알고리즘 복잡도의 점근적 표기법

HaRim Jung, Ph.D.
Visiting Professor / Senior Researcher
SKKU Institute for Convergence / Convergence Research Institute
Sungkyunkwan University, Korea

알고리즘 복잡도의 점근적 표기법 (1/11)

❑ 알고리즘의 (시간) 복잡도

- 알고리즘의 효율성을 분석하기 위해 알고리즘의 실행 시간(running time)을 **입력 크기 n**에 따른 **단위 연산 수**로 정의한 것으로 n에 대한 함수로 표현할 수 있음
- 동일한 문제에 대한 해결책을 찾는 두 개의 알고리즘 A1과 A2가 존재한다고 가정
  - 입력 크기 n에 대한 A1의 (시간) 복잡도는 n이고 A2의 복잡도는 n^2이라고 하면 A1가 더 효율적
  - 입력 크기 n에 대한 A1의 복잡도는 $0.1n^2$이고 A2의 복잡도는 1,000n이라고 하면 어떤 알고리즘이 더 효율적일까?
    - n이 10,000보다 작다면 A1이 더 효율적이고, n이 10,000보다 크다면 A2가 더 효율적임

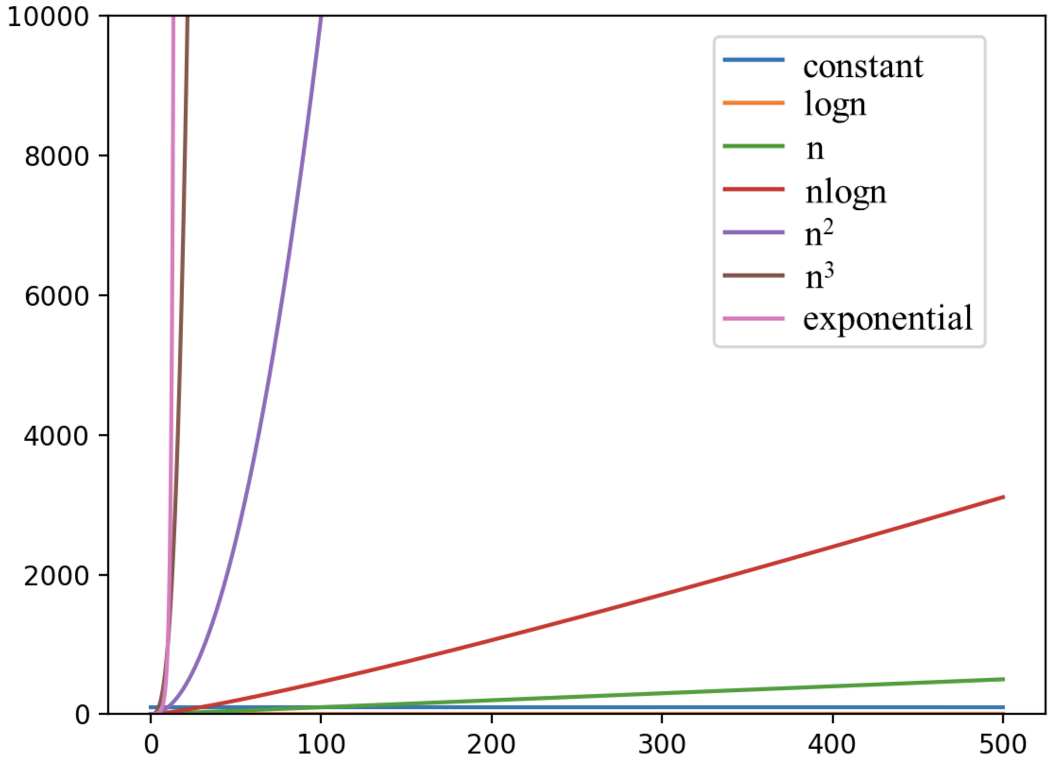
❑ 알고리즘의 점근적 복잡도(asymptotic complexity)와 복잡도 카테고리

