Data Structures
Chapter 1

Revisit – Step Count

- Why step count?
- It is to compare the **time complexities** of two programs that compute the same function and also to predict the **growth rate** in run time.

- **Example**: Let's compute the step count for three programs and compare their time complexities.

1. $T_{add}(n)$ – adding two numbers
2. $T_{sum}(n)$ – adding list of numbers
3. $T_{mtx}(n)$ – adding two matrix

Revisit – Step Count

Program add	step count

`float add(int a, int b) {`
`  return    a + b;`
`}`
	
	1

Program sum of list	step count

`float sum(float list[], int n) {`
`  float total = 0;`
`  int i;`
`  for (i=0; i<n; i++)`
`    total += list[i];`
`  return total;`
`}`
	
	1
	
	n + 1
	n
	1

Program sum of matrix	step count

`void add(int a[][MAX_SIZE], int b[][MAX_SIZE],`
`         int c[][MAX_SIZE}, int rows, int cols) {`
`  for(int i=0; i<rows; i++)`
`    for(iny j=0; j<cols; j++)`
`      c[i][j] = a[i][j] + b[i][j];`
`}`
	
	rows + 1
	rows * (cols+1)
	rows * cols

Revisit – Step Count

- $\boldsymbol{T_{add}(n)} = 2$

- $\boldsymbol{T_{sum}(n)} = 1 + 2(n+1) + 2n + 1 = 4n + 4$
  $= c * n + c'$

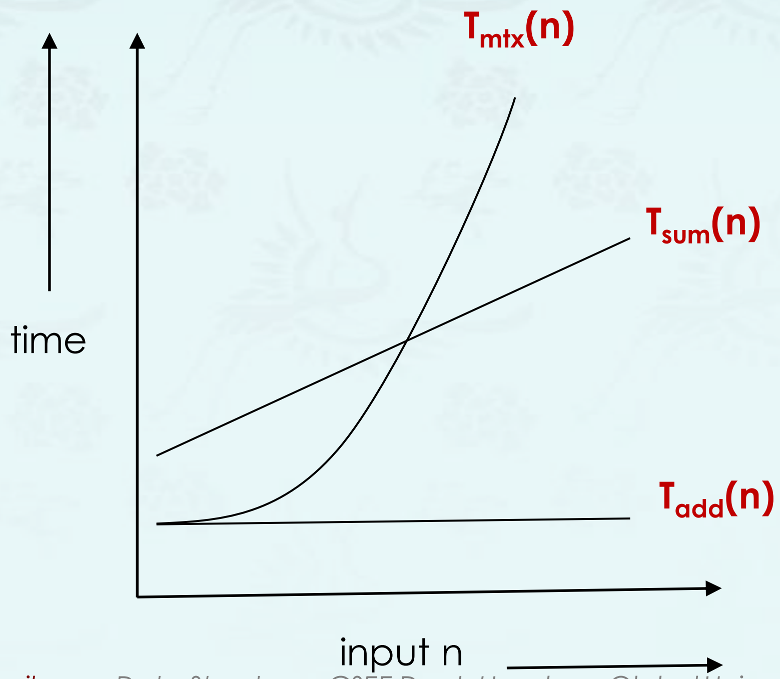
- $\boldsymbol{T_{mtx}(n)} = 2\,rows * cols + 2\,rows + 1$
  $= a * n^2 + b * n + c$

Revisit – Step Count

- $\boldsymbol{T_{add}(n)} = 2$ → O(1)
- $\boldsymbol{T_{sum}(n)} = 1 + 2(n+1) + 2n + 1 = 4n + 4$
  $= c * n + c'$ → O(n)
- $\boldsymbol{T_{mtx}(n)} = 2\,rows * cols + 2\,rows + 1$
  $= a * n^2 + b * n + c$ → O(n^2)

Revisit – Step Count

- $T_{add}(n) = 2$ → O(1)
- $T_{sum(n)} = 1 + 2(n+1) + 2n + 1 = 4n + 4$ → O(n)
  $= c * n + c'$
- $T_{mtx(n)} = 2\,rows * cols + 2\,rows + 1$ → O(n²)
  $= a * n^2 + b * n + c$

