2. Prologue

2.1 Introduction

2.2 Computational complexity

2.3 Time complexity of common functions

2.4 Recurrence relation

2.5 Fibonacci

2.2 Computational complexity

$$Performance = Efficiency\ (효율) = \frac{Solution\ (fix)}{Cost}$$

숙제를 해야 하는데 며칠 걸리나?

$$Performance = Effective\ (효과) = \frac{Solution}{Cost\ (fix)}$$

하루에 몇 문제나 풀 수 있을까?

- Solution
  - No solution, no performance

- Cost
  - Resource = temporal + spatial
    - temporal resource: CPU
    - spatial resource: memory

2.2 Computational complexity

- Three aspects of performance
  - Best case
    - Game score
  - Average case
    - GPA
  - Worst case
    - ATM

Key point 1:
Performance of **worst case** is important

2.2 Computational complexity

- Space complexity

  “the amount of memory that it needs to run to completion”

- Time complexity

  “the amount of computer time that it needs to run to completion”

- Power complexity

  “the amount of electricity that it needs to run to completion”

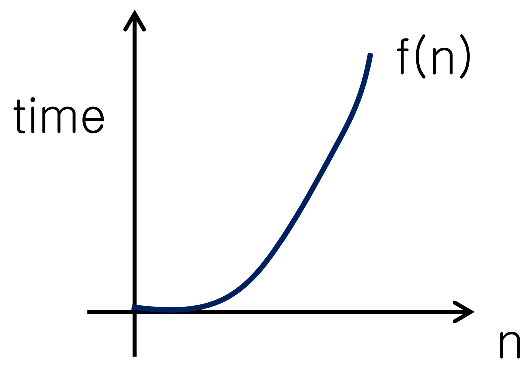
Key point 2:
Performance of **time** is important

2.2 Computational complexity

(1) Asymptotic complexity (漸近法)

- Performance depends on “input”
  - If input is n, then the performance is f(n)
  - Performance of an algorithm: (n, f(n))

$$n_0 < n_1 \rightarrow f(n_0) \leq f(n_1)$$



Key point 3:
Performance is a function of **input size**

2.2 Computational complexity

(1) Asymptotic complexity

- The size of input ➔ n
- The complexity → function of n, f(n)
- To estimate the complexity function for **reasonably large length of input**

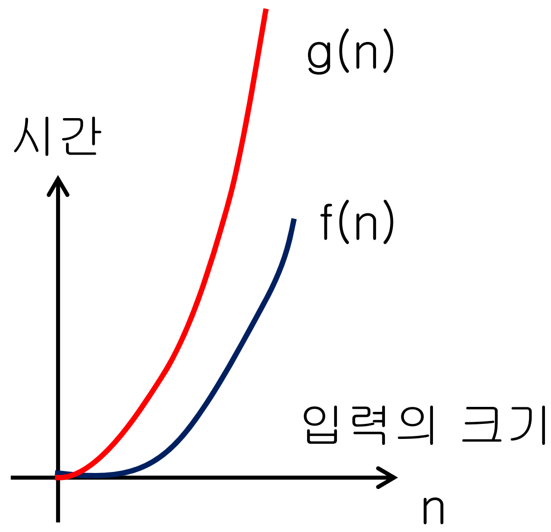
- Examples of f(n): O(n), Ω(n), Θ(n)
  - Asymptotic upper bound: O(n) → **Worst case**
  - Asymptotic lower bound: Ω(n) → Best case
  - Asymptotic tight bound: Θ(n) → Exact case

Key point 4:
Input of **very large size** is important

2.2 Computational complexity

(1) Asymptotic complexity

- The worst of f(n) is g(n)?
  - In the worst case, f(n) is better than g(n)
  - g(n)
    - A standard for measurements
    - E.g. 1, n, log n, n^2, n log n, n^n
  - f(n) is better than g(n) → f(n) ≤ g(n)
  - The upper bound of f(n) is g(n)



