Time Complexity of Algorithms

(Asymptotic Notations)

What is Complexity?

- The level in difficulty in solving mathematically posed problems as measured by
  - The time

    (time complexity)
  - number of steps or arithmetic operations

    (computational complexity)
  - memory space required
  - (space complexity)

Major Factors in Algorithms Design

1. Correctness

An algorithm is said to be correct if

- For every input, it halts with correct output.
- An incorrect algorithm might not halt at all OR
- It might halt with an answer other than desired one.
- Correct algorithm solves a computational problem

2. Algorithm Efficiency

Measuring efficiency of an algorithm,

- do its analysis i.e. growth rate.
- Compare efficiencies of different algorithms for the same problem.

Algorithms Growth Rate

Algorithm Growth Rates

- It measures algorithm efficiency

What means by efficient?

- If running time is bounded by polynomial in the input

Notations for Asymptotic performance

- How running time increases with input size
- O, Omega, Theta, etc. for asymptotic running time
- These notations defined in terms of functions whose domains are natural numbers
- convenient for worst case running time
- Algorithms, asymptotically efficient best choice

Complexity Analysis

- Algorithm analysis means predicting resources such as
  - computational time
  - memory
  - computer hardware etc
- Worst case analysis
  - Provides an upper bound on running time
  - An absolute guarantee
- Average case analysis
  - Provides the expected running time
  - Very useful, but treat with care: what is “average”?
    - Random (equally likely) inputs
    - Real-life inputs

Worst-case Analysis

Let us suppose that

- D_n = set of inputs of size n for the problem
- I = an element of D_n.
- t(I) = number of basic operations performed on I
- Define a function W by

  $$W(n) = \max\{t(I) \mid I \in D_n\}$$

  called the worst-case complexity of the algorithm
- W(n) is the maximum number of basic operations performed by the algorithm on any input of size n.
- Please note that the input, I, for which an algorithm behaves worst depends on the particular algorithm.

Average Complexity

- Let Pr(I) be the probability that input I occurs.
- Then the average behavior of the algorithm is defined as

  $A(n) = \Sigma Pr(I)\ t(I),$ summation over all $I \in D_n$

- We determine t(I) by analyzing the algorithm, but Pr(I) cannot be computed analytically.
- Average cost =

  $A(n) = Pr(succ)Asucc(n) + Pr(fail)Afail(n)$

- An element I in D_n may be thought as a set or equivalence class that affect the behavior of the algorithm

Worst Analysis computing average cost

- Take all possible inputs, compute their cost, take average

Asymptotic Notations Properties

- Categorize algorithms based on asymptotic growth rate e.g. linear, quadratic, polynomial, exponential
- Ignore small constant and small inputs
- Estimate upper bound and lower bound on growth rate of time complexity function
- Describe running time of algorithm as n grows to ∞.
- Describes behavior of function within the limit.

Limitations

- not always useful for analysis on fixed-size inputs.
- All results are for *sufficiently large* inputs.

Asymptotic Notations

Asymptotic Notations Θ, O, Ω, *o*, ω

- We use Θ to mean "order exactly",
- O to mean "order at most",
- Ω to mean "order at least",
- o to mean "tight upper bound",
- ω to mean "tight lower bound",

Define a ***set*** of functions: which is in practice used to compare two function sizes.

Big-Oh Notation (O)

