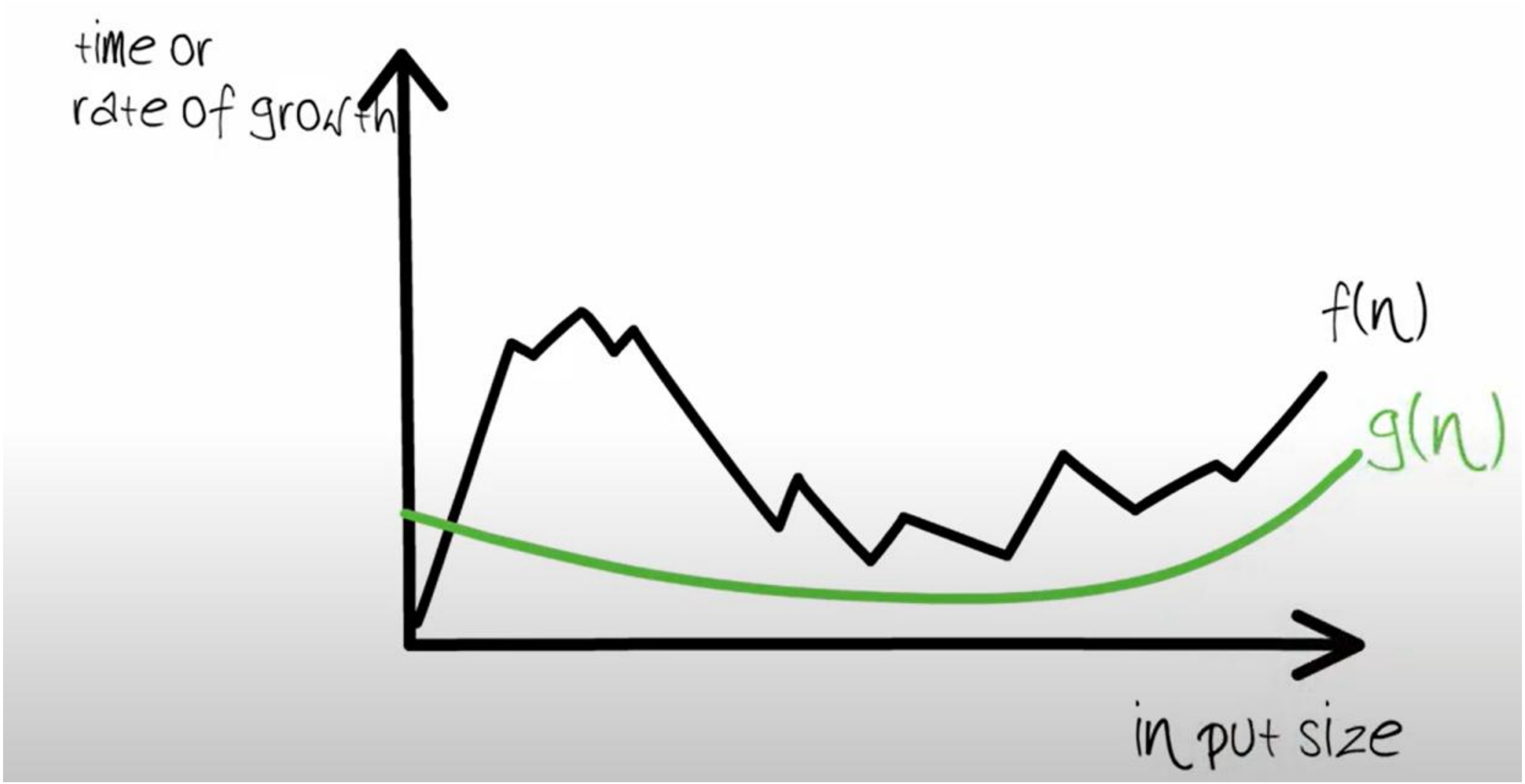

BEST CASE
Omega-Q Notation Ω

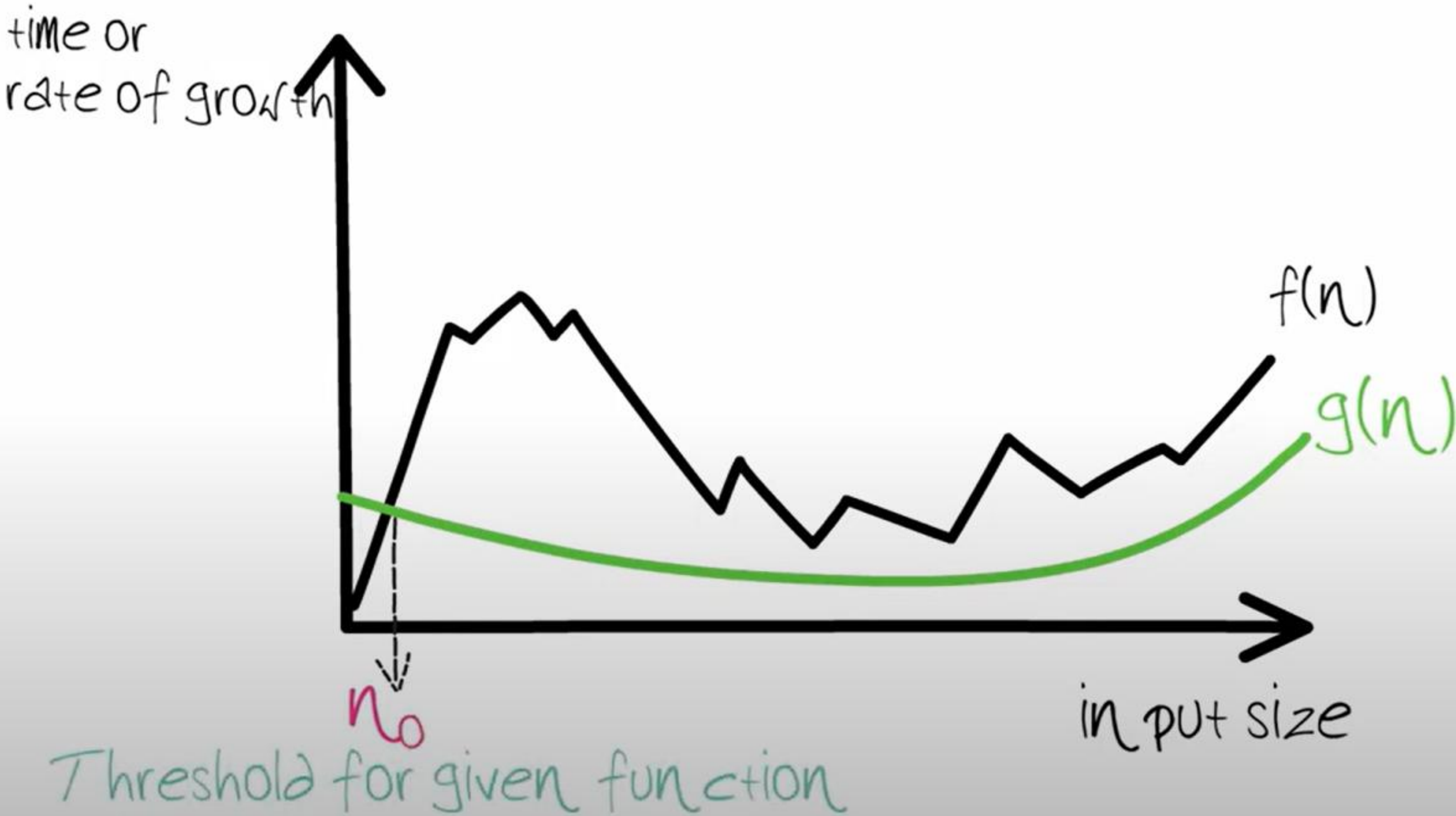
Omega-Ω Notation Ω