Key point 5:
Some **key functions** are used for standards

(2) Big-O Notation

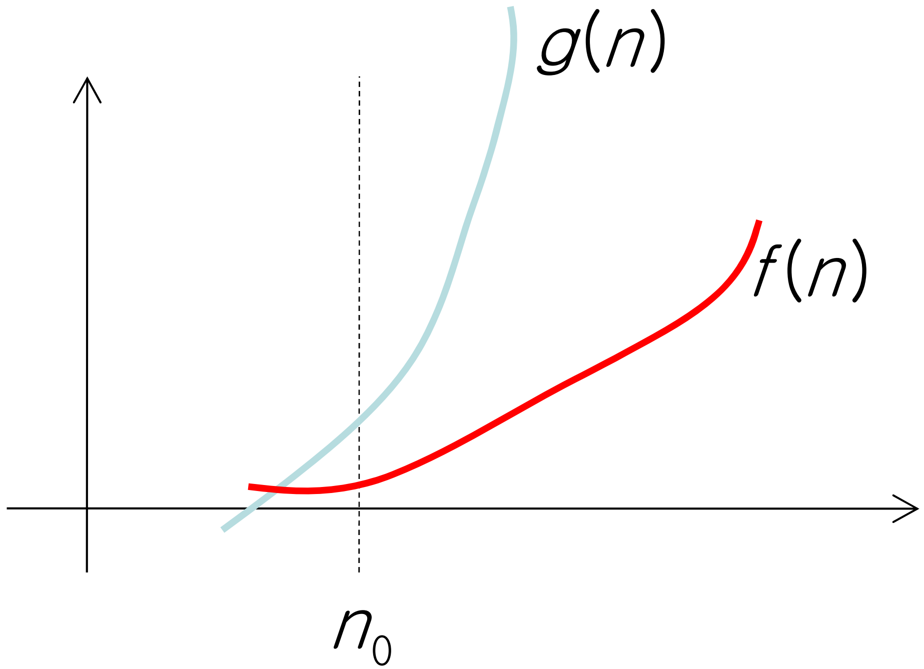
$$f(n) \text{ is } O(g(n)) \text{ as } n \to \infty, \text{ if and only if}$$
$$\exists n_0, \exists M > 0 \text{ such that } |f(n)| \le M|g(n)| \text{ for } n > n_0$$

– To describe an **asymptotic upper bound** for the magnitude of a function

(2) Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if
$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n > n_0$

– f(n) = O(g(n))



2.2 Computational complexity

(2) Big-O Notation

$$f(n) \text{ is } O(g(n)) \text{ as } n \to \infty, \text{ if and only if}$$
$$\exists n_0, \exists M > 0 \text{ such that } |f(n)| \le M|g(n)| \text{ for } n > n_0$$

– f(n) = O(g(n))

- For $n > n_0$, f(n) has no chance to be greater than g(n).
- Suppose f(n) is the time required to execute a function with n inputs.
- Even at worst case, the function finishes no later than g(n).
- The upper bound of the time required to finish the function is g(n).
- The upper bound of f(n) is g(n)

2.2 Computational complexity

(2) Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n > n_0$

- $f(n) = O(\ g(n)\ )$
  - If f(n) = n, which function of the followings can be g(n)?
    - n
    - n^2
    - n^3
    - n^5
    - e^n

오늘 나온 숙제를 나는 2일이면 다 할 수 있다. 그런데, 교수님은 숙제 기간을 며칠 줄까요?라고 묻는다. 나는 며칠이 필요하다고 해야 할까?

1) 1일
2) 2일
3) 3일
4) 4일
5) 5일

2.2 Computational complexity

(2) Big-O Notation

$$f(n) \text{ is } O(g(n)) \text{ as } n \to \infty, \text{ if and only if}$$
$$\exists n_0, \exists M > 0 \text{ such that } |f(n)| \leq M|g(n)| \text{ for } n > n_0$$

- $f(n) = O(g(n))$
  - f(n) is faster than g(n)
  - g(n) is slower than f(n) → $g(n) = \Omega(f(n))$

- $g(n) = \Omega(f(n))$, if $g(n) \geq M f(n)$

A가 3일만에 숙제를 하고 B가 4일만에 숙제를 한다면,
A는 B보다 빠르다 또는 → A = O (B)
B는 A보다 느리다. → B = Ω (A)

2.2 Computational complexity

(2) Big-O Notation

$$f(n) \text{ is } O(g(n)) \text{ as } n \to \infty \text{, if and only if}$$
$$\exists n_0, \exists M > 0 \text{ such that } |f(n)| \leq M|g(n)| \text{ for } n > n_0$$