If f, g: N $\rightarrow$ R$^+$, then we can define Big-Oh as

For a given function $g(n) \geq 0$, denoted by $O(g(n))$ the set of functions,

$O(g(n)) = \{f(n)$: there exist positive constants c and n_o such that

$0 \leq f(n) \leq cg(n)$, for all $n \geq n_o \}$

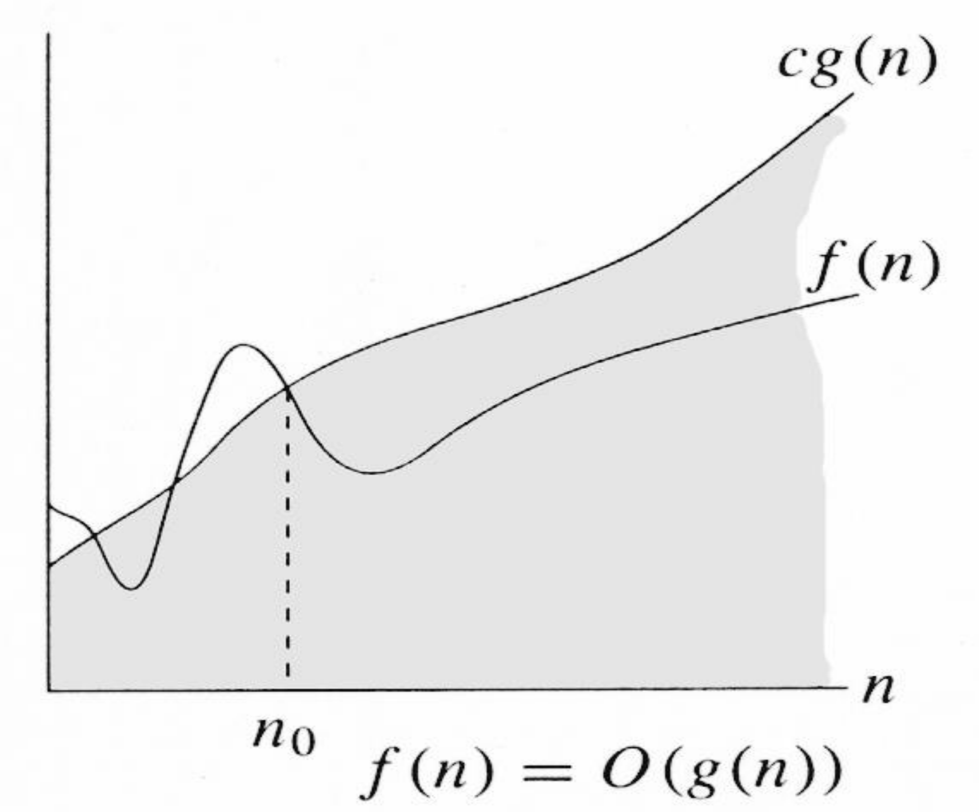
$f(n) = O(g(n))$ means function $g(n)$ is an asymptotically upper bound for $f(n)$.

We may write f(n) = O(g(n)) OR f(n) $\in$ O(g(n))

Intuitively:

Set of all functions whose *rate of growth* is the same as or lower than that of *g*(*n*).

Big-Oh Notation



$$f(n) \in O(g(n))$$

$$\exists\, c > 0, \exists\, n_0 \geq 0 \text{ and } \forall n \geq n_0,\ 0 \leq f(n) \leq c.g(n)$$

$g(n)$ is an *asymptotic upper bound* for $f(n)$.

Examples

Example 1: Prove that $2n^2 \in O(n^3)$

Proof:

Assume that $f(n) = 2n^2$, and $g(n) = n^3$

$f(n) \in O(g(n))$?

Now we have to find the existence of c and n_0

$f(n) \leq c.g(n) \rightarrow 2n^2 \leq c.n^3 \rightarrow 2 \leq c.n$

if we take, $c = 1$ and $n_0 = 2$ OR

$c = 2$ and $n_0 = 1$ then

$2n^2 \leq c.n^3$

Hence $f(n) \in O(g(n))$, $c = 1$ and $n_0 = 2$

Examples

Example 2: Prove that $n^2 \in O(n^2)$

Proof:

Assume that $f(n) = n^2$, and $g(n) = n^2$

Now we have to show that $f(n) \in O(g(n))$

Since

$f(n) \leq c.g(n) \rightarrow n^2 \leq c.n^2 \rightarrow 1 \leq c$, take, $c = 1$, $n_0 = 1$

Then

$n^2 \leq c.n^2$ for $c = 1$ and $n \geq 1$

Hence, $2n^2 \in O(n^2)$, where $c = 1$ and $n_0 = 1$

Examples

Example 3: Prove that $1000.n^2 + 1000.n \in O(n^2)$

Proof:

Assume that $f(n) = 1000.n^2 + 1000.n$, and $g(n) = n^2$

We have to find existence of c and n_0 such that

$0 \leq f(n) \leq c.g(n)$ ⍰ $n \geq n_0$

$1000.n^2 + 1000.n \leq c.n^2 = 1001.n^2$, for $c = 1001$

$1000.n^2 + 1000.n \leq 1001.n^2$

⍰ $1000.n \leq n^2$ ⍰ $n^2 \geq 1000.n$ ⍰ $n^2 - 1000.n \geq 0$

⍰ $n (n-1000) \geq 0$, this true for $n \geq 1000$

$f(n) \leq c.g(n)$ ⍰ $n \geq n_0$ and $c = 1001$

Hence $f(n) \in O(g(n))$ for $c = 1001$ and $n_0 = 1000$

Examples

Example 4: Prove that $n^3 \notin O(n^2)$

Proof:

On contrary we assume that there exist some positive constants c and n_0 such that

$0 \leq n^3 \leq c.n^2 \quad \forall\, n \geq n_0$

$0 \leq n^3 \leq c.n^2 \Rightarrow n \leq c$

Since c is any fixed number and n is any arbitrary constant, therefore $n \leq c$ is not possible in general.

Hence our supposition is wrong and $n^3 \leq c.n^2$, $\forall\, n \geq n_0$ is not true for any combination of c and n_0.

And hence, $n^3 \notin O(n^2)$

Some More Examples

1. $n^2 + n^3 = O(n^4)$
2. $n^2 / \log(n) = O(n \,.\, \log n)$
3. $5n + \log(n) = O(n)$
4. $n^{\log n} = O(n^{100})$
5. $3^n = O(2^n \,.\, n^{100})$
6. $n! = O(3^n)$
7. $n + 1 = O(n)$
8. $2n+1 = O(2n)$
9. $(n+1)! = O(n!)$
10. $1 + c + c^2 + \ldots + c^n = O(c^n)$ for $c > 1$
11. $1 + c + c^2 + \ldots + c^n = O(1)$ for $c < 1$

Big-Omega Notation (Ω)

If f, g: N $\rightarrow$ R$^+$, then we can define Big-Omega as

For a given function $g(n)$ denote by $\Omega(g(n))$ the set of functions,

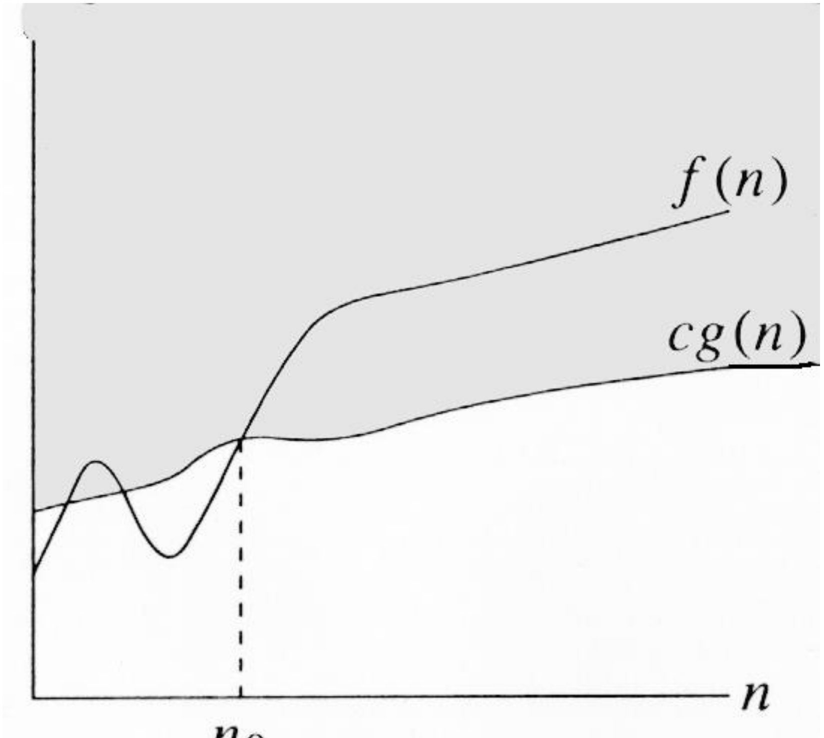
$\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_o such that $0 \le cg(n) \le f(n)$ for all $n \ge n_o\}$