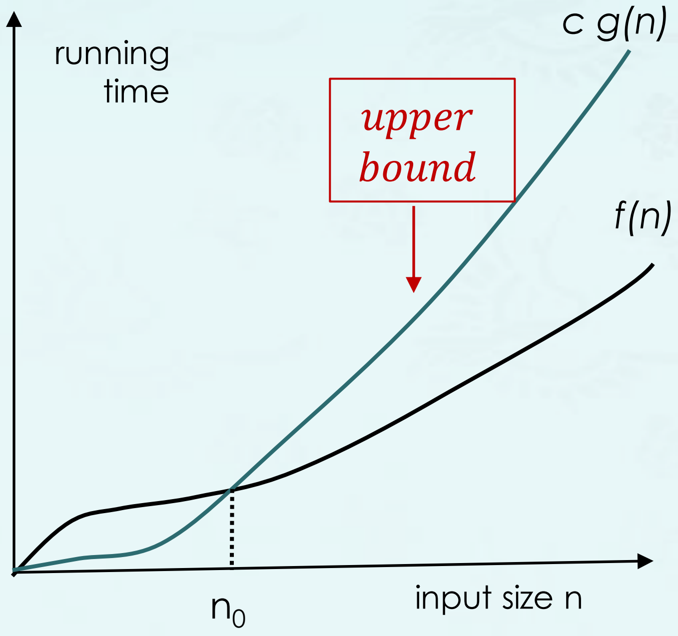


Asymptotic Analysis 점근적 분석과 표기법

- **The "Big-Oh" Notation:**
- Let f(n) and g(n) be functions mapping nonnegative integers to real numbers. We say that ***f(n) is O(g(n))*** iff there are positive constants ***c*** and $\boldsymbol{n_0}$ such that
  - $\boldsymbol{f(n)} \le \boldsymbol{c}\,\boldsymbol{g(n)}, for\ n \ge \boldsymbol{n_0}$.
  - Then it is pronounced as "$f(n)$ $\boldsymbol{is}$ $big\ Oh\ of\ g(n)\ or\ \boldsymbol{f(n)} = \boldsymbol{O(g(n))}$"

Asymptotic Analysis 점근적 분석과 표기법

- **The "Big-Oh" Notation:**
- Let f(n) and g(n) be functions mapping nonnegative integers to real numbers. We say that ***f(n) is O(g(n))*** iff there are positive constants ***c*** and $\boldsymbol{n_0}$ such that
  - $\boldsymbol{f(n) \leq c\, g(n)}, for\ n \geq \boldsymbol{n_0}.$
  - Then it is pronounced as "$f(n)\ \boldsymbol{is}\ big\ Oh\ of\ g(n)\ or\ \boldsymbol{f(n) = O(g(n))}$"



Example: Justify that the function $\boldsymbol{8n - 2}$ **is** $\boldsymbol{O(n)}$.
Given $f(n) = 8n - 2, g(n) = n,$
we need to find c and n_0 such that
$8n - 2 \leq c\, n$ for every integer $n \geq n_0$.

An easy choice among many is $\boldsymbol{c = 8}$ **and** $\boldsymbol{n_0 = 1}$.
Therefore, $f(n)\ is\ O(n)$.

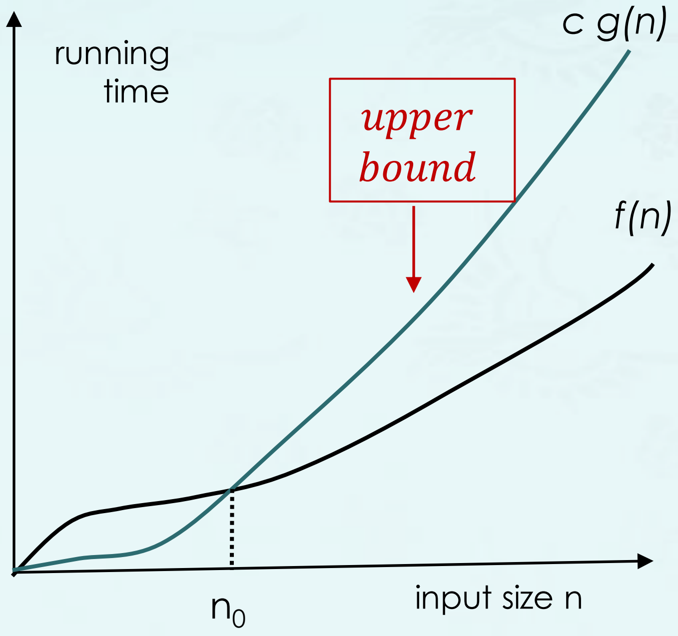
$g(n) = n$

Asymptotic Analysis 점근적 분석과 표기법

- **Example:** Find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ to justify that the function $7\boldsymbol{n} + 5$ **is** $\boldsymbol{O}(\boldsymbol{n})$.

