# **Lecture 2-Asymptotic Analysis**

**Worst Case Analysis/Upper Bound:**

* Notation : O(Big oh)
* Equation :
  + F(n)=0(g(n))
  + F(n) is big oh of g(n)
  + F(n)<=c g(n),n>n.
  + Where c is cost ,positive value (can be in fraction)
  + **Definition:** Worst case analysis provides an upper bound on the running time of an algorithm. It represents the maximum amount of time an algorithm will take to run for any input of size n.
  + **Inequality:** F(n) <= c \* g(n), for n > n0, where c is a positive constant and n0 is a threshold value.
  + **Interpretation:** This means that the running time of the algorithm, F(n), is bounded by a constant multiple of the function g(n) for sufficiently large values of n.

**Best Case Analysis/Lower Bound:**

* Notation : Ω(Big omega)
* Equation :
  + F(n)= Ω(g(n))
  + F(n) is big omega of g(n)
  + F(n)>=c g(n),n>n.
  + Where c is cost ,positive value (can be in fraction)
  + **Definition:** Best case analysis provides a lower bound on the running time of an algorithm. It represents the minimum amount of time an algorithm will take to run for any input of size n.
  + **Inequality:** F(n) >= c \* g(n), for n > n0, where c is a positive constant and n0 is a threshold value.
  + **Interpretation:** This means that the running time of the algorithm, F(n), is bounded from below by a constant multiple of the function g(n) for sufficiently large values of n.

**Average Case Analysis/Tight Bound**:

* Notation : θ(Big theta)
* Equation :
  + F(n)= θ(g(n))
  + F(n) is big theta of g(n)
  + C2 g(n)<=f(n)<=c1 g(n),n>n.
  + Where c is cost ,positive value (can be in fraction)
  + **Definition:** Average case analysis provides a tight bound on the running time of an algorithm. It represents the average amount of time an algorithm will take to run for any input of size n.
  + **Inequality:** c2 \* g(n) <= F(n) <= c1 \* g(n), for n > n0, where c1 and c2 are positive constants and n0 is a threshold value.
  + **Interpretation:** This means that the running time of the algorithm, F(n), is bounded both from above and below by constant multiples of the function g(n) for sufficiently large values of n.

# **Step count or finding asymptotic time complexity of an algorithm**

|  |  |
| --- | --- |
| 1. For(int i=0;i<n;i++) | 1. For(int i=0;i<n;i++) |
| 1. { | 1. { |
| 1. //instructions | 1. For(int j=0;j<n;j++) |
| 1. //instructions | 1. { |
| 1. } | 1. //instructions |
|  | 1. } |
|  | 1. } |

# **Step Count or Frequency count**

|  |
| --- |
|  |
| For(int i=0;i<n;i++) |
| { |
| For(int i=0;i<m;i++) |
| { |
| cout<<”Testing \n”; |
| } |
| } |

T(n)=n+1+nm+n+nm=2nm+2n+1

O( nm )