- **입력 크기 n의 값이 무한히(혹은 충분히) 큰 경우일 때 알고리즘의 복잡도**
  - n의 값이 충분히 큰 경우 A1과 A2 중 더 효율적인 알고리즘은?
  - n에 대한 함수로 표현되는 (시간) 복잡도는 최고차 항의 **차수(order)**가 궁극적으로 지배
    - 예1: n = 100,000인 경우 A1의 시간 복잡도의 값은 10,000,000,000이고 A2의 시간 복잡도 값은 100,000,000임
      - » 이와 같이 n의 값이 무한히 큰 경우 상수(constant) 0.1과 1,000은 무시할 수 있음
    - 예2: n = 1,000인 경우 특정 알고리즘 A3의 시간 복잡도 $f(n) = n^2 + n + 1$의 값은 1,001,001이고 이 중 첫 번째 항인 n^2 의 값이 전체의 약 99%인 1,000,000이고 두 번째 항 n의 값이 1,000으로 전체의 약 1%를 차지
      - » 이와 같이 n의 값이 무한히 큰 경우 최고차항 이외의 항은 무시할 수 있음

알고리즘 복잡도의 점근적 표기법 (2/11)

❑ 알고리즘의 점근적 복잡도(asymptotic complexity)와 복잡도 카테고리 contd.

- 최고차 항을 기준으로 알고리즘 복잡도의 카테고리를 다음과 같이 나눌 수 있음



1	**상수 시간 (constant time)**
logn	**로그 시간 (logarithmic time)**
n	**선형 시간 (linear time)**
nlogn	**로그 선형 시간 (log-linear time)**
n^2	**제곱 시간 (quadratic time)**
n^3	**세제곱 시간 (cubic time)**
2^n	**지수 시간 (exponential time)**

– 알고리즘의 효율성은 **점근적 복잡도에 기반하여 분석**하며 위의 복잡도 카테고리에서 보이는 단순한 **복잡도 함수**(e.g., 1, logn, n, nlogn, …)를 사용 → **입력 크기 n의 값이 증가함에 따라 알고리즘의 단위 연산 수가 얼마나 빠르게 증가하는가에 초점**을 맞춤

알고리즘 복잡도의 점근적 표기법 (3/11)

❑ 점근적 표기법(asymptotic notation)

- 알고리즘의 점근적 복잡도를 표기하는 방법(i.e., **O** 표기법, **Ω** 표기법, **Θ** 표기법, **o** 표기법, ω 표기법)