7n + 5 is O(n), we have to find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ such that

$$7n + 5 \le c\,n \;\; for\; n \;\ge\; \boldsymbol{n_0}$$



Asymptotic Analysis 점근적 분석과 표기법

- **Example:** Find c and n_0 to justify that the function $7n + 5$ **is** $O(n)$.

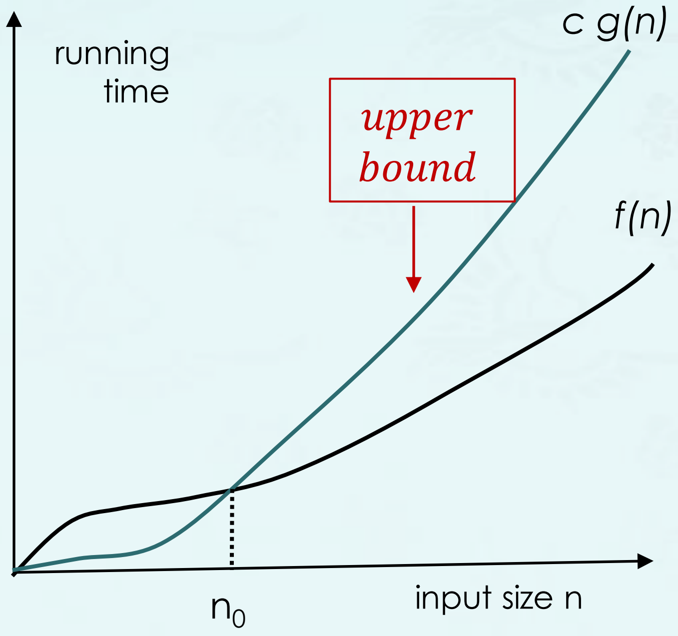
7n + 5 is O(n), we must find c and n_0 such that

$7n + 5 \le c\,n$ $for\ n \ge n_0$

$7n + 5 \le 7n + n$

$7n + 5 \le 8n$, $for\ n \ge n_0 = 5$

Therefore, $7n + 5 \le c\,n$ for $c = 8$ and $n_0 = 5$



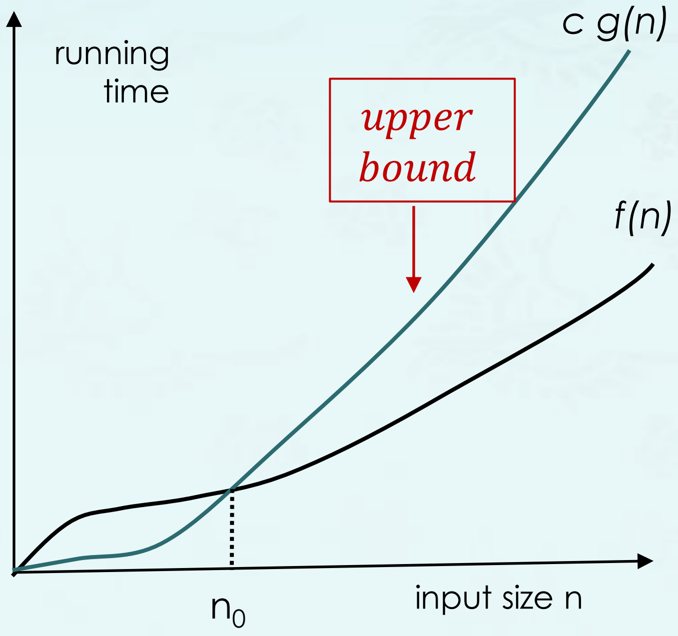
Asymptotic Analysis 점근적 분석과 표기법

- **Example:** Find c and n_0 to justify that the function $7n + 5$ **is** $O(n)$.

