

## TIME COMPLEXITY & BIG O NOTATION 2

Ex: This morning, I wanted to eat some pizzas; so I asked my brother to get me some from Dominos (3km far).


He got me the pizza & I was happy only to realize it was to be for 29 friends who came to my house for a surprise visit!

My brother can get 2 pizzas for me on his bike but pizza for 29 friends




is too huge of an input for him which he cannot handle.

2 PIZZAS  $\rightarrow$   OKAY! NOT A BIG DEAL.

68 PIZZAS  $\rightarrow$   NOT POSSIBLE!  
IN SHORT TIME.

$\rightarrow$  What is Time Complexity?

$\rightarrow$  Time complexity is the study of efficiency of algorithms.

 Time Complexity  $\Rightarrow$  How time taken to execute an algorithm grows with the size of the input!

$\rightarrow$  Consider two developers who created an algorithm to sort  $n$  numbers. Shulham & Rohan did this independently.

When ran for input size  $n$ , following results were recorded.

no. of elements ( $n$ )	Shulham's Algo	Rohan's Algo
10 elements	90 ms	122 ms
20 "	110 ms	124 ms
110 "	180 ms	131 ms
1000 "	2s	800 ms



→ We can see that initially Shubham's algorithm was shining for smaller input but as the number of elements increases Rohan's also algorithm looks good!

→ Time Complexity: Sending GTA V to a friend.  
→ Let us say you have a friend living 5km away from your place. You want to send him a game.

Final exams are over & you want him to get this 60GB file for you. How will you send it to him?

NOTE: Both of you are using 304G with 1GB/s data limit.

→ The best way to send him the game is by delivering it to his house.

Copy the game to a Hard disk & send it!

→ Will you do the same thing for sending a game like Minesweeper which is in KBs of size?

→ No, because you can send it via internet.

→ As the file size grows, time taken by online sending increases linearly  $\rightarrow O(n)$

↓  
Big O of  $n$

→ As the file size grows, time taken by physical sending remains constant.  $O(1)$  or  $O(c)$

↓  
Big O of 1



→ Calculating order in terms of input size:  
 → In order to calculate the order, most impactful term containing  $n$  is taken into account.  
 ↓  
Size of input.

→ Let us assume that formula of an algorithm in terms of input size  $n$  looks like this.

$$\text{Algo 1} \rightarrow \underbrace{K_1 n^2}_{\text{Highest order term}} + \underbrace{K_2 n + 36}_{\text{can ignore lower order terms}} \Rightarrow O(n^2)$$

$$\text{Algo 2} \rightarrow K_1 K_2^2 + K_3 + 8$$

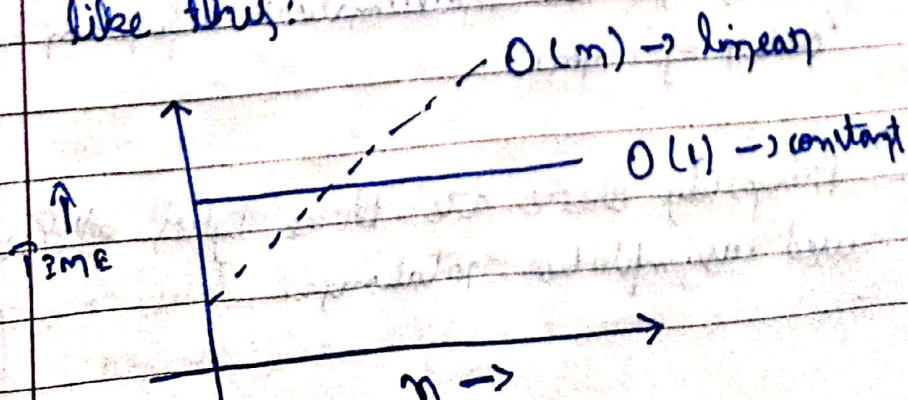
$$\Downarrow$$

$$K_1 K_2^2 n^0 + K_3 + 8 \Rightarrow O(n^0) \text{ OR } O(1)$$

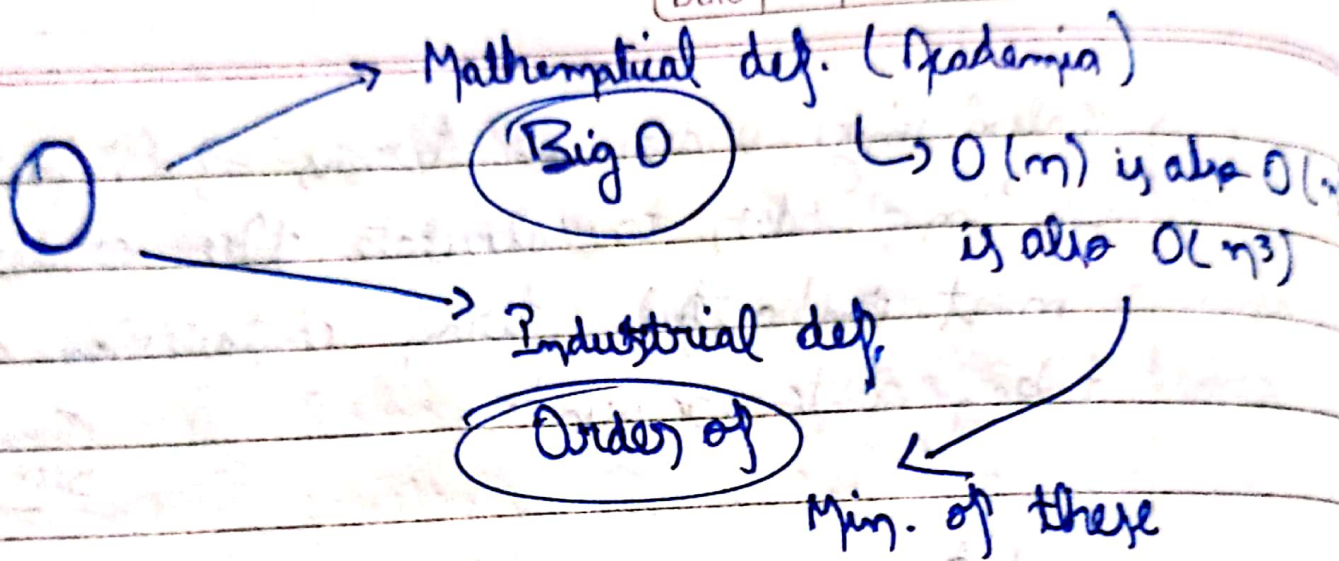
NOTE: These are formulas for time taken by them.

→ Visualizing Big O.

→ If we were to plot  $O(1)$  &  $O(n)$  on a graph, they will look something like this:







## Big O Complexity Chart