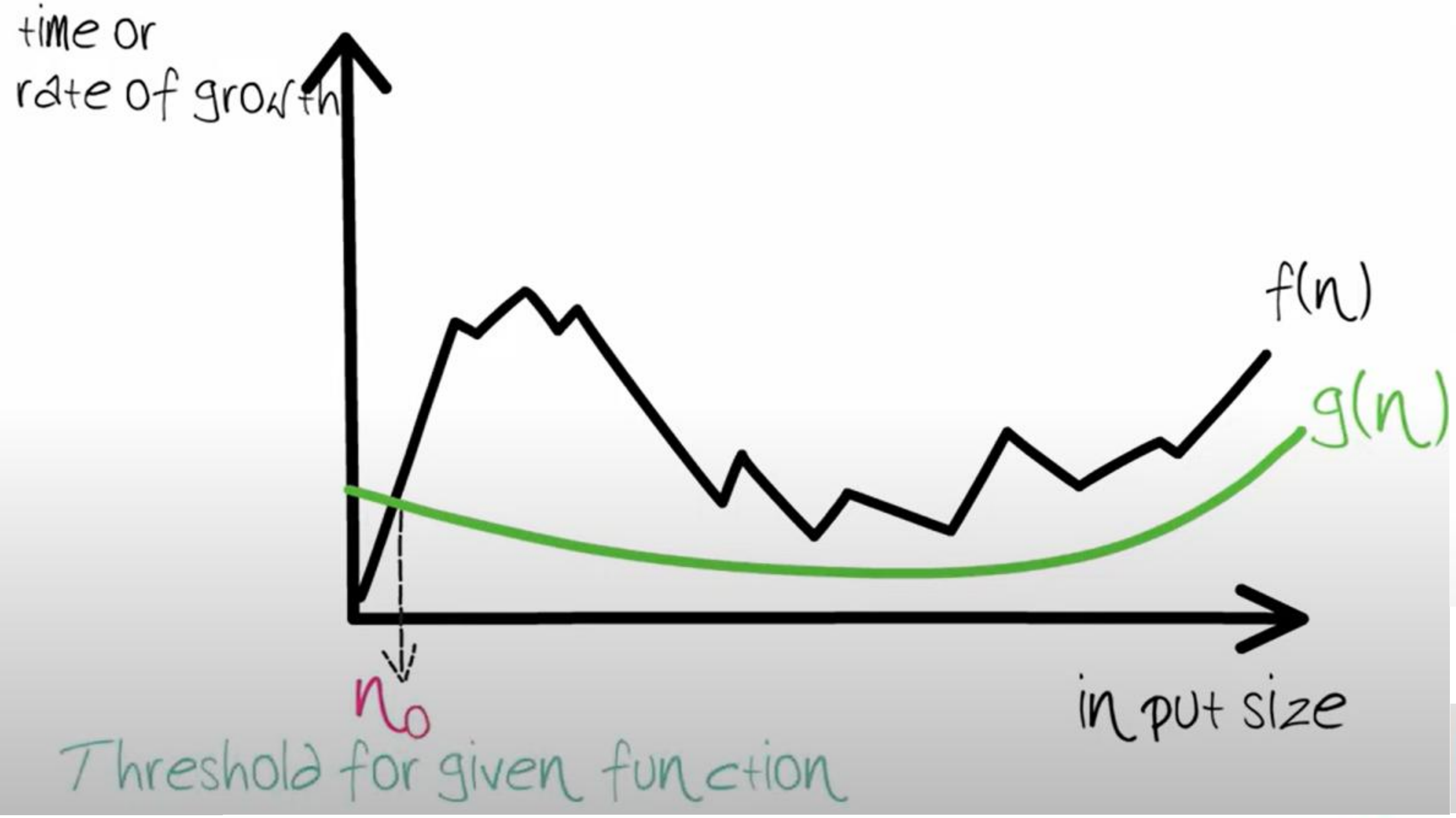
f(n) = Our Algorithm

Worst Case (Big Oh) : Upperbound Function

Best Case (Omega) : Lowerbound Function


time or
rate of growth