$f(n) = \Omega(g(n))$, means that function $g(n)$ is an asymptotically lower bound for $f(n)$.

We may write $f(n) = \Omega(g(n))$ OR $f(n) \in \Omega(g(n))$

Intuitively:
Set of all functions whose *rate of growth* is the same as or higher than that of $g(n)$.

Big-Omega Notation



$f(n) = \Omega(g(n))$

$f(n) \in \Omega(g(n))$

$\exists\ c > 0, \exists\ n_0 \geq 0\ , \forall n \geq n_0, f(n) \geq c.g(n)$

***g*(*n*) is an *asymptotically lower bound* for *f*(*n*).**

Examples

Example 1: Prove that $5.n^2 \in \Omega(n)$

Proof:

Assume that $f(n) = 5.n^2$, and $g(n) = n$

$f(n) \in \Omega(g(n))$?

We have to find the existence of c and n_0 s.t.

$c.g(n) \leq f(n)$ for all $n \geq n_0$

$c.n \leq 5.n^2 \Rightarrow c \leq 5.n$

if we take, $c = 5$ and $n_0 = 1$ then

$c.n \leq 5.n^2$ for all $n \geq n_0$

And hence $f(n) \in \Omega(g(n))$, for $c = 5$ and $n_0 = 1$

Examples

Example 2: Prove that $5.n + 10 \in \Omega(n)$

Proof:

Assume that $f(n) = 5.n + 10$, and $g(n) = n$

$f(n) \in \Omega(g(n))$?

We have to find the existence of c and n_0 s.t.

$c.g(n) \leq f(n)$ for all $n \geq n_0$

$c.n \leq 5.n + 10 \Rightarrow c.n \leq 5.n + 10.n \Rightarrow c \leq 15.n$

if we take, $c = 15$ and $n_0 = 1$ then

$c.n \leq 5.n + 10$ for all $n \geq n_0$

And hence $f(n) \in \Omega(g(n))$, for $c = 15$ and $n_0 = 1$

Examples

Example 3: Prove that $100.n + 5 \notin \Omega(n^2)$

Proof:

Let $f(n) = 100.n + 5$, and $g(n) = n^2$

Assume that $f(n) \in \Omega(g(n))$?

Now if $f(n) \in \Omega(g(n))$ then there exist c and n_0 s.t.

$c.g(n) \leq f(n)$ for all $n \geq n_0$ $\Rightarrow$

$c.n^2 \leq 100.n + 5 \Rightarrow$ $c.n \leq 100 + 5/n$ $\Rightarrow$

$n \leq 100/c$, for a very large n, which is not possible

And hence $f(n) \notin \Omega(g(n))$

Theta Notation (Θ)