7n + 5 is O(n), we must find c and n_0 such that

$$7n + 5 \le c\,n \quad for\ n \ge n_0$$
$$7n + 5 \le 7n + n$$
$$7n + 5 \le 8n, \quad for\ n \ge n_0 = 5$$

Therefore, $7n + 5 \le c\,n$ for $c = 8$ and $n_0 = 5$



$$7n + 5 \le c\,n \quad for\ n \ge n_0$$
$$7n + 5 \le 12n \quad for\ n \ge n_0 = 1$$

Therefore, $7n + 5 \le c\,n$ for $c = 12$ and $n_0 = 1$

Asymptotic Analysis 점근적 분석과 표기법

- **Example:** Find c and n_0 to justify that the function $27n^2 + 16n$ **is** $O(n^2)$.

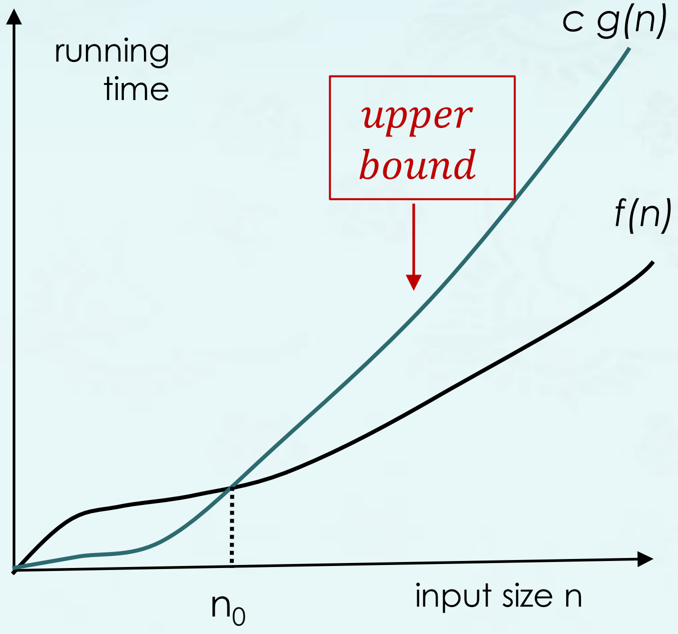
$27n^2 + 16n$ is $O(n^2)$, we must find c and n_0 such that

For $16n \le n^2$

$$27n^2 + 16n \le 27n^2 + n^2$$

$$27n^2 + 16n \le 28n^2 \; for\; n \ge n_0 = 16$$

Hence, c = 28 and n_0 = 16, Therefore, $f(n) = O(n^2)$.



$27n^2 + 16n$ is $O(n^2)$, we have to find c and n_0 such that

$$27n^2 + 16n \le 43n^2$$

$$27n^2 + 16n \le 43n^2 \; for\; n \ge n_0 = 1$$

Hence, c = 43 and n_0 = 1, Therefore, $f(n) = O(n^2)$.

Asymptotic Analysis 점근적 분석과 표기법

- **More Examples:**

1) $3n + 2 =$ [redacted]
2) $3n + 3 =$ [redacted]
3) $100n + 6 =$ [redacted]
4) $10n^2 + 4n + 2 =$ [redacted]
5) $6 * 2^n + n^2 =$ [redacted]
6) $3n + 3 =$ [redacted]
7) $10n^2 + 4n + 2 =$ [redacted]
8) $3n + 2 \neq O(1)$ $as\ 3n + 2\ is$ $\textbf{not}$ $\leq c\ for\ any\ c\ and\ all\ n, n \geq n_0$.
9) $10n^2 + 4n + 2 \neq O(n)$

Asymptotic Analysis 점근적 분석과 표기법

- **Preferred Big-Oh usage:**

- **Pick the tightest bound.** If $f(N) = 5N$, then:
  - $f(N) = O(N^5)$
  - $f(N) = O(N^3)$
  - $f(N) = O(N \log N)$
  - $\mathbf{f(N) = O(N)}$ ← preferred or right!

Asymptotic Analysis 점근적 분석과 표기법

- **Preferred Big-Oh usage:**

- **Pick the tightest bound.** If $f(N) = 5N$, then:
  - $f(N) = O(N^5)$
  - $f(N) = O(N^3)$
  - $f(N) = O(N \log N)$
  - $\mathbf{f(N) = O(N)}$ ← preferred or right!
- **Ignore constant factors and low order terms:**
  - $f(N) = \mathbf{O(N)}$, *not* $f(N) = O(5N)$
  - $f(N) = \mathbf{O(N^3)}$, *not* $f(N) = O(N^3 + N^2 + 15)$

- Wrong: $f(N) \le O(g(N))$
- Wrong: $f(N) \ge O(g(N))$
- Right: $\mathbf{f(N) = O(g(N))}$

Asymptotic Analysis 점근적 분석과 표기법

- Suppose two algorithms, A and B, solving the same problem have the running time of $O(n)$ and $O(n^2)$, respectively.
- Then algorithm A is **asymptotically better** than algorithm B.