f(n)
g(n)
input size


time or
rate of growth
f(n)
g(n)
input size
n0
Threshold for given function



So formally

$f(n) >= C . g(n)$

$$\boxed{f(n) = \Omega(g(n))}$$

Find lower bound for $f(n) = 5n^2$

Find lower bound for $f(n) = 5n^2$

$$\boxed{f(n) >= c.g(n)}$$

$5n^2 >= c.\, n^2$

from c = 5 and n =1 it hold true

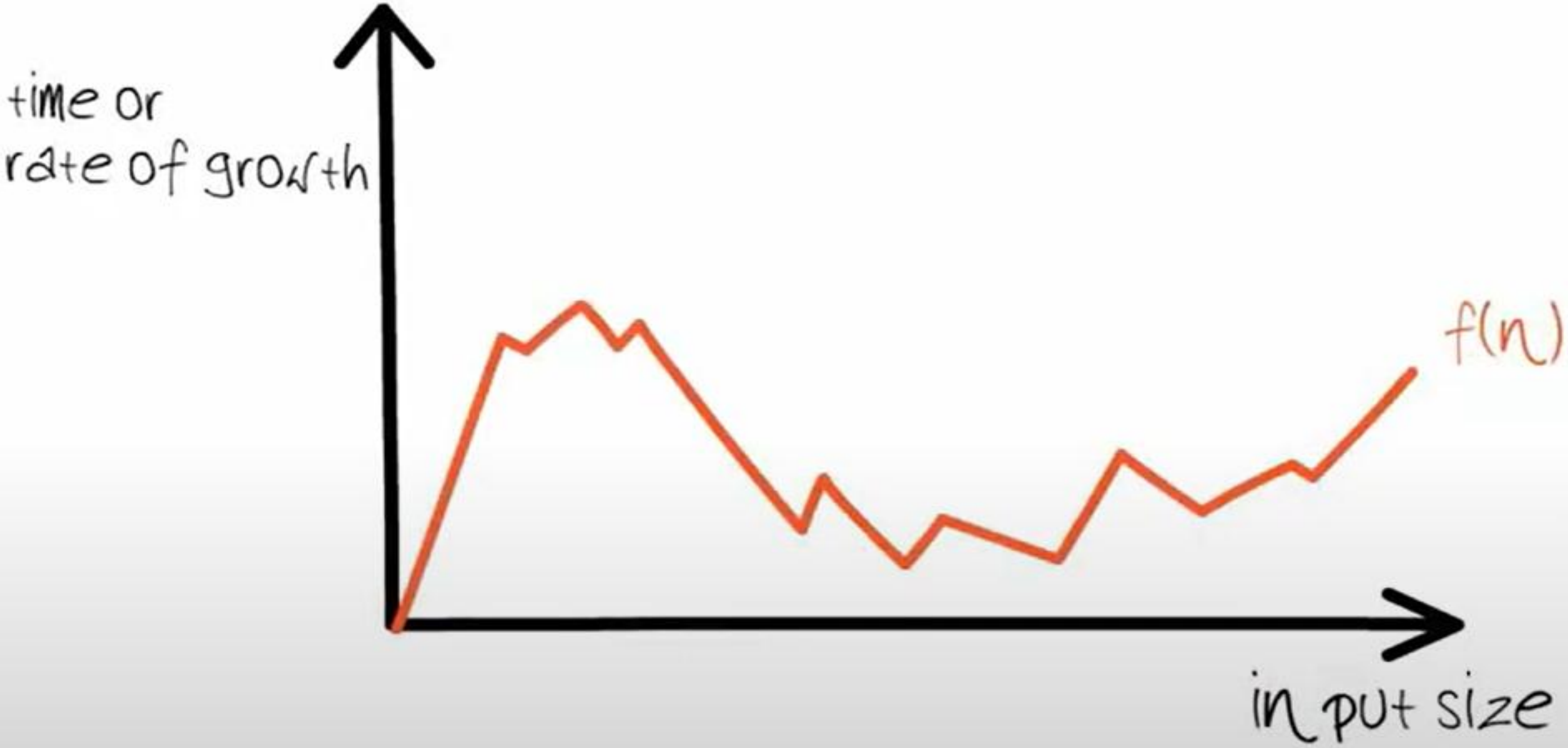
So $\boxed{5n^2 = \Omega(n^2)}$

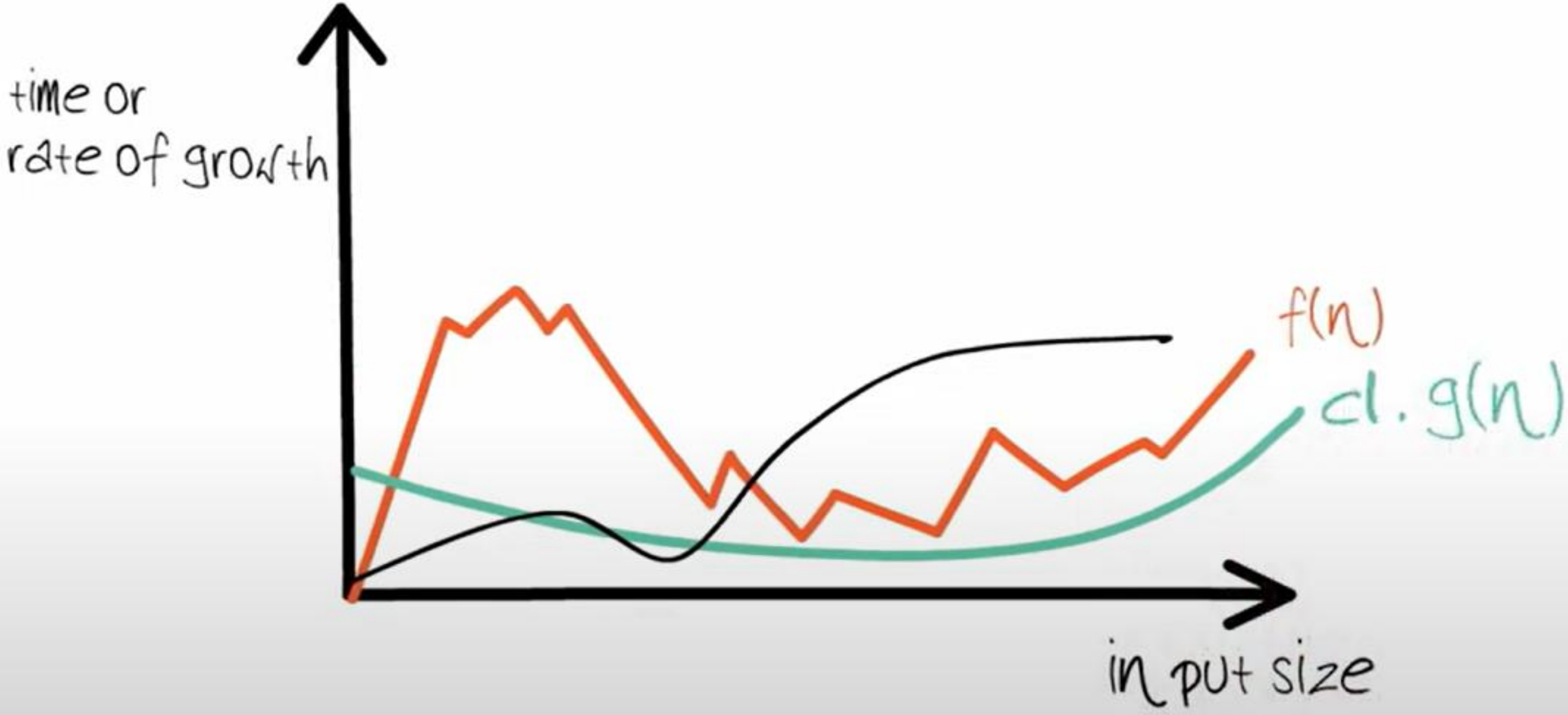