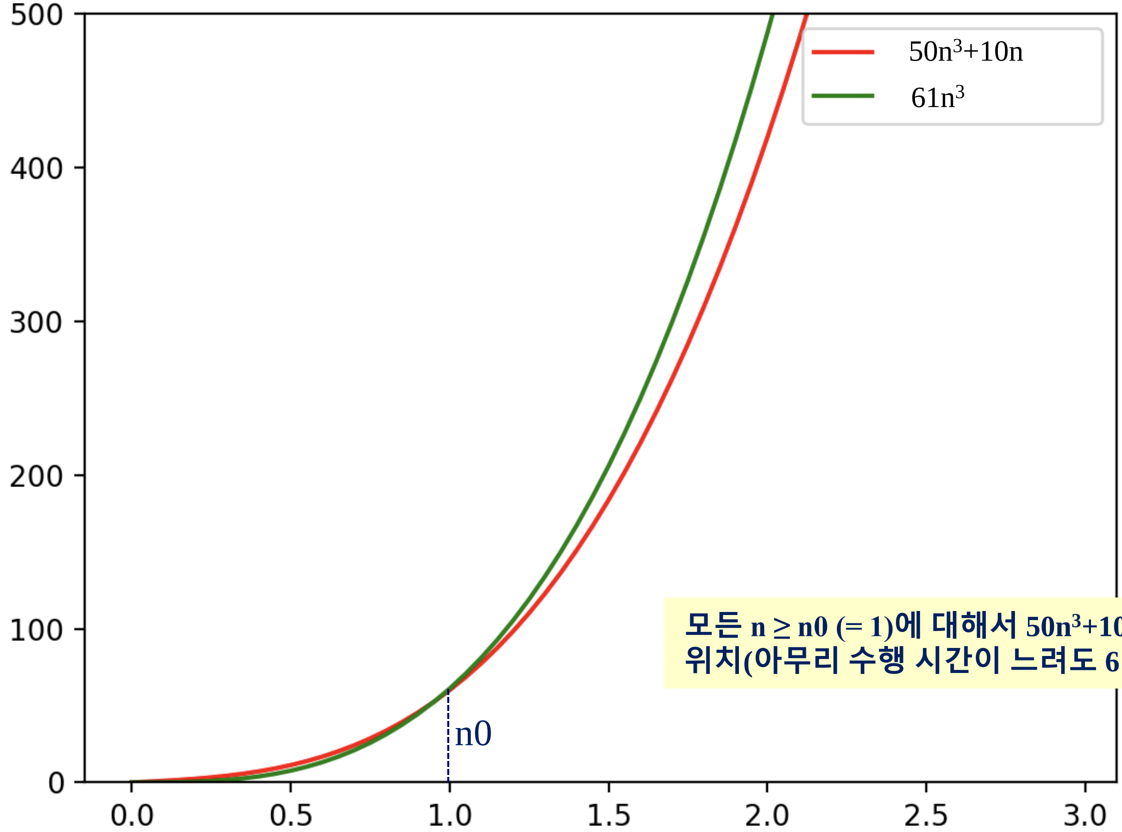
❑ O(Big-Oh) 표기법

- 알고리즘 복잡도의 점근적 상한(asymptotic upper bound)을 표시
- 특정 알고리즘의 복잡도 g(n)과 (복잡도 카테고리 내의) 복잡도 함수 f(n)에 대해서 다음을 만족하면 $g(n) \in O(f(n))$ (“g(n)은 f(n)의 Big-Oh에 속한다.”)
  - $n \geq n0$인 모든 정수 n에 대해서 항상 $g(n) \leq c \times f(n)$이 성립하는 실수 $c > 0$와 음이 아닌 정수 n0이 존재

    복잡도 함수 f(n)
  - $g(n) \in O(n^3)$의 의미는 특정 알고리즘의 입력 크기 n이 어떤 임의의 n0보다 큰 값부터 무한대까지의 값을 가질 때 이 알고리즘의 복잡도 g(n)은 어떤 3차 함수 cn^3보다는 **작은** 값을 가지게 된다는 것을 의미함
    - 즉, 입력의 크기 n에 대해서 이 알고리즘의 수행시간은 **아무리 느려도** $c \cdot f(n)$과 같거나 빠르다(**궁극적으로, $c \cdot f(n)$이 이 알고리즘의 복잡도의 상한이다**)라는 것을 의미
- 예제 1: $50n^3+10n \in O(n^3)$?
  - $50n^3+10n \leq n^3$의 양변을 n^3으로 나누면 $50+\frac{10}{n^2} \leq c$이 되는데, $c = 61$과 $n0 = 1$을 선택하면 ‘Big-Oh’의 정의에 의해서 $50n^3+10n \in O(n^3)$이라고 결론지을 수 있음

알고리즘 복잡도의 점근적 표기법 (4/11)

- O(Big-Oh) **표기법** contd.
  - 예제 1: $50n^3+10n \in O(n^3)$? contd.
    - g(n): $50n^3+10n$과 $61 \cdot f(n)$: $61n^3$에 대한 그래프



모든 $n \geq n0$ (= 1)에 대해서 $50n^3+10n$은 그래프 상에서 $61n^3$ 아래에 위치(아무리 수행 시간이 느려도 $61n^3$과 같거나 빠름)

알고리즘 복잡도의 점근적 표기법 (5/11)

- O(Big-Oh) **표기법** contd.
  - 예제 2: $n^2+10n \in O(n^2)$?
    1. $n \geq 10$인 모든 정수 n에 대해서 $n^2+10n \leq 2n^2$ 이 성립하므로, $c = 2$와 $n0 = 10$을 선택하면 'Big-Oh'의 정의에 의해서 $n^2+10n \in O(n^2)$이라고 결론지을 수 있음
    2. $n \geq 1$인 모든 정수 n에 대해서 $n^2+10n \leq n^2+10n^2 = 11n^2$이 성립하므로, $c = 11$과 $n0 = 1$을 선택하면 'Big-Oh'의 정의에 의해서 $n^2+10n \in O(n^2)$이라고 결론지을 수 있음

