Big Oh (O) Notation

Best Case

Worst Case

Average Case

Linear Search

Best Case : 0

Worst Case : N

Average Case : N/2

This notation is used to find upper bound of an algorithm running time:

It tells us that running time of that algorithm will not be more than that upper bound

It represents the worst case of an algorithm time complexity. i.e. The longest amount of time an algorithm can possibly take to complete

A function f(n) = O g(n) , if there exists a value of positive integer n and n0 and positive constant c such that

f(n) <= c g(n) for all n > n0

Hence function g(n) is an upper bound for function f(m) , as g(n) grows faster than f(n)

Which Algo is better than the other one and how??

By using Asymptotic Notation : We use this notation to compare one algorithm with other Algorithm

Big Oh O > A function f(n) is said to be a Big O of g(n) if and only if there exists a constant c and a constant n0 such that

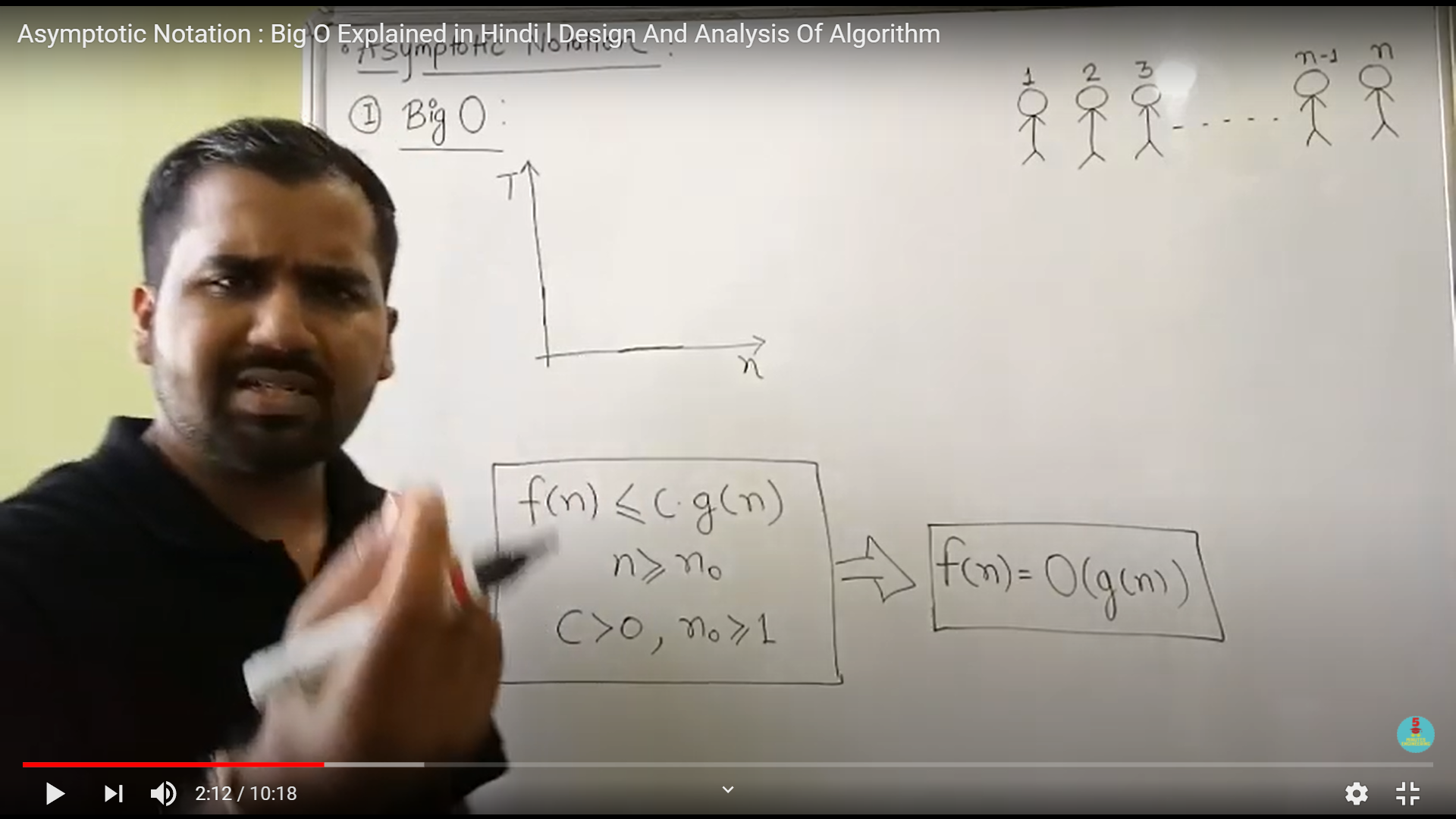
0 <= f(n) <= c g(n)

