Asymptotic Notations

RM.Periakaruppan

Associate Professor

Dept. of Applied Mathematics and Computational Sciences

PSG College of Technology

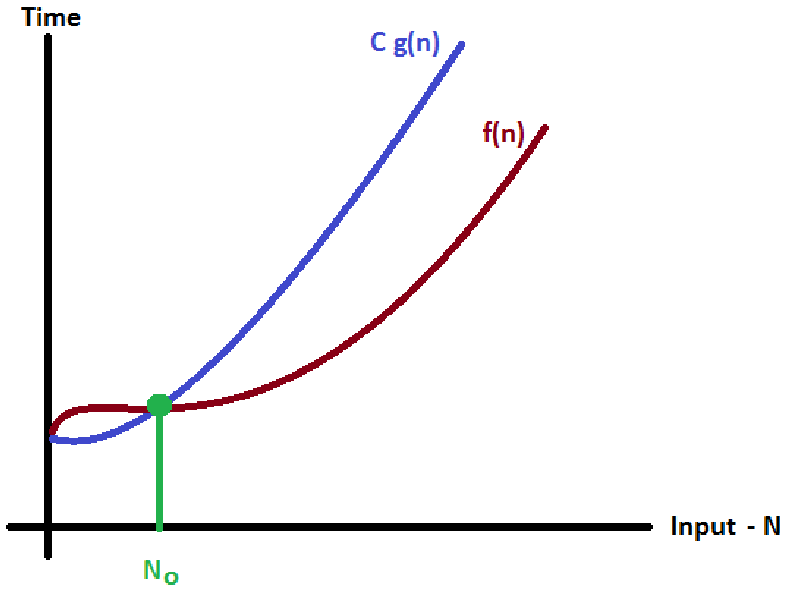
Asymptotic notations

- The following notations are used to express the time complexity of an algorithm
- Big Oh (O) – gives an upper bound of an algorithm
- Big Omega (Ω) – gives a lower bound of an algorithm
- Big Theta (Θ) – gives tight bound, where the upper bound and lower bound are same for an algorithm

Big Oh notation

Definition

f(n) = O(g(n)) means there are positive constants c and no, such that 0 ≤ f(n) ≤ cg(n) for all n ≥ no

- (ie) as n increases f(n) doesn't grows faster than g(n)
- 

Big Oh Example

- Consider the function $f(n) = 2n + 3$
- $2n + 3 <= 4n$ for every $n>=2$
- Here $f(n) = 2n + 3$, $g(n) = n$, $c = 4$, $n0=2$

Here $g(n)=n$ is an upper bound of $f(n)$ and hence $f(n)=O(g(n))= O(n)$

Since $O(n)<O(n\log n)<O(n^2)..<O(n^k)<O(2^n)<(3^n)..<O(k^n)$

It is correct to say $f(n) = O(n^2),...O(2^n)$.

However the closest upper bound is used. Hence $f(n)=O(n)$ is more meaningful.