    → $g(n) \in O(f(n))$를 보이는데 있어서 c와 n0에 대해 한 가지 값만 있는 것이 아니므로 **적당히 큰 c와 n0을 선택하여** 보이면 됨
  - 예제 3: $\frac{n(n-1)}{2} \in O(n^2)$?
    - $n \geq 0$인 모든 정수 n에 대해서 $\frac{n(n-1)}{2} \leq \frac{n^2}{2}$ 이 성립하므로 $c = \frac{1}{2}$ 과 $n0 = 0$을 선택하면, $\frac{n(n-1)}{2} \in O(n^2)$이라고 결론지을 수 있음
  - 예제 4: $n^3 \in O(n^2)$?
    - 양변을 n^2으로 나누면, $n \leq c$가 되는데 c에 아무리 큰 수를 대입해도 그 보다 더 큰 n이 존재하므로 $n^3 \notin O(n^2)$이라고 결론지을 수 있음 (즉, $n \geq n0$인 모든 n에 대해서 $n^3 \leq cn^2$ 이 성립하는 c와 n0값은 존재하지 않음)

알고리즘 복잡도의 점근적 표기법 (6/11)

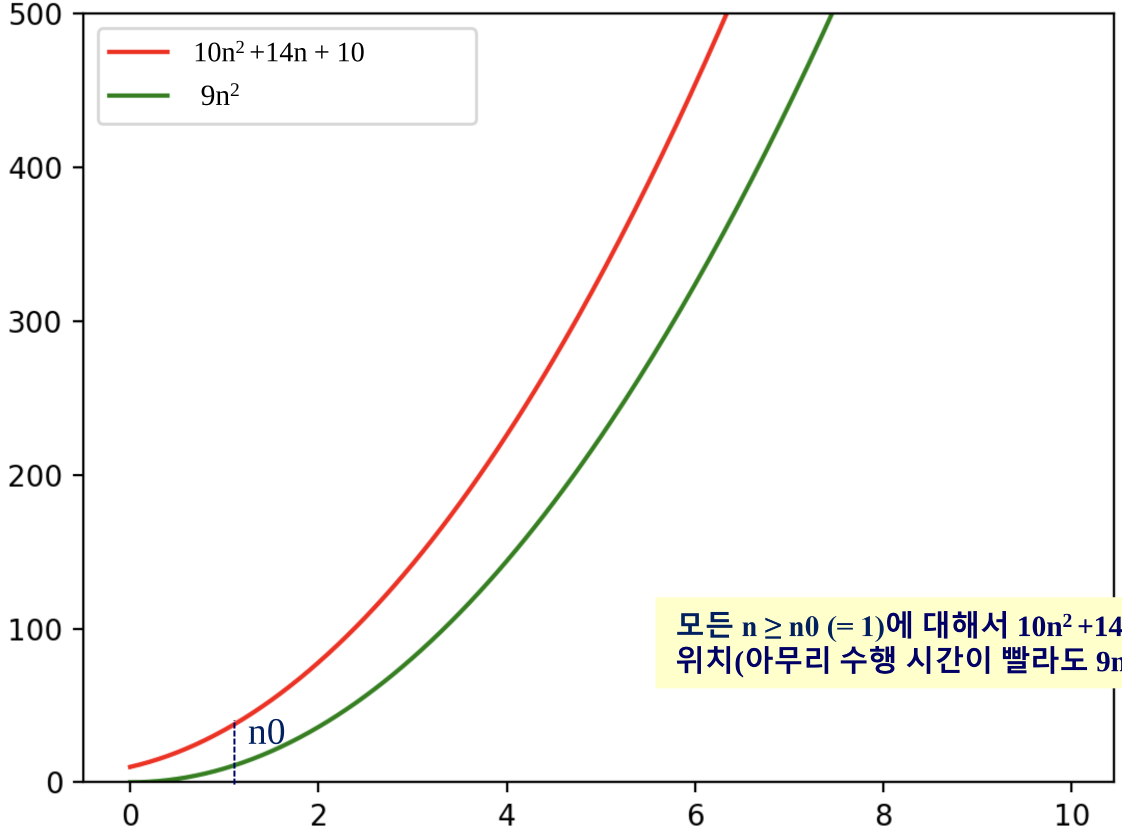
❑ Ω(Omega) 표기법

- 알고리즘 복잡도의 점근적 하한(asymptotic lower bound)을 표시
- 특정 알고리즘의 복잡도 g(n)과 (복잡도 카테고리 내의) 복잡도 함수 f(n)에 대해서 다음을 만족하면 $g(n) \in \Omega(f(n))$ (“g(n)은 f(n)의 Omega에 속한다.”)
  - $n \geq n0$인 모든 정수 n에 대해서 항상 $g(n) \geq c \times f(n)$이 성립하는 실수 $c > 0$와 음이 아닌 정수 n0이 존재
  - $g(n) \in \Omega(n^2)$의 의미는 특정 알고리즘의 입력 크기 n이 어떤 임의의 n0보다 큰 값부터 무한대까지의 값을 가질 때 이 알고리즘의 복잡도 g(n)은 어떤 2차 함수 cn^2보다는 **큰** 값을 가지게 된다는 것을 의미함
    - 즉, 입력의 크기 n에 대해서 이 알고리즘의 수행시간은 **아무리 빨라도** $c \cdot f(n)$과 같거나 느리다(**궁극적으로, $c \cdot f(n)$이 이 알고리즘의 복잡도 하한이다**)라는 것을 의미
- 예제 1: $10n^2 + 14n + 10 \in \Omega(n^2)$? 9
  - $n \geq 1$인 모든 정수 n에 대해서 $10n^2 + 14n + 10 \leq n^2$이 성립하므로, $c = 9$와 $n0 = 1$을 선택하면 ‘Omega’의 정의에 의해서 $n^2+10n \in O(n^2)$이라고 결론지을 수 있음