Here f(n) is time and n is input

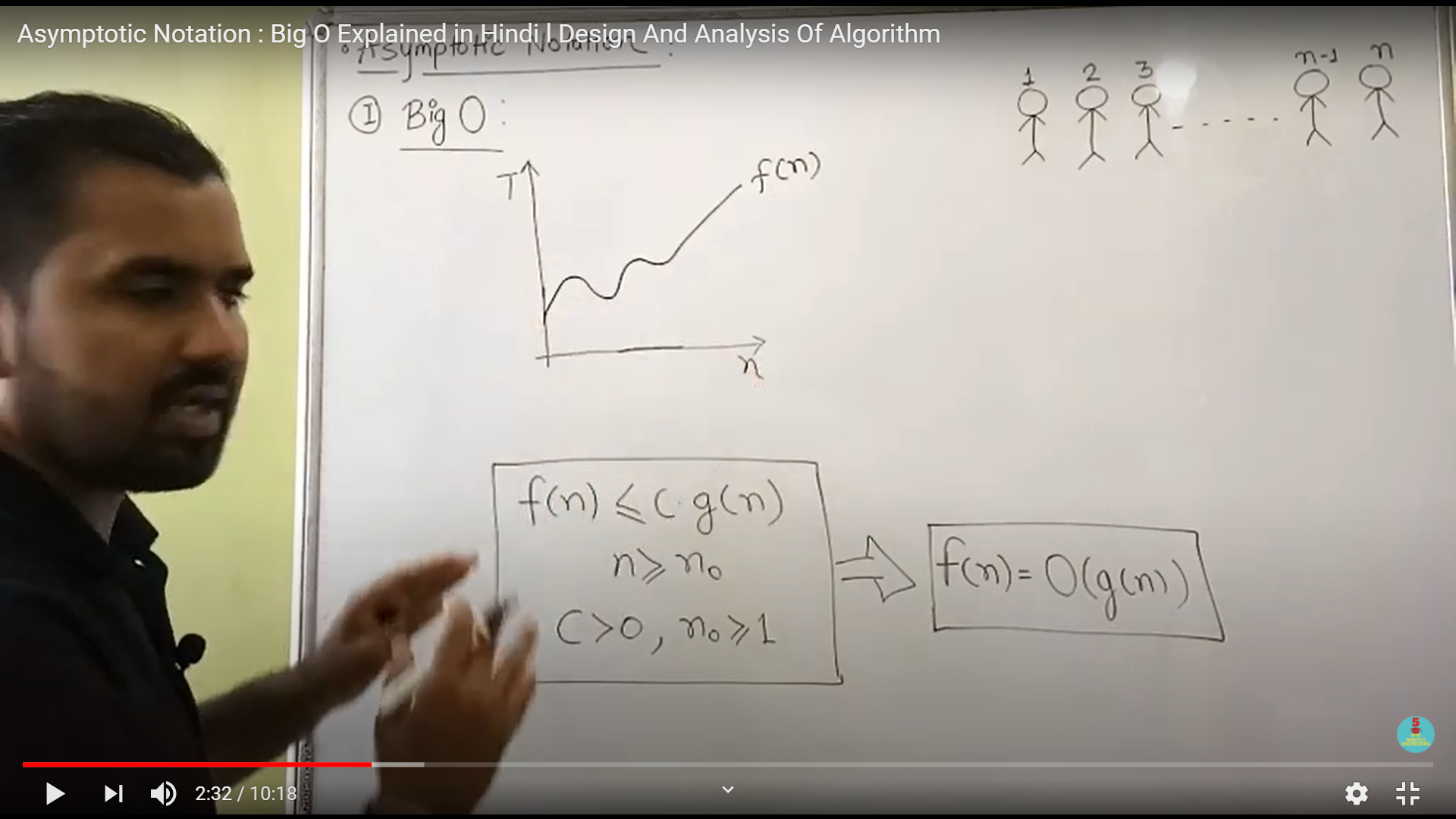
To understand any Algorithm complexity, we use Big O notation