If f, g: N $\rightarrow$ R$^+$, then we can define Big-Theta as

For a given function $g(n)$ denoted by $\Theta(g(n))$ the set of functions,

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2 and n_o such that

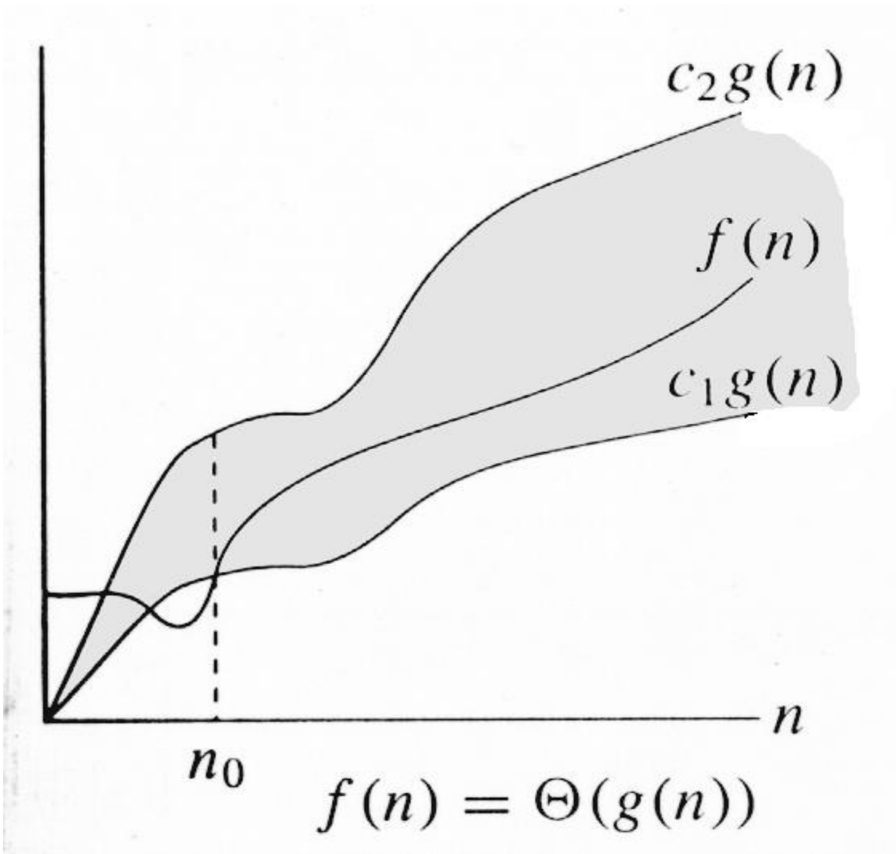
$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_o \}$

$f(n) = \Theta(g(n))$ means function $f(n)$ is equal to $g(n)$ to within a constant factor, and $g(n)$ is an asymptotically tight bound for $f(n)$.

We may write f(n) = Θ(g(n)) OR f(n) $\in$ Θ(g(n))

Intuitively: Set of all functions that have same *rate of growth* as *g*(*n*).

Theta Notation



$f(n) \in \Theta(g(n))$

$\exists\ c_1 > 0,\ c_2 > 0,\ \exists\ n_0 \geq 0,\ \forall\ n \geq n_0,\ c_2.g(n) \leq f(n) \leq c_1.g(n)$

***We say that g*(*n*) is an *asymptotically tight bound* for *f*(*n*).**

Theta Notation

Example 1: Prove that $\frac{1}{2}.n^2 - \frac{1}{2}.n = \Theta(n^2)$

Proof

Prove that $\frac{1}{2}.n^2 - \frac{1}{2}.n = \Theta(n^2)$

Assume

$f(n) = \frac{1}{2}.n^2 - \frac{1}{2}.n$

$g(n) = n^2$

$c_1.g(n) \leq f(n) \leq c_2.g(n) \; \forall n \geq n_o$

$f(n) \leq c_2.g(n)$

$\frac{1}{2}.n^2 - \frac{1}{2}.n \geq \frac{1}{2}.n^2 - \frac{1}{2}.n. \frac{1}{2}.n \geq \frac{1}{2}.n^2 - 1/4.n^2 = -1/2n^2$

Assume that $f(n) = \frac{1}{2}.n^2 - \frac{1}{2}.n$, and $g(n) = n^2$

$f(n) \in \Theta(g(n))$?

We have to find the existence of c_1, c_2 and n_0 s.t.