※ $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$

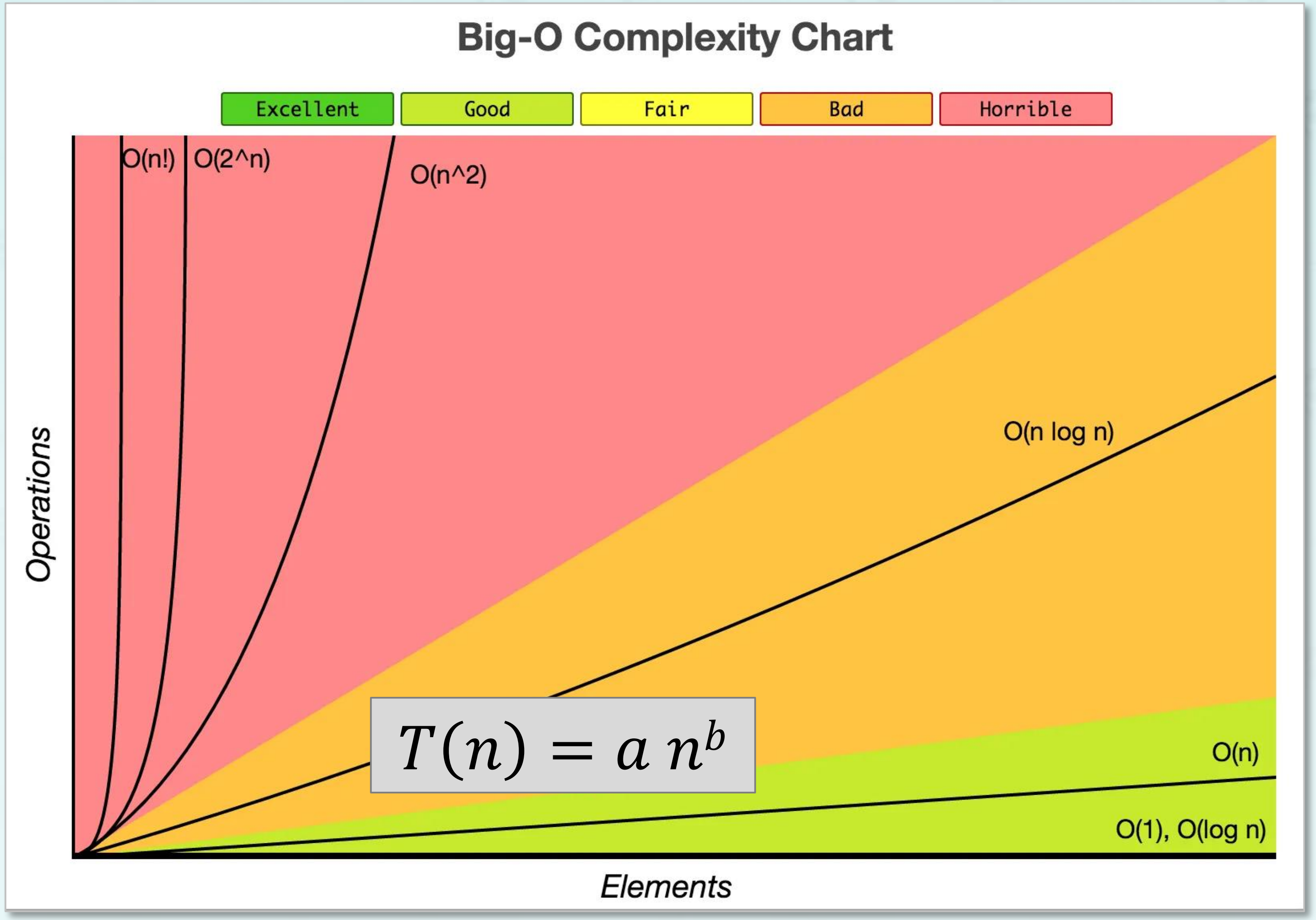
logarithmic linear linearithmic quadratic cubic exponential

