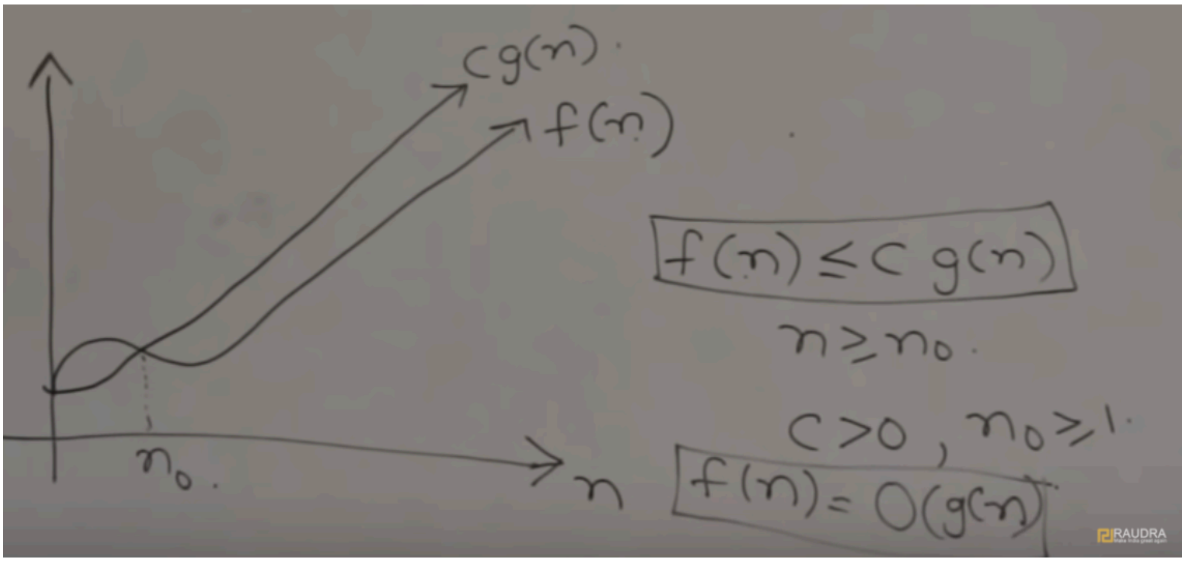
Design an Analysis of Algorithm

Notaions

Asymptotic notations

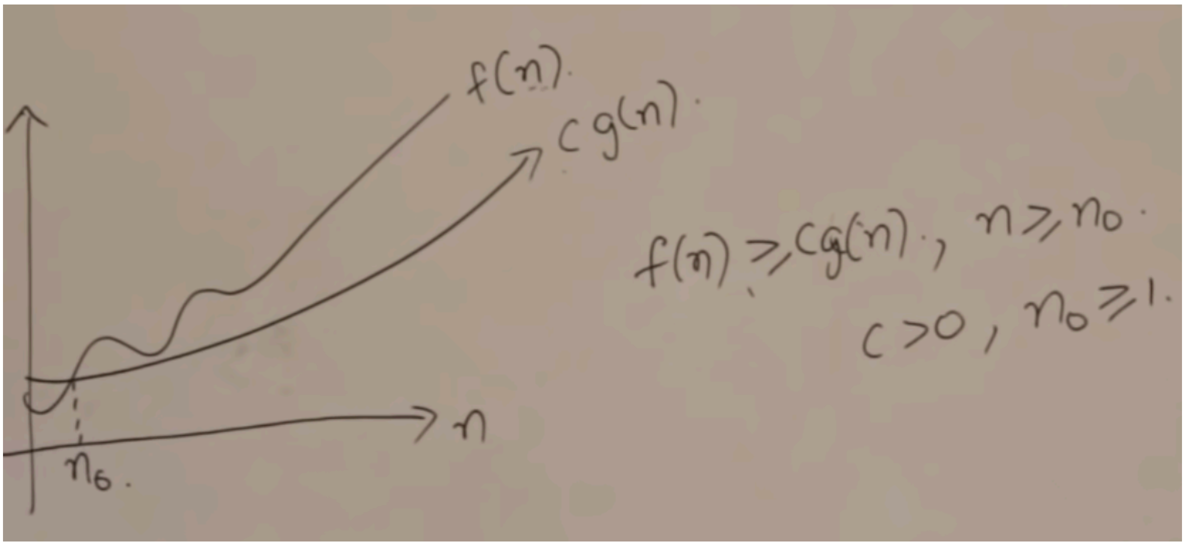
- **Big(oh) - O** : Worst case of a function, Upper bound



example: If f(n) = 3n+2 and g(n) = n prove f(n) = O(g(n)) {Find constants c and n˚}
3n+2 ≤ cn
n=1, c=1; 5≤1 => false
n=1, c=2; 5≤2 => false
n=1, c=3; 5≤3 => false
n=2, c=3; 8≤6 => false
n=3, c=4; 11≤12 => true
n=4, c=5; 14≤20 => true
Therefore, 3n+2 ≤ 4n=> ≥n2
i.e. 3n+2 ≤ O(n)
Hence proved when c=4 and n≥2.

