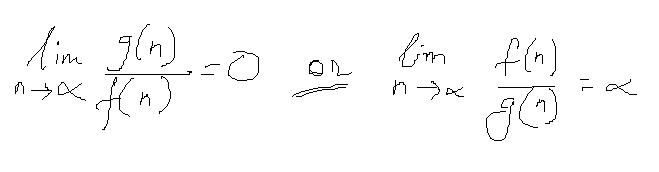
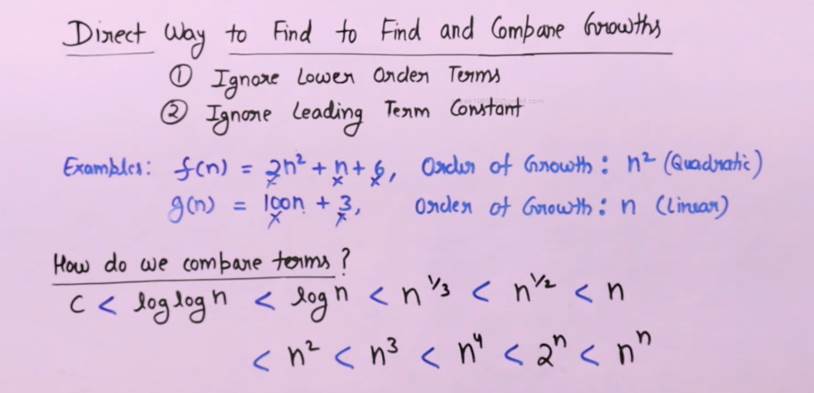
**Asymptotic Analysis:**

Order of Growth - A function f(n) is said to be growing faster/have an higher Order of Growth than g(n) if and only if



|  |  |
| --- | --- |
|  | \*where f(n) and g(n) are time taken by 2 different algorithm |

Asymptotic Analysis – It is an idea to measure order of growth. In Asymptotic analysis, we evaluate the performance of an algorithm in terms of input size i.e., if there are no input to the algorithm, it is considered to work in constant time. This analysis does not depend on users machine, programming language etc.



\*\*If OoG of one algo is nlog n and other is n then nlog n/n = log n; which means the first algo has higher OoG.

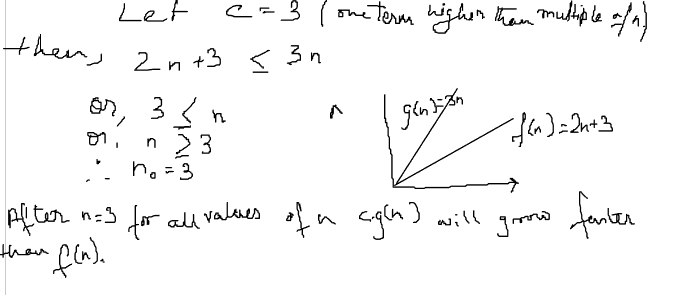
Asymptotic Notations are the mathematical notations to represent order of growth of any mathematical function. Following are the commonly used asymptotic notations to calculate the running time complexity of an algorithm.

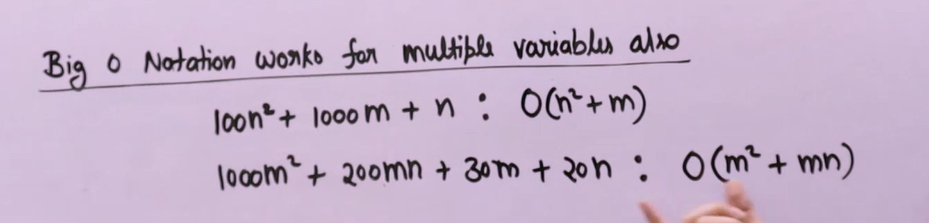
* Big Ο Notation – Exact or Upper bound
* Ω Notation/Omega – Exact or lower bound
* θ Notation/Theta – Exact bound

**Bog O Notation** - We say f(n) = O(g(n)) if and only if there exist a constant c and no such that

            f(n) <= c.g(n) for all n>=no

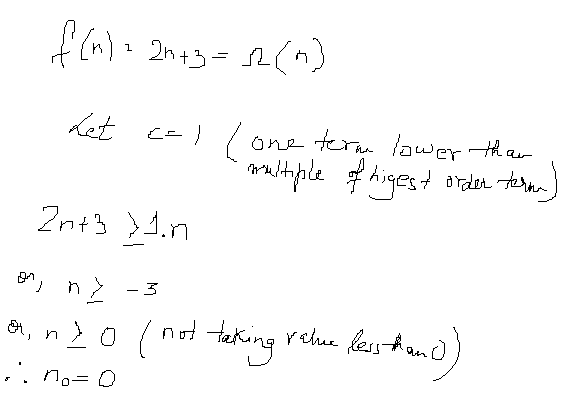
Let's assume, f(n) = 2n+3 then we know f(n) = O(n) {since n is the highest order term}





**Omega Notation** - We say f(n) = Ω(g(n)) if and only if there exist a constant c (where c>0) and no(where no>=0) such that

f(n) >= c.g(n)  for all n >= no



 If the condition is satisfied for n >= -3 the the condition will also be satisfied by n >= 0

**Theta Notation** - We say f(n) = θ(g(n)) if and only if there exist constant C1,C2(where C1>0 and C2>0) and no(where no>=0) such that

C1.g(n) <= f(n) <= C2.g(n)  for all n >= no

**Analysis of Common Loops** -