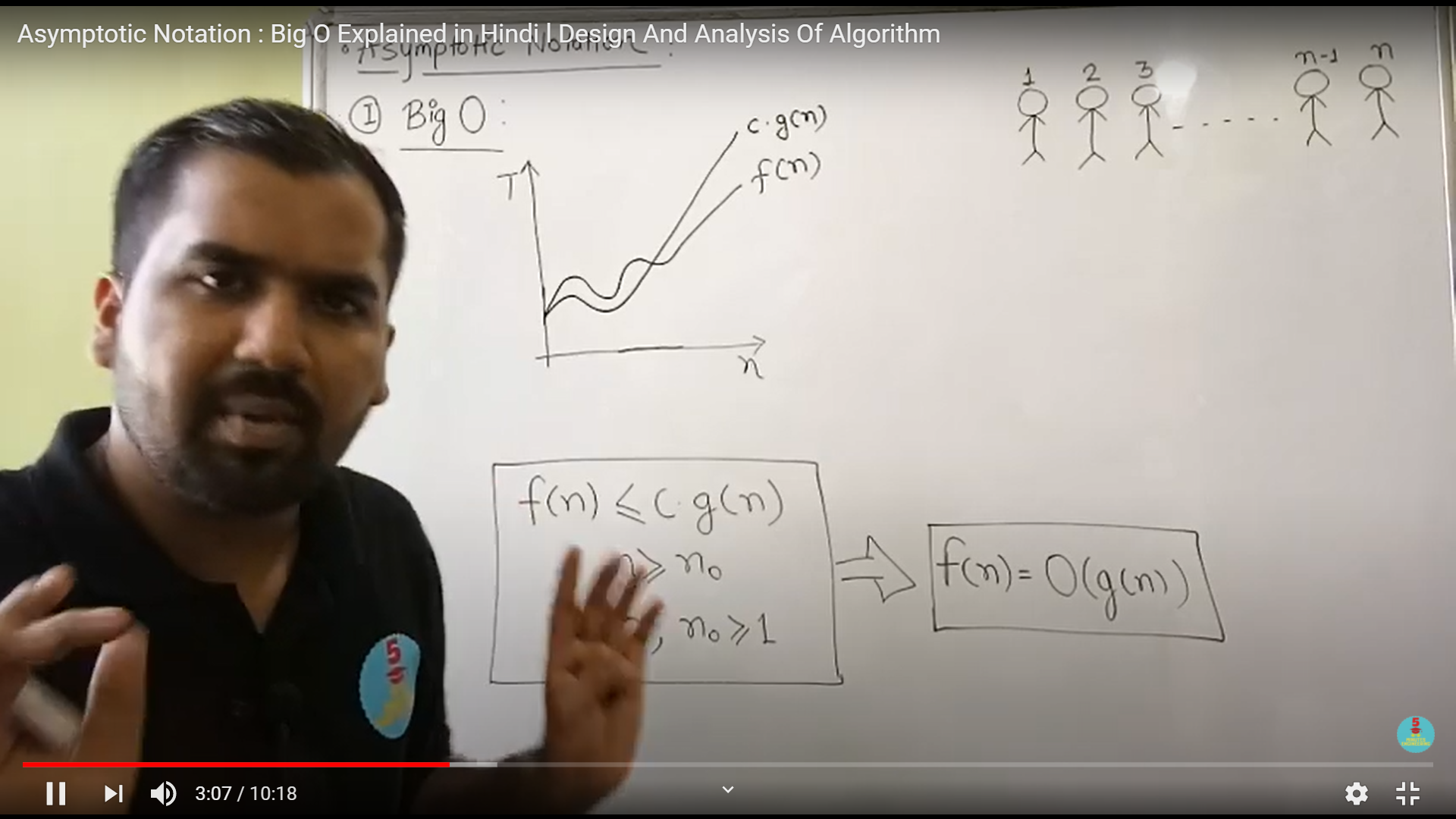
In this chart, x represents no of inputs and y represents Time growth

