Asymptotic analysis

- **Asymptotic analysis** refers to computing the running time of any operation in mathematical units of computation.
- **Asymptotic analysis** is input bound i.e., if there's no input to the algorithm, it is concluded to work in a constant time. Other than the "input" all other factors **are** considered constant.
- The goal is to determine the best case, worst case and average case time **required** to execute a given task.

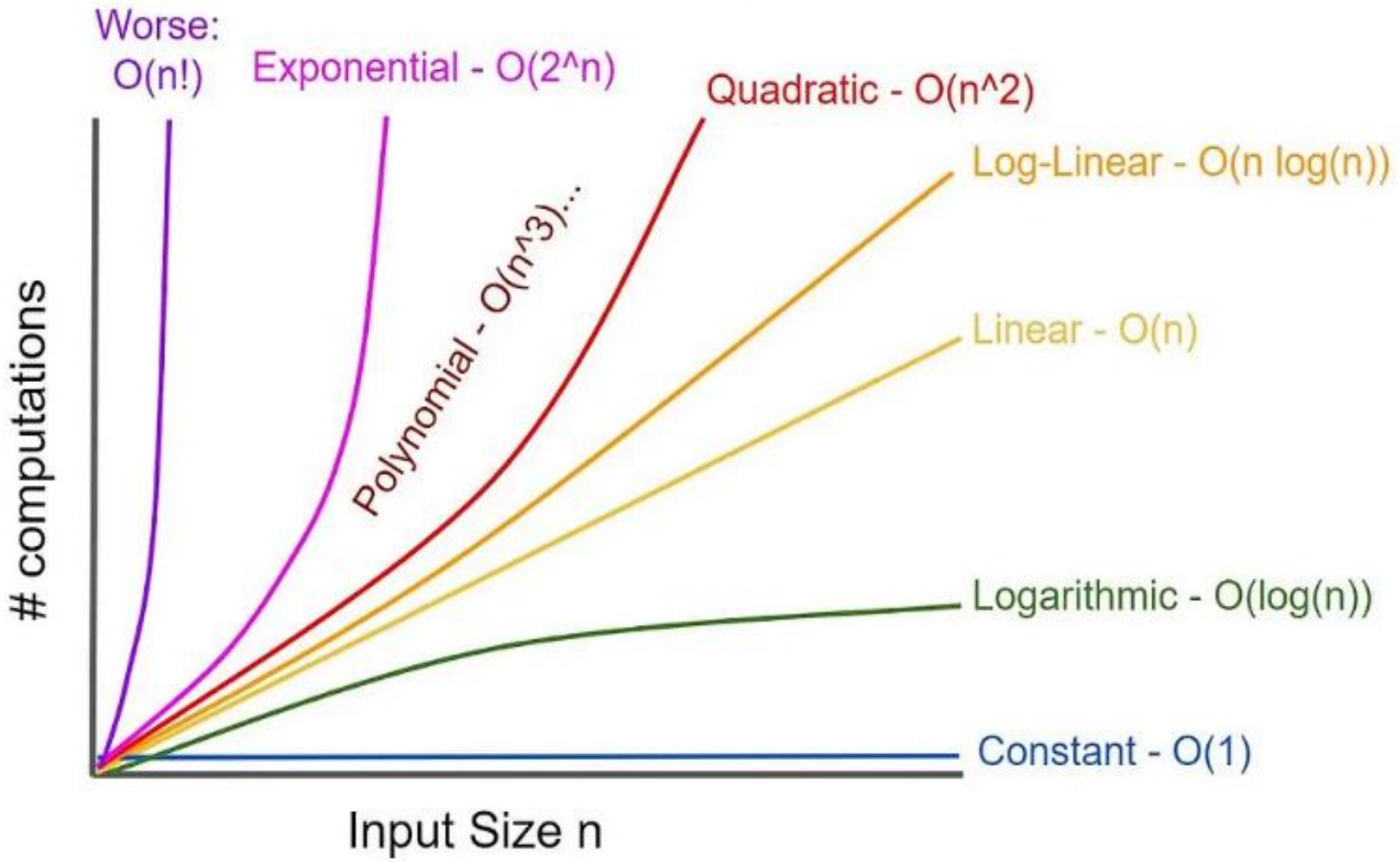
Asymptotic notation

- It describes the rate of growth of functions
- It is a way to compare the sizes of functions
- Focus on what is important by abstracting low order terms and constant factors.
- Allows us to evaluate the speed of an algorithm independent of the hardware/software environment