AVERAGE CASE

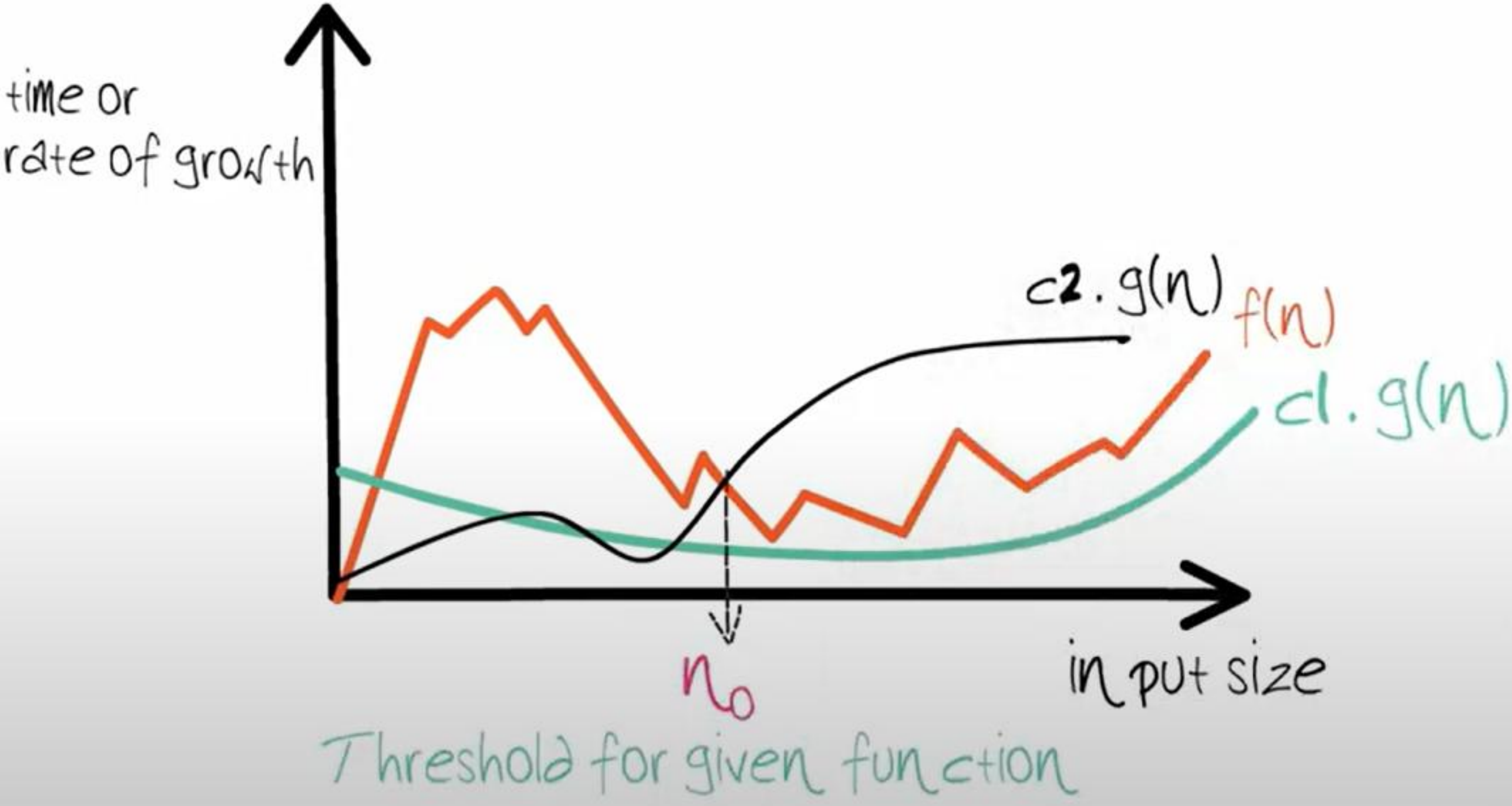
Theta Notation θ

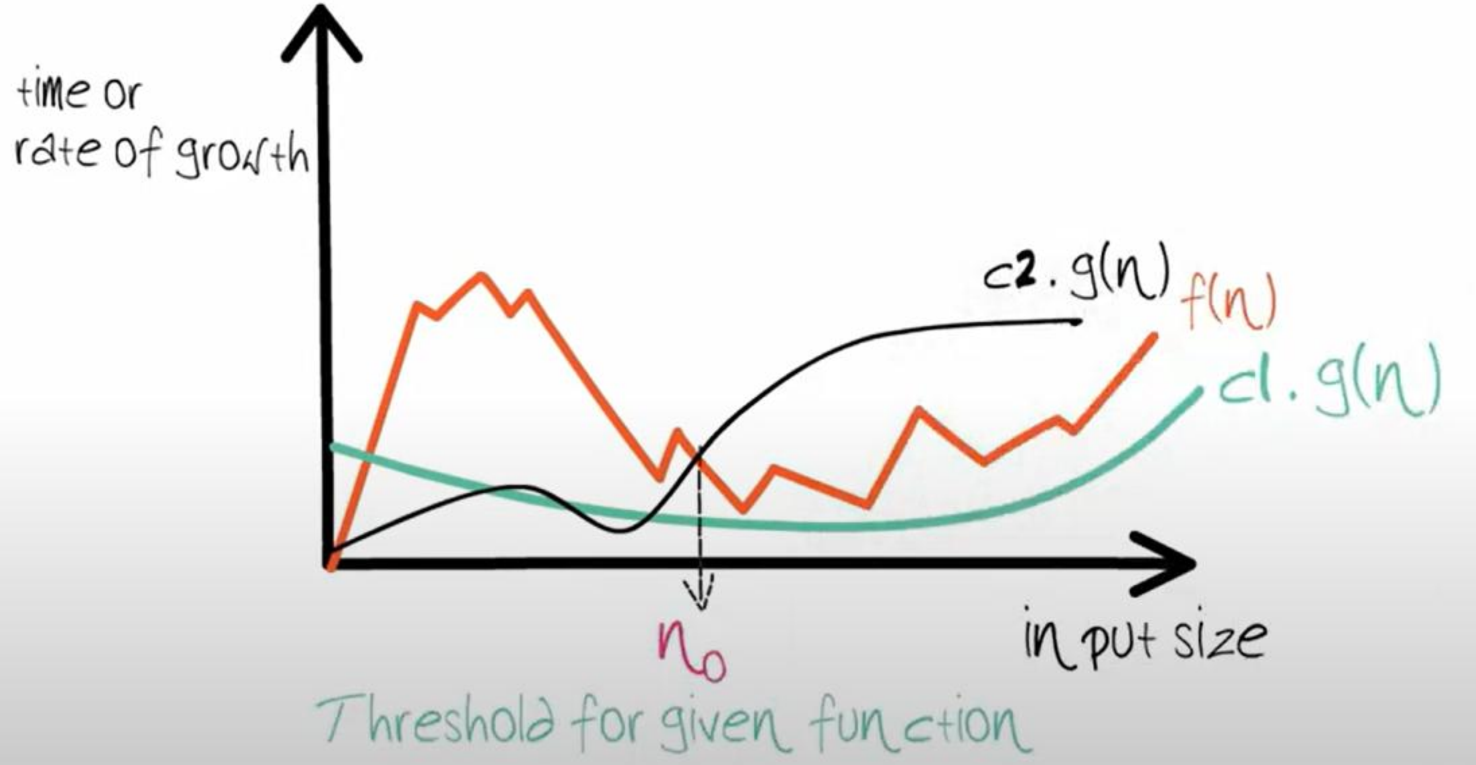
Theta Notation Θ

f(n) = Our Algorithm




time or
rate of growth
f(n)
c1. g(n)
input size


time or
rate of growth
c2 . g(n)
f(n)
c1 . g(n)
input size
n_0
Threshold for given function