- f(n) = O (g(n)) and f(n) = Ω (g(n))
  - f(n) ≤ M g(n) and f(n) ≥ M g(n)
  - f(n) = Θ (g(n))

f(n)과 g(n)은 같은 비율로 증가함

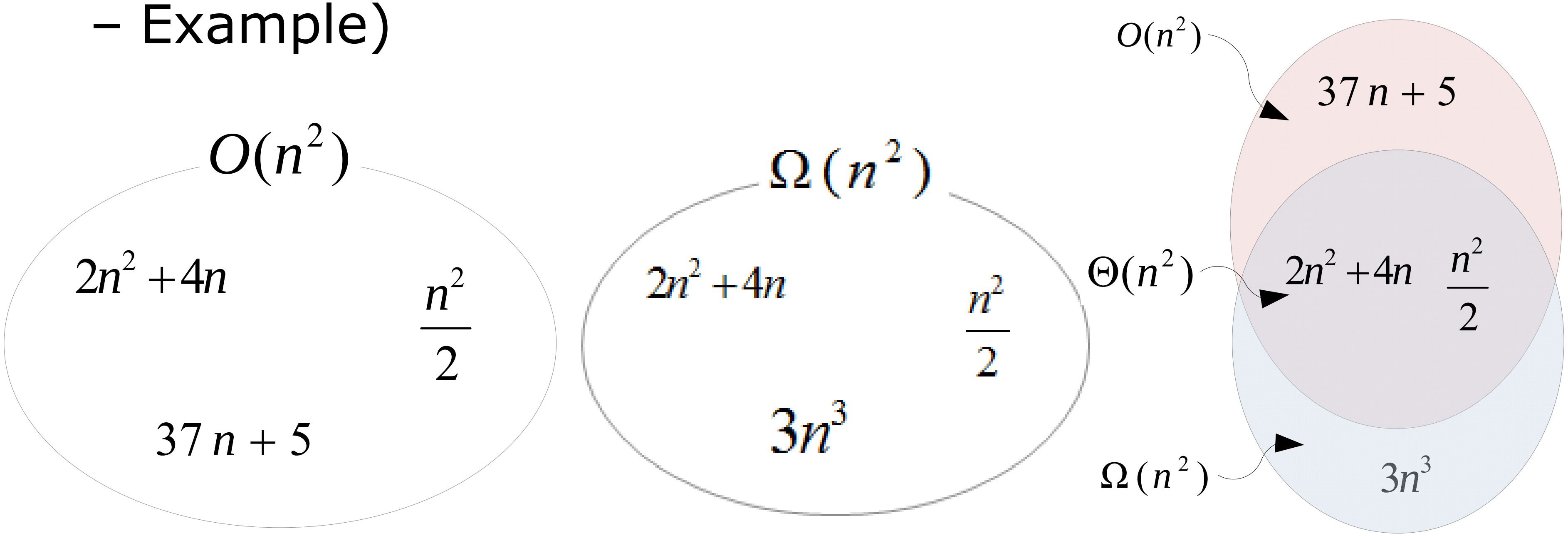
2.2 Computational complexity

(2) Big-O Notation

$f(n)$ is $O(g(n))$ as $n \to \infty$, if and only if

$\exists n_0, \exists M > 0$ such that $|f(n)| \le M|g(n)|$ for $n > n_0$

– Example)



2.2 Computational complexity

- Summing up
  - Performance → Efficiency → Solution / Cost

    → f(n): n (size of input), f (time)

    f(n)의 그래프를 그릴 수 있나요?

  - Worst case performance → upper bound

    $f(n) = O(g(n))$, $f(n) <= M\ g(n)$, for $n > n_0$

    f(n)은 항상 g(n)보다 빠름.
    f(n)의 worst case는 g(n).
    f(n)의 upper bound는 g(n).
    ……

2.2 Computational complexity

- Summing up
  - Increase ratio is important

    $f(n) = 1000n$, then $f(n) = O(n)$

    $g(n) = 0.0001n$, then $g(n) = O(n)$

    f(n)과 g(n)은 똑같이 n에 비례해서 증가함.

  - Higher term overrides lower term

    ex) n^2 overrides n

    $f(n) = n^2 + n$, then $f(n) = O(n^2)$

    증명할 수 있나요?

Quiz2

– Determine whether f = O(g) or g = O(f) or both

- $f(n) = n^{1/2}$, $g(n) = n^{2/3}$
- $f(n) = n^2/\log n$, $g(n) = n(\log n)^2$
- $f(n) = n^{0.1}$, $g(n) = (\log n)^{10}$
- $f(n) = n!$, $g(n) = 2^n$
- $f(n) = \sum_{i=1}^{n} i^k$, $g(n) = n^{k+1}$