정수함수 대수 선형함수 선형대수 loglinear 2/3승함수 지수함수

$$\boxed{T(n) = a\, n^b}$$

Asymptotic Analysis 점근적 분석과 표기법

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$$



$$T(n) = a\,n^b$$

Asymptotic Analysis 점근적 분석과 표기법

[**Omega**] ***f(n) = Ω (g(n))*** iff there exist positive constants c and n_0 such that

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

Asymptotic Analysis 점근적 분석과 표기법

[**Omega**] ***f(n) = Ω (g(n))*** iff there exist positive constants c and n_0 such that

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

- **Example:** Let's suppose we have

$$f(n) = 5n^2 + 2n + 1$$

$$g(n) = n^2$$

For all $n \geq 0$, this $(2n + 1)$ will be ≥ to 1, **if** we have $c = 5$ and $n_0 = 0.$

Then, $5\, n^2 \leq f(n)$, for all $n \geq 0$

Therefore, we can say that the time complexity of $f(n)$ is $\Omega\, (\boldsymbol{n^2})$;

Asymptotic Analysis 점근적 분석과 표기법

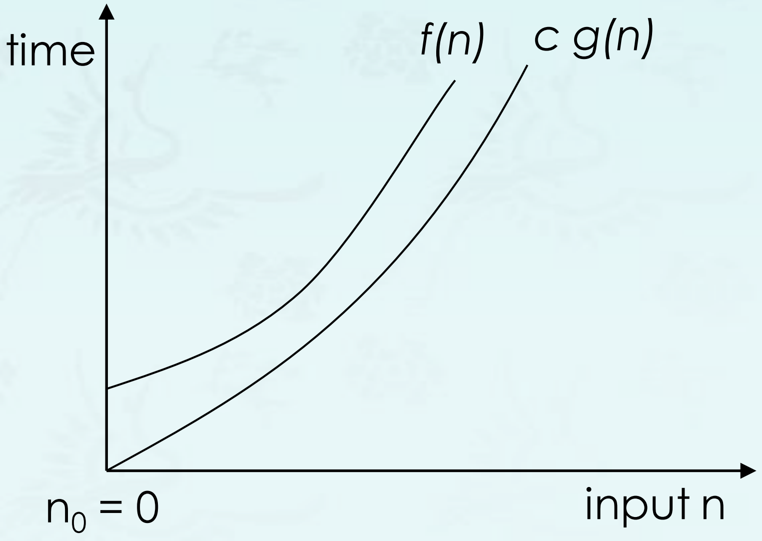
[Omega] ***f(n) = Ω (g(n))*** iff there exist positive constants c and n_0 such that

$$f(n) \geq c\, g(n), for\; n \geq n_0.$$

- **Example:** Let's suppose we have

$$f(n) = 5n^2 + 2n + 1$$

$$g(n) = n^2$$



- **Omega** notation gives us the **lower bound** of the growth rate of a function.

Asymptotic Analysis 점근적 분석과 표기법

[Omega] ***f(n) = Ω (g(n))*** iff there exist positive constants c and n_0 such that

$$f(n) \geq c\, g(n), for\ n \geq n_0$$

- **More Example:**

1) $3n + 2 = \Omega(n)\ since\ \mathbf{3n + 2} \geq \mathbf{3n}\ for\ n \geq 1$

2) $3n + 3 = \Omega(n)\ since\ 3n + 3 \geq 3n\ for\ n \geq 1$

3) $100n + 6 = \Omega(n)\ since\ 100n + 6 \geq 100n\ for\ n \geq 1$

4) $100n^2 + 4n + 2 = \Omega(n^2)\ since\ 100n^2 + 4n + 2 \geq n^2\ for\ n \geq 1$

5) $6 * 2^n + n^2 = \Omega(2^n)\ since\ 6 * 2^n + n^2 \geq 2^n\ for\ n \geq 1$

- **Omega** notation gives us the **lower bound** of the growth rate of a function.