$$c1.g(n) <= f(n) <= c2.g(n)$$

$$f(n) = \theta\, g(n)$$

where c1, c2, n > 0

Prove that $f(n) = \Theta\, g(n)$ where

$f(n) = 3n + 2$ and $g(n) = n$

Lowerbound: $c1.\, g(n) <= f(n)$

$c1.\, n <= 3n + 2$

$c1 = 1$ and $n>= 1$

$n <= 3n+2$

for $n>=1$ is True

Upperbound: $f(n) <= c2.\, g(n)$

$3n + 2 <= c2 .\, n$

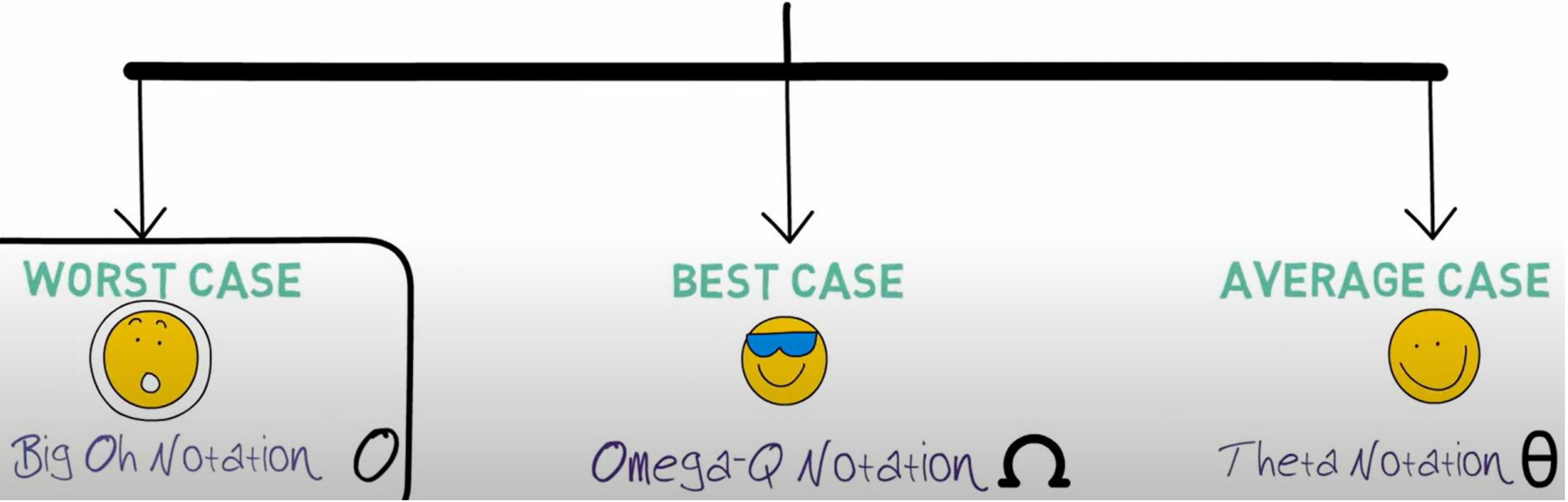
for $c2 = 4$ and $n>=1$

$3n + 2 <= 4n$

for $n>=1$ is True

$3n+2 = \Theta(n)$

ASYMPTOTIC NOTATION



Some Common Algo's Complexities

Input

n $f(n)$	$\lg n$	n	$n \lg n$	n^2	2^n	$n!$
10	0.003 μs	0.01 μs	0.033 μs	0.1 μs	1 μs	3.63 ms
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	1 ms	77.1 years
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	1 sec	8.4×10^{15} yrs
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	18.3 min	
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	13 days	
100	0.007 μs	0.1 μs	0.644 μs	10 μs	4×10^{13} yrs	
1,000	0.010 μs	1.00 μs	9.966 μs	1 ms		
10,000	0.013 μs	10 μs	130 μs	100 ms		
100,000	0.017 μs	0.10 ms	1.67 ms	10 sec		
1,000,000	0.020 μs	1 ms	19.93 ms	16.7 min		
10,000,000	0.023 μs	0.01 sec	0.23 sec	1.16 days		
100,000,000	0.027 μs	0.10 sec	2.66 sec	115.7 days		
1,000,000,000	0.030 μs	1 sec	29.90 sec	31.7 years		

Figure 2.4: Growth rates of common functions measured in nanoseconds

Book: Algorithm Design Manual by Skiena