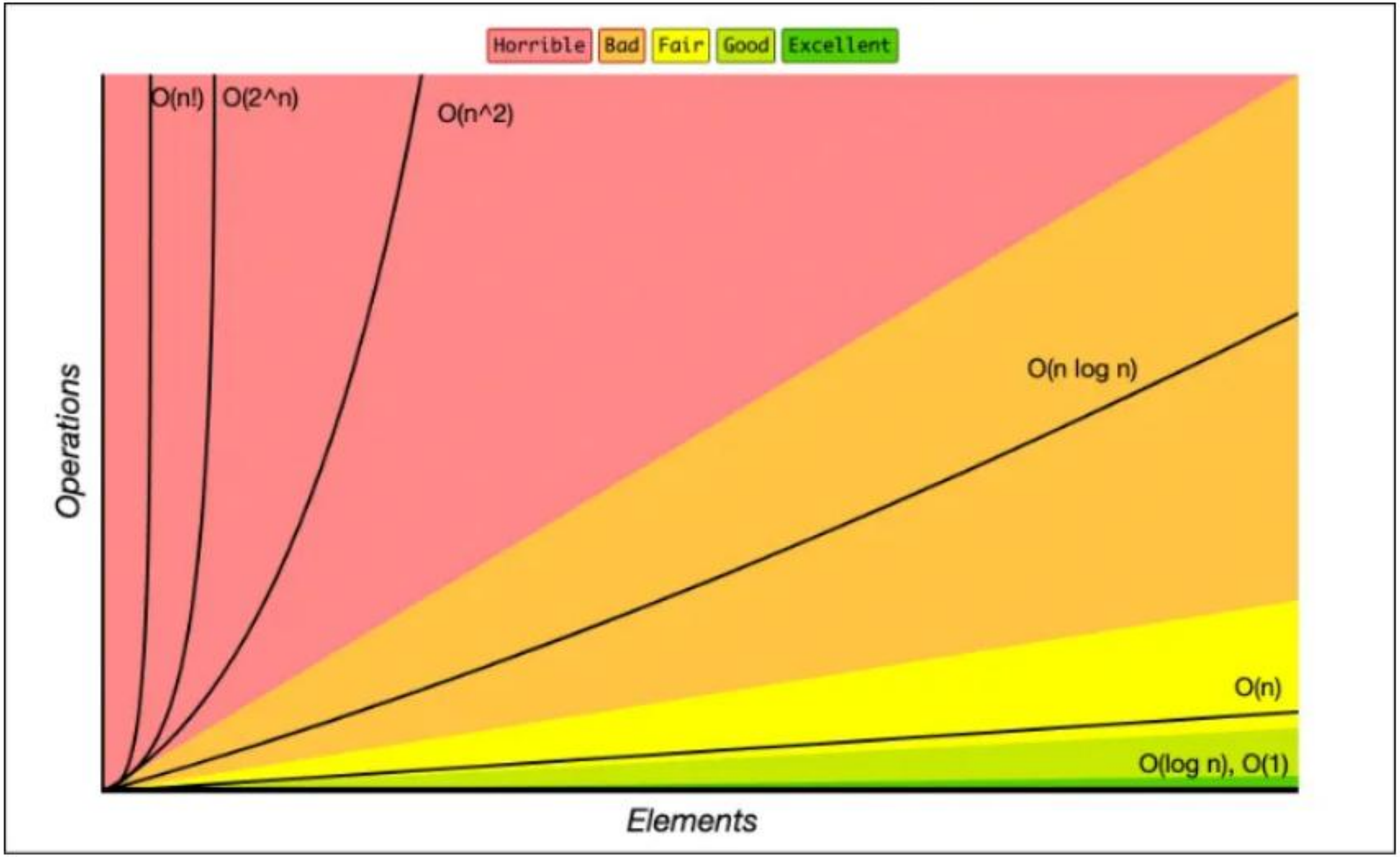
Types of Asymptotic Notation

1. **Big-O Notation** (O) – **Big O notation** describes worst case scenario.
2. Omega **Notation** (Ω) – Omega(Ω) **notation** describes best case scenario.
3. Theta **Notation** (θ) – This **notation** represents the average complexity of an algorithm.

- **Best case** is the function which performs the minimum number of steps on input data of n elements.
- Worst **case** is the function which performs the maximum number of steps on input data of size n.
- Average **case** is the function which performs an average number of steps on input data of n elements.



