**TIME COMPLEXITY**

Let consider , we have 2 functions

Like f(n) and g(n) for constant c

after a particular value of constant k

* this shows Upper Bound for f(n)

Lets say f(n)=n+10

And g(n)=n

* g(n) >= c\*f(n)
* c\*n >= n+10
* n>= n+10 (for c=1) => not satisfied
  + 0 >= 10 (not possible for any value of n)
* 2\*n>= n+10 (for c=2) => not satisfied as (n should be atleast 10 else satisfied)
  + n>=10 (satisfied, possible for any value of n>=10)
    - so g(n)=2\*n , n>=10
      * where c=2,threshold k=10
    - after threshold => g(n) always greater than f(n)
* Case 2

Lets say f(n)=n^2+n+4

And g(n)=n^2

* g(n) >= c\*f(n)
* c\*(n^2) >= n^2+n+4
* 3\*n^2>= n^2+n+4 (for c=3) => satisfied
  + n>=2 (satisfied, possible for any value of n>=2)
    - so g(n)=3\*n^2 , n>=2
      * where c=3,threshold k=2
    - after threshold => g(n) always greater than f(n)

To write everything in one line

We will use Big-Oh notations

* f(n)=O(g(n))
  + Basically it checks the worst case of this function
    - i.e it shows the time complexity of worst case of your code
    - then it will take time either same or less than it

**ALGORITHM**

Suppose you are given 10 algorithm => how to find which one is best

WHICH ALGORITHM IS BEST?

An Algorithm which consumes less time and less complexity

How to figure out , which is best ?

* Just run on your system
  + i.e Experimental Analysis is not good to measure time complexity
  + Not good, as it depends on hardware specification and background processing and need to check for multiple test case
    - It is time taking and need variety of test case

We need a approach which is hardware independent, Environmental independent approach

Therefore we will use theoretical Approach

Main(){

Int n=\_\_\_\_\_\_\_

Int i= \_\_\_\_\_\_\_

While(i<=n)

{

Sop(i)

I=i=1

}

}

THEORITICAL ANALYSES:- time required to run each statement is k

Lets calculate

c1+c2+c3(n+1)+ c4n+ c5n+ c6n

(c3+c4+c5)n+ c1+ c2+ c3+c6

* a\*n + b
* This is a function to calculate time complexity of a given algorithm