Since $\log n < n$ it is incorrect to say $f(n)=O(\log n)$

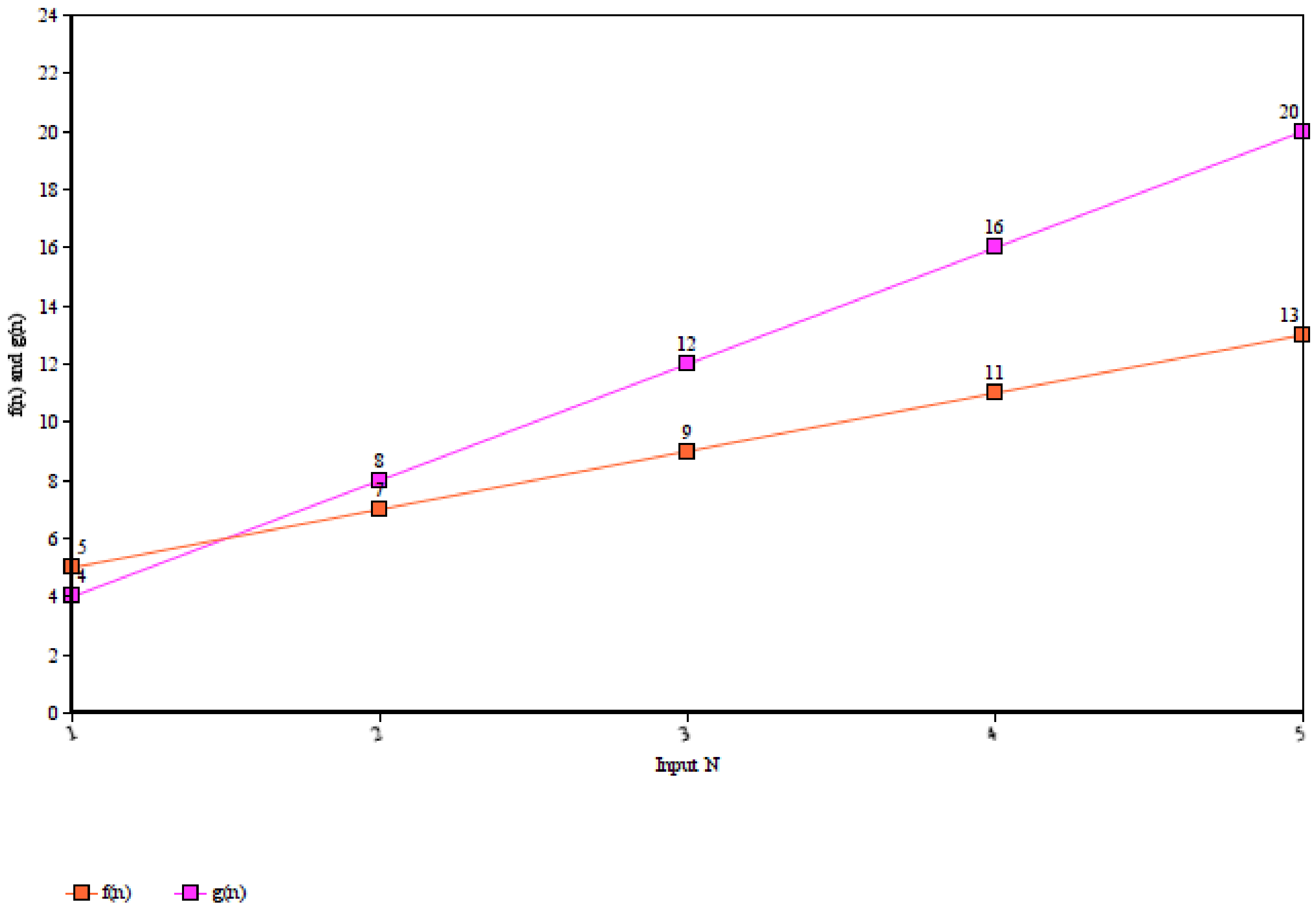
Big Oh Example-How to find Upper bound?

- Consider the function $f(n) = 2n + 3$
- $2n + 3 <= 2n + 3n$
- $2n + 3 <= 5n$ for every $n>=1$

Big Oh Example (Contd)

Input n	Frequency $f(n)=2n+3$	• $g(n)=4n$
1	5	• 4
2	7	8
3	9	12
4	11	16
5	13	20

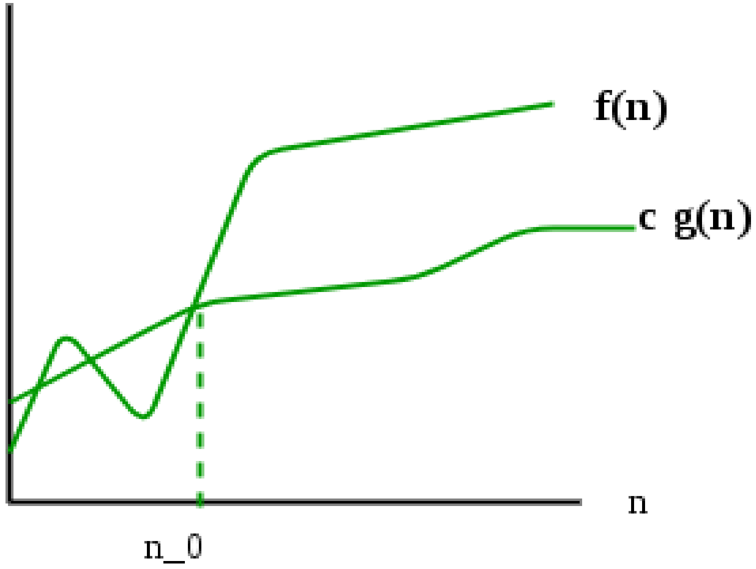
Big Oh Example (Contd)