$c_1.g(n) \leq f(n) \leq c_2.g(n)$ for all $n \geq n_0$

Since, $\frac{1}{2} n^2 - \frac{1}{2} n \leq \frac{1}{2} n^2$ $\forall n \geq 0$ if $c_2 = \frac{1}{2}$ and

$\frac{1}{2} n^2 - \frac{1}{2} n \geq \frac{1}{2} n^2 - \frac{1}{2} n . \frac{1}{2} n \; (\forall n \geq 2) = \frac{1}{4} n^2$, $c_1 = \frac{1}{4}$

Hence $\frac{1}{2} n^2 - \frac{1}{2} n \leq \frac{1}{2} n^2 \leq \frac{1}{2} n^2 - \frac{1}{2} n$

$c_1.g(n) \leq f(n) \leq c_2.g(n)$ $\forall n \geq 2$, $c_1 = \frac{1}{4}$, $c_2 = \frac{1}{2}$

Hence $f(n) \in \Theta(g(n)) \Rightarrow \frac{1}{2}.n^2 - \frac{1}{2}.n = \Theta(n^2)$

Theta Notation

Example 2: Prove that $a.n^2 + b.n + c = \Theta(n^2)$ where a, b, c are constants and $a > 0$

Proof

If we take $c_1 = \frac{1}{4}.a$, $c_2 = 7/4.a$ and

$$n_0 = 2.\max((|b|/a), \sqrt{(|c|/a)})$$

Then it can be easily verified that

$0 \leq c_1.g(n) \leq f(n) \leq c_2.g(n), \forall n \geq n_0$, $c_1 = \frac{1}{4}.a$, $c_2 = 7/4.a$

Hence $f(n) \in \Theta(g(n)) \Rightarrow a.n^2 + b.n + c = \Theta(n^2)$

Hence any polynomial of degree 2 is of order $\Theta(n^2)$

Theta Notation

Example 1: Prove that $2.n^2 + 3.n + 6 \notin \Theta(n^3)$

Proof: Let $f(n) = 2.n^2 + 3.n + 6$, and $g(n) = n^3$

we have to show that $f(n) \notin \Theta(g(n))$

On contrary assume that $f(n) \in \Theta(g(n))$ i.e.

there exist some positive constants c_1, c_2 and n_0 such that: $c_1.g(n) \leq f(n) \leq c_2.g(n)$

$c_1.g(n) \leq f(n) \leq c_2.g(n) \Rightarrow c_1.n^3 \leq 2.n^2 + 3.n + 6 \leq c_2.\ n^3 \Rightarrow$

$c_1.n \leq 2 + 3/n + 6/n^2 \leq c_2.\ n \Rightarrow$

$c_1.n \leq 2 \leq c_2.\ n$, for large $n \Rightarrow$

$n \leq 2/c_1 \leq c_2/c_1.n$ which is not possible

Hence $f(n) \notin \Theta(g(n)) \Rightarrow 2.n^2 + 3.n + 6 \notin \Theta(n^3)$

Little-Oh Notation

o-notation is used to denote a upper bound that is not asymptotically tight.

For a given function $g(n) \geq 0$, denoted by $o(g(n))$ the set of functions,

$$o(g(n)) = \left\{ \begin{array}{l} f(n): \text{for any positive constants } c, \text{ there exists a constant } n_o \\ \text{such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_o \end{array} \right\}$$

f(n) becomes insignificant relative to g(n) as n approaches infinity

$$\text{e.g., } 2\text{n} = o(n^2) \text{ but } 2n^2 \neq o(n^2)..$$

$$\lim_{\text{n}\to\infty} \frac{f(n)}{g(n)} = 0$$

g(n) is an upper bound for f(n), not asymptotically tight

Examples

Example 1: Prove that $2n^2 \in o(n^3)$

Proof:

Assume that $f(n) = 2n^2$, and $g(n) = n^3$

$f(n) \in o(g(n))$?

Now we have to find the existence n_0 for any c

$f(n) < c.g(n)$ this is true

$\Rightarrow 2n^2 < c.n^3 \Rightarrow 2 < c.n$

This is true for any c, because for any arbitrary c we can choose n_0 such that the above inequality holds.