**Practical*: time of algo will not exceed this

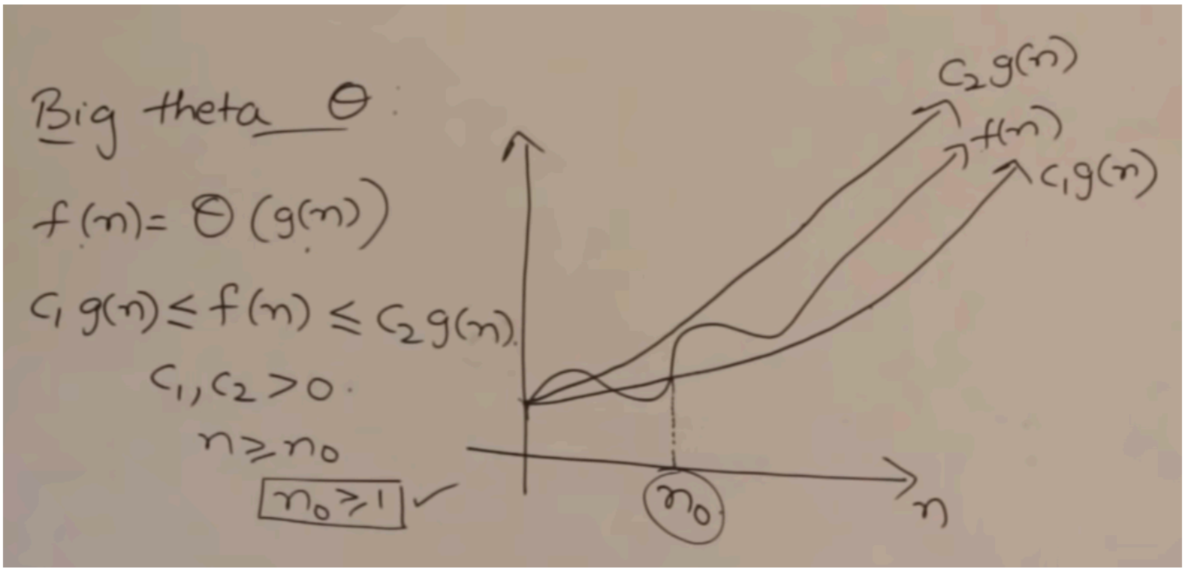
- **Big(omega) - Ω** : Best case of a function, Lower bound



example: 3n+2 and g(n) = n prove f(n) = O(g(n)) {Find constants c and n˚}
3n+2 ≥ cn
n=1, c=1; 5≥ => true
n=1, c=2; 5≥2 => true
n=1, c=3; 5≥3 => true
n=2, c=3; 8≥6 => true
n=3, c=4; 11≥12 => false
n=4, c=5; 14≥20 => false
Therefore, 3n+2 ≥3n => n≥1
i.e. 3n+2 ≥ Ω(n)
Hence proved when c=1 and n≥1.

**Practical*: also will never do better than this

- **Big (theta) - ø :** Asymptotically equal, Average case



example: 3n+2 and g(n) = n prove f(n) = O(g(n)) {Find constants c and n˚}
f(n)≤c_2 g(n)
3n+2≤4n, n˚≥2,

$f(n) \geq c_1\, g(n)$
$3n+2 \geq n,\ n^{\circ} \geq 1$

**Practical*: Average case; used when upper and lower bound are same

Example of analysis
A = [5,3,2,4,6,7,1], linear search 5
Worst case = O(n) => element is at end of array
Best case = Ω(1) => element is first element of array
Average case = ø(n/2) => ø(n) => element is at middle; will not be needed in this also as Ω and O are different

Order of complexities
- O(1) : constant
- O(logn) : logarithmic
- O(n) : linear
- O(nlogn)
- $O(n^2)$: square
- $O(n^3)$: cubical
- $O(2^n)$: exponential
- O(n!) : factorial
- $O(n^n)$