→ $g(n) \in \Omega(f(n))$를 보이는데 있어서 c와 n0에 대해 한 가지 값만 있는 것이 아니므로 **적당히 작은 c와 n0을 선택하여** 보이면 됨

알고리즘 복잡도의 점근적 표기법 (7/11)

- Ω(Omega) **표기법** contd.
  - 예제 1: $10n^2 + 14n + 10 \in \Omega(n^2)$? contd.
    - g(n): $10n^2 + 14n + 10$과 $9 \cdot f(n)$: $9n^2$에 대한 그래프



모든 $n \geq n0$ (= 1)에 대해서 $10n^2 + 14n + 10$은 그래프 상에서 $9n^2$ 위에 위치(아무리 수행 시간이 빨라도 $9n^2$과 같거나 느림)

알고리즘 복잡도의 점근적 표기법 (8/11)

- Ω(Omega) **표기법** contd.
  - 예제 2: $n^2 + 10n \in \Omega(n^2)$?
    - $n \geq 0$인 모든 정수 n에 대해서 $n^2 + 10n \geq n^2$ 이 성립하므로, $c = 1$와 $n0 = 0$을 선택하면, $n^2 + 10n \in \Omega(n^2)$이라 결론지을 수 있음
  - 예제 3: $\frac{n(n-1)}{2} \in \Omega(n^2)$?
    - $n \geq 2$인 모든 정수 n에 대해서 $n - 1 \geq \frac{n}{2}$ 이 성립하므로 2보다 같거나 큰 모든 n에 대해서 $\frac{n(n-1)}{2} \geq \frac{n}{2} \times \frac{n}{2} = \frac{n}{4}$ 이 성립. 따라서 $c = \frac{1}{4}$ 과 $n0 = 2$를 선택하면 $\frac{n(n-1)}{2} \in \Omega(n^2)$이라고 결론지을 수 있음
  - 예제 4: $n^3 \in \Omega(n^2)$?
    - $n \geq 1$인 모든 정수 n에 대해서 $n^3 \geq 1 \times n^2$이 성립하므로 $c = 1$와 $n0 = 1$을 선택하면 $n^3 \in \Omega(n^2)$이라고 결론지을 수 있음
  - 예제 5: $n \in \Omega(n^2)$?
    - 모순에 의한 증명(proof by contradiction)
      - $n \in \Omega(n^2)$이라고 가정하면 $n \geq n0$인 모든 정수 n에 대해서 $n \geq c \cdot n^2$이 성립하는 실수 $c > 0$, 그리고 음이 아닌 정수 n0이 존재함
      - 양변을 $c \cdot n$으로 나누면 $\frac{1}{c} \geq n$이 되는데 $\frac{1}{c}$ 보다 큰 n이 항시 존재하기 때문에 이 부등식은 성립할 수 없으므로 위의 가정은 모순임 $\rightarrow n \notin \Omega(n^2)$