Lets create Chart



With Big O we should think upper bound always

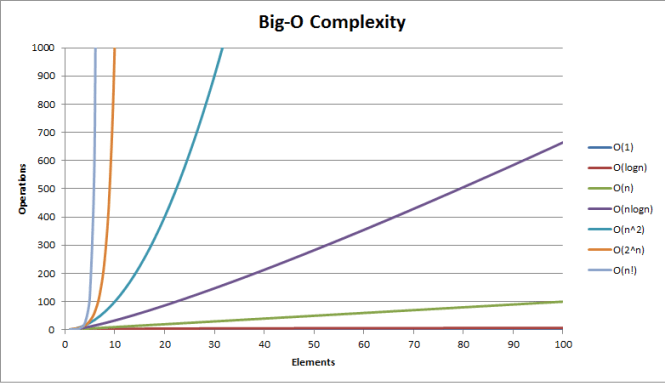




Function (n) is smaller than g(n) and g(n) is the upper bound of the f(n)

In Linear search

:Worst Case complexity : O(n)



notion of *O(n)* linear time, meaning that the size of the input affects the growth of the algorithm proportionally...and the same goes for, for example, quadratic time *O(n2)* etc..even algorithms, such as permutation generators, with *O(n!)* times, that grow by factorials.

For example, the following function is *O(n)* because the algorithm grows in proportion to its input *n*:

f(int n) {

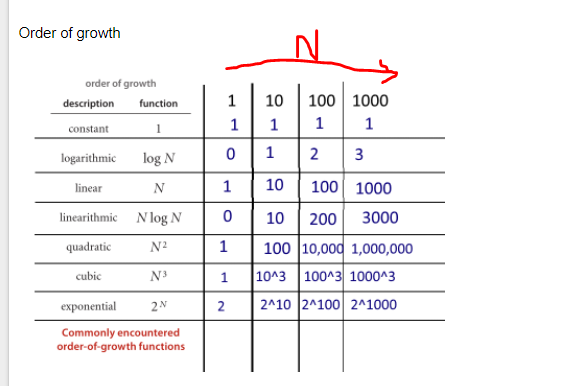
int i;

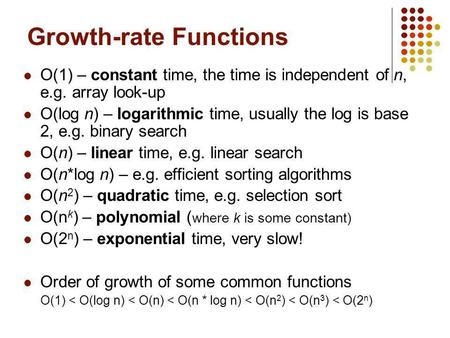
for (i = 0; i < n; ++i)

printf("%d", i);

}

Similarly, if there was a nested loop, the time would be O(n2).





Log2 (8) is written as 2 3 = 8

1 4 6 8 10 12 16 18 19

Log10 (100) is written as 2 10 =100

8 > 4 > 2 > 1

100 > 10 > 1

Best Case

Worst Case

Average Case