Big Omega notation

Definition

$f(n) = \Omega(g(n))$ means there are positive constants c and no, such that $0 \leq f(n) >= cg(n)$ for all $n \geq no$

- (ie) as n increases, f(n) doesn't grows slower than g(n)
- 

f(n) = Omega(g(n))

Big Omega Example

- Consider the function $f(n) = 2n + 3$
- $2n + 3 >= n$ for every $n>=1$
- Here $f(n) = 2n + 3$, $g(n) = n$, $c = 1$, $n0=1$

Here $g(n)=n$ is an lower bound of $f(n)$ and hence $f(n)=\Omega(g(n))= \Omega(n)$

Since $1<\log n < n$

It is correct to say $f(n) = \Omega(\log n),\Omega(1)$

Since $n^2 > n$ it is incorrect to say $f(n)=\Omega(n^2)$

Big Omega Example-How to find Lower bound?

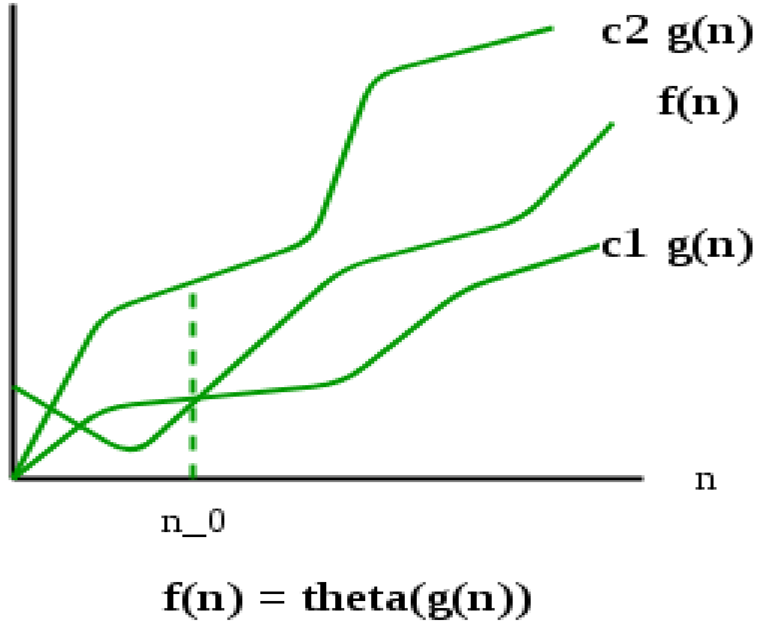
- Consider the function f(n) = 2n + 3
- 2 n + 3 >= 1 . n for all n >=1
- (Drop all lower terms and leading coefficients)
-

Theta notation

Definition

$f(n) = \Theta(g(n))$ means there are positive constants c1, c2 and no, such that $c1g(n) \leq f(n) \leq c2g(n)$ for all $n \geq no$

- (ie) as n increases f(n) grows at the same rate as g(n)



Theta Example

- Consider the function $f(n) = 2n + 3$
- $n <= 2n + 3 < = 5n$ for every $n >= 1$
- Here $f(n) = 2n + 3$, $g(n) = n$, $c1 = 1$, $c2=5$,$n0=1$

Here $g(n)=n$ is an tight bound of $f(n)$ and hence $f(n)=\Theta(g(n))= \Theta(n)$

Since $\log n \# n$ or $n^2 \# n$

It is incorrect to say $f(n) = \Theta(\log n)$ or $f(n)=\Theta(n^2)$

Example 2

- Consider the function $f(n) = n^2 \log n + n$
- Upper Bound
- $n^2 \log n + n \le n^2 \log n + n^2 \log n$
- $n^2 \log n + n \le 2 n^2 \log n$ for all $n \ge 1$
- Hence upper bound is $O(n^2 \log n)$.
- Lower Bound
- $n^2 \log n + n \ge n^2 \log n$ for all $n \ge 1$

Hence lower bound is $\Omega(n^2 \log n)$.

Tight Bound

Since $n^2 \log n \le n^2 \log n + n \le 2 n^2 \log n$ for all $n \ge 1$

We have $\Theta(n^2 \log n)$