Time Complexities Graph



Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

- **O(1)** — Excellent/Best
- **O(log n)** — Good
- **O(n)** — Fair
- **O(n log n)** — Bad
- **O(n^2)**, **O(2^n)** and **O(n!)** — Horrible/Worst

- The growth rate is not affected by
  - constant factors or
  - lower-order terms
- Examples
  - $10^2 n + 10^5$ is a linear function
  - $10^5 n^2 + 10^8 n$ is a quadratic function

- $10n^2+3n+2$ -$O(n^2)$
- $4n+3\log n$ - $O(n)$
- $n\log n+3n+\log n$- $O(n\log n)$
- $3n^2+2n^3+8n+4$- $O(n^3)$

- **Growth rates of functions:**
  - **Linear** $\approx \boldsymbol{n}$
  - **Quadratic** $\approx \boldsymbol{n}^2$
  - **Cubic** $\approx \boldsymbol{n}^3$

Constant –O(1)

```
int square(int a)
{
        return a*a;
}
```

```
int sum(int a, int int b)
{
    return a+b;
}
```

Types of loops

- Linear
- Logarithmic
- Nested loops
  - Quadratic
  - Dependent
  - Linear logarithmic
  - Cubic

Linear

```
i=1
while(i<=N)
    -------
    i=i+1
```

```
for(i=0; i < N; i++)
{
  statement;
}
```

```
i=1
while(i<=N)
    -------
    i=i+2
```

```
int sum(int A[], int n)
{
   int sum = 0, i;
   for(i = 0; i < n; i++)
      sum = sum + A[i];
   return sum;
}
```

Logarithmic

```
i=1
while(i<=100)
     -------
     i=i*2
```

```
i=100
while(i>=1)
     -------
     i=i/2
```

O(logn)

Quadratic

- $O(n^2)$

```
for(i=0; i < N; i++)
{
  for(j=0; j < N;j++)
  {
    statement;
  }
}
```

Linear logarithmic

```
 i=1
While(i<=N)
    j=1
    while(j<=N)
            -----
            j=j*2
     i=i+1
```

O(n logn)

Dependent loop

```
for(i=1; i<=n;i++)
   for(j=i+1;j<=n;j++)
     ----
```

$O(n^{2)}$

cubic

```
for(i=0; i<n;i++)
   for(j=0;j<n;j++)
         for(k=0;k<n;k++)
                  ----
```

$O(n^3)$

Properties of Asymptotic Notations

- If **$f(n) = O(g(n))$**, then there exists positive constants c, n0 such that **$0 \leq f(n) \leq c.g(n)$**, for all $n \geq n0$

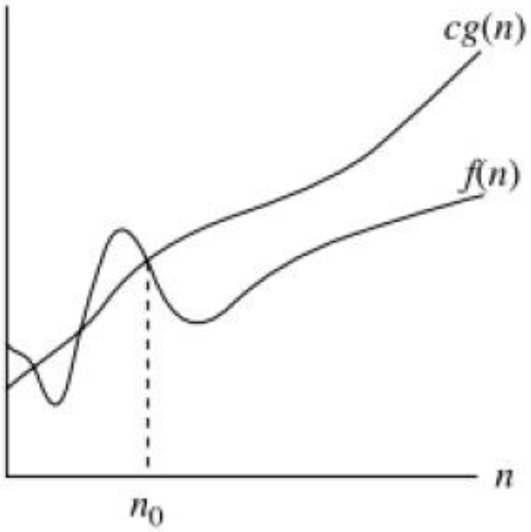
- If **$f(n) = \Omega(g(n))$**, then there exists positive constants c, n0 such that **$0 \leq c.g(n) \leq f(n)$**, for all $n \geq n0$

- If **$f(n) = \Theta(g(n))$**, then there exists positive constants c1, c2, n0 such that **$0 \leq c1.g(n) \leq f(n) \leq c2.g(n)$**, for all $n \geq n0$