Hence $f(n) \in o(g(n))$

Examples

Example 2: Prove that $n^2 \notin o(n^2)$

Proof:

Assume that $f(n) = n^2$, and $g(n) = n^2$

Now we have to show that $f(n) \notin o(g(n))$

Since

$f(n) < c.g(n)$ ⍰ $n^2 < c.n^2$ ⍰ $1 \leq c$,

In our definition of small o, it was required to prove for any c but here there is a constraint over c .

Hence, $n^2 \notin o(n^2)$, where $c = 1$ and $n_0 = 1$

Examples

Example 3: Prove that $1000.n^2 + 1000.n \notin o(n^2)$

Proof:

Assume that $f(n) = 1000.n^2 + 1000.n$, and $g(n) = n^2$

we have to show that $f(n) \notin o(g(n))$ i.e.

We assume that for any c there exist n_0 such that

$0 \leq f(n) < c.g(n) \quad \forall\, n \geq n_0$

$1000.n^2 + 1000.n < c.n^2$

If we take $c = 2001$, then, $1000.n^2 + 1000.n < 2001.n^2$

$\Rightarrow 1000.n < 1001.n^2$ which is not true

Hence $f(n) \notin o(g(n))$ for $c = 2001$

Little-Omega Notation

Little-ω notation is used to denote a lower bound that is not asymptotically tight.

For a given function $g(n)$, denote by $\omega(g(n))$ the set of all functions.

$\omega(g(n)) = \{f(n)$: for any positive constants c, there exists a constant n_o such that $0 \le cg(n) < f(n)$ for all $n \ge n_o\}$

f(*n*) becomes arbitrarily large relative to *g*(*n*) as *n* approaches infinity

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

e.g., $\frac{n^2}{2} = \omega(n)$ but $\frac{n^2}{2} \neq \omega(n^2)$..

Examples

Example 1: Prove that $5.n^2 \in \omega(n)$

Proof:

Assume that $f(n) = 5.n^2$, and $g(n) = n$

$f(n) \in \Omega(g(n))$?

We have to prove that for any c there exists n_0 s.t.,

$c.g(n) < f(n)$ for all $n \geq n_0$

$c.n < 5.n^2 \Rightarrow c < 5.n$

This is true for any c, because for any arbitrary c e.g. c = 1000000, we can choose n_0 = 1000000/5 = 200000 and the above inequality does hold.

And hence $f(n) \in \omega(g(n))$,

Examples

Example 2: Prove that $5.n + 10 \notin \omega(n)$

Proof:

Assume that $f(n) = 5.n + 10$, and $g(n) = n$

$f(n) \notin \Omega(g(n))$?

We have to find the existence n_0 for any c, s.t.

$c.g(n) < f(n) \quad \forall\, n \geq n_0$

$c.n < 5.n + 10$, if we take $c = 16$ then

$16.n < 5.n + 10 \Leftrightarrow 11.n < 10$ is not true for any positive integer.

Hence $f(n) \notin \omega(g(n))$

Examples

Example 3: Prove that $100.n \notin \omega(n^2)$

Proof:

Let $f(n) = 100.n$, and $g(n) = n^2$

Assume that $f(n) \in \omega(g(n))$

Now if $f(n) \in \omega(g(n))$ then there n_0 for any c s.t.

$c.g(n) < f(n) \quad \forall\, n \geq n_0$ this is true

$\Rightarrow c.n^2 < 100.n \Rightarrow c.n < 100$

If we take $c = 100$, $n < 1$, not possible

Hence $f(n) \notin \omega(g(n))$ i.e. $100.n \notin \omega(n^2)$

Usefulness of Notations

- It is not always possible to determine behaviour of an algorithm using Θ-notation.
- For example, given a problem with n inputs, we may have an algorithm to solve it in $a.n^2$ time when n is even and c.n time when n is odd. OR
- We may prove that an algorithm never uses more than $e.n^2$ time and never less than f.n time.
- In either case we can neither claim $\Theta(n)$ nor $\Theta(n^2)$ to be the order of the time usage of the algorithm.
- Big O and Ω notation will allow us to give at least partial information