Asymptotic Analysis 점근적 분석과 표기법

[Theta] ***f(n)* = Θ *(g(n))*** iff there exist positive constants c and n_0 such that

- $c_1 g(n) \le f(n) \le c_2 g(n), for\ n \ge n_0.$

- **Example:** Let's suppose we have

$$f(n) = 5n^2 + 2n + 1$$
$$g(n) = n^2$$

- Then, we can choose $c_1 = 5, c_2 = 8$, and $n_0 = 1$; and our inequality will hold. Therefore, we can say that the time complexity of

$$f(n) = 5n^2 + 2n + 1 = \Theta\,(n^2)$$

Asymptotic Analysis 점근적 분석과 표기법

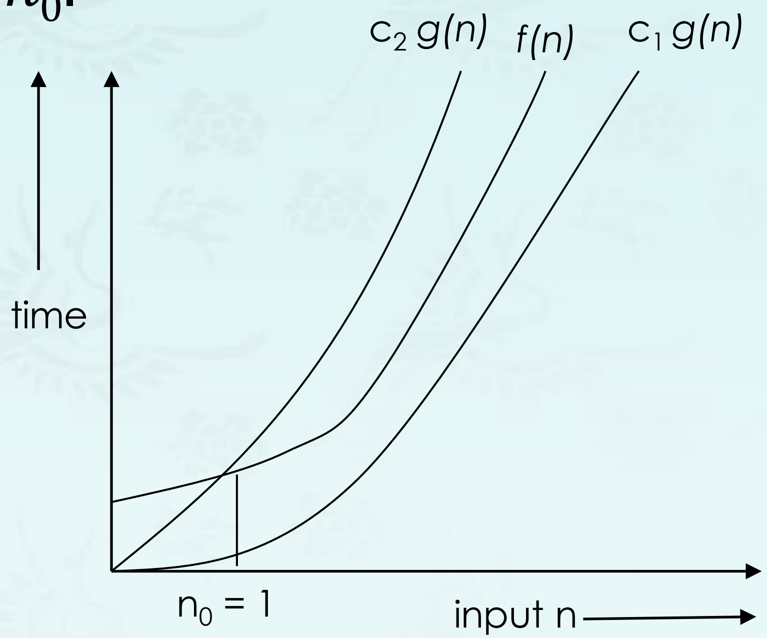
[Theta] *f(n)* = Θ (*g(n)*) iff there exist positive constants c and n_0 such that

- $c_1 g(n) \le f(n) \le c_2 g(n), for\ n \ge n_0.$

- **Example:** Let's suppose we have

$$f(n) = 5n^2 + 2n + 1$$

$$g(n) = n^2$$



- **Θ notation** best describes or give the best idea about the growth rate of the function because it gives us a **tight bound** unlike **O and Ω** which give us **upper bound** and **lower bound**, respectively.

Asymptotic Analysis 점근적 분석과 표기법

[Theta] *f(n)* = Θ *(g(n))* iff there exist positive constants c and n_0 such that

- $c_1 g(n) \le f(n) \le c_2 g(n), for\ n \ge n_0.$

- **More Examples:**

1) $3n + 2 = \Theta(n)$

since $\mathbf{3n} \le \mathbf{3n+2} \le \mathbf{4n}\ for\ all\ n \ge 2, c_1 = 3, c_2 = 4, and\ n_0 = 2$

2) $3n + 3 = \Theta(n)$

3) $10n^2 + 4n + 2 = \Theta(n^2)$

4) $6 * 2^n + n^2 = \Theta(2^n)$

5) $10 * \log n + 4 = \Theta(\log n)$

Asymptotic Analysis - Quiz

- Example: Running time estimates - empirical analysis
  - Personal computer executes 10^9 compares/second
  - Supercomputer executes 10^{13} compares/second

	Selection sort **(N^2)**			Merge sort **(N $\log_2$ N)**		
N	Million	10 million	Billion	Million	10 million	Billion
PC	16.7 min			instant	0.2 sec	
Super Com	0.1 sec			Instant	Instant	Instant

$log_{10}2 \cong 0.3$
$86{,}400\text{sec}/day$
instant < 0.1 sec

Use a reasonable or understandable time units.
Do not say, for example, "3660 days" nor "1220 seconds", but 10.0 years or 20.3 min, respectively.

※ **Bottom line**: Good algorithms are better than supercomputers.

Data Structures
Chapter 1

1. Recursion
2. Performance Analysis
3. **Asymptotic Analysis**
   - Revisit – Step Count
   - Asymptotic Analysis
   - Asymptotic Notations