Big O – gives upper bound i.e. max value specify the worst case

O-notation

$$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \} .$$

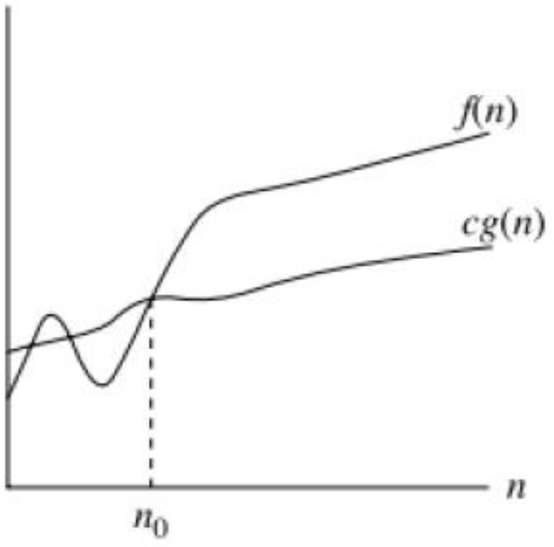


$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

Omega – gives lower bound i.e. min value specify the best case

Ω-notation

$$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0\} .$$

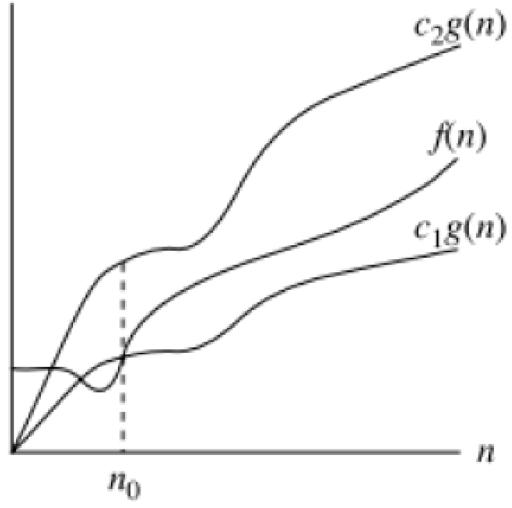


$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Theta – gives average value

Θ-notation

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\} .$$



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

Given functions $f(n)$ and $g(n)$, we say that $f(n)$ is $O(g(n))$ if there are positive constants
c and n_0 such that

$$f(n) \le cg(n) \quad \text{for } n \ge n_0$$

Example: $2n + 3$ is $O(n)$ Prove it

2n+3<=c. n

n=1, 5<=c. 1 c=5,6,7…

n=2, 7<=c.2 c= 4,5,6…

n=3, 9 <=c.3 c=3,4,5…

n=4, 11<=c.4 c=3,4,5…

n=5, 13<=c.5 c=3,4,5…

2n+3 <= 3. n for n>=3

2n+3<=5.n for n>=1

Given functions $f(n)$ and $g(n)$, we say that $f(n)$ is Ω ($g(n)$) if there are positive constants
c and n_0 such that

$$f(n) >= cg(n) \quad \text{for } n \geq n_0$$

Example: $2n + 3$ is Ω (n) Prove it

2n+3>=c.n

n=1 5>=c.1 c=5,4,3,2,1

n=2 7>=c.2 c=3,2,1

n=3 9>=c.3 c=3,2,1

n=5 13>=c.5 c=2,1

n=10 23>=c.10 c=2,1

n=50 103>=c.50 c=2,1

2n+3>=2.n for n>=1

- **$7n - 3$ is $O(n)$ Prove it**
- **$7n-3$ is $\Omega(n)$ prove it**

$7n - 3$ is $O(n)$

$f(n) <= c.g(n)$

$7n-3 <= c.n$

$n=1 \quad 4 <= c.1 \quad c=4,5,6\ldots$

$n=2 \quad 11 < = c.2 \quad c=6,7,8..$

$n=50 \quad 347 <= c.50 \quad c=7, 8,9$

$7n-3 <= 7.n$ for $n >= 1$

7n-3 is Ω (*n*) prove it

$f(n) >= cg(n)$ for $n \geq n_0$

7n-3>=c.n

N=1

4>=c.1 , c=4,3,2,1

N=2

11>=c.2, c=5,4,3,2,1

N=3

18>=c.3, c=6,5,4,….

N=5

32>=c.5, c=6,5,4,3….

7n-3>=4n for n>=1

7n-3>=6n for n>=3

- $3n^2+5n+2$ is $O(n^2)$
- $3n^2+5n+2$ is not $O(n)$

- $N=4$
- $70 \le C.\ 16$, $c=5,6$ $n \ge 3$
- $3n^2+5n+2 \le 5n^2$, $n \ge 3$

Little oh ‘o’

- We formally define $o(g(n))$ (little-oh of g of n) as the set $f(n) = o(g(n))$ for any positive constant $c > 0$ and there exists a value $n_0 > 0$ such that 0⩽f(n)⩽c.g(n) for all $n >= n_0$.
- Let us consider the function f$(n) = 4.n^3 + 10.n^2 + 5.n + 1$
- $g(n) = n^4$
- Hence, the complexity of f(n) can be represented as o(g(n)) , i.e. $o(n^4)$.

little-omega ‘ω’

- We formally define $\omega(g(n))$ (little-oh of g of n) as the set $f(n) \;=\; \omega(g(n))$ for any positive constant $c > 0$ and there exists a value $n_0 > 0$ such that $0 \geq f(n) \geq c.\,g(n)$ for all $n >= \; n_0$.

- Let us consider the same function f$(n) = 4.\,n^3 + 10.\,n^2 + 5.\,n + 1$
- $g(n) = n^2$
- The complexity of f(n) can be represented as ω(g(n)) , i.e. $\omega(n^2)$.