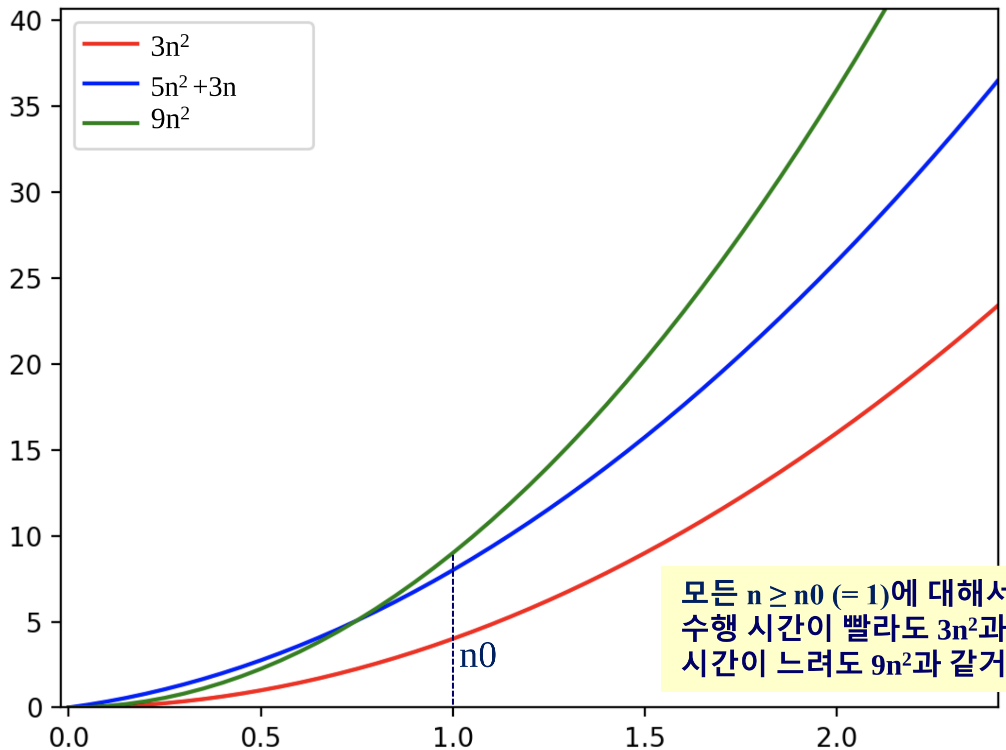
알고리즘 복잡도의 점근적 표기법 (9/11)

❑ Θ(Theta) **표기법**

- 복잡도 함수 f(n)에 대해서 **$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$**의 관계가 성립함
- 특정 알고리즘의 복잡도 g(n)과 (복잡도 카테고리 내의) 복잡도 함수 f(n)에 대해서 다음을 만족하면 $g(n) \in \Theta(f(n))$ (“g(n)은 f(n)과 동일한 차수(order)이다.”)
  - $n \geq n0$인 모든 정수 n에 대해서 **$c \times f(n) \leq g(n) \leq d \times f(n)$**이 성립하는 실수 $c > 0$, $d > 0$, 그리고 음이 아닌 정수 n0이 존재
  - 예를 들어, $g(n) = \frac{n(n-1)}{2}$은 $g(n) \in O(n^2)$이면서 $\Omega(n^2)$이므로 $g(n) \in \Theta(f(n))$
  - 일반적으로 Θ(f(n))는 복잡도 카테고리를 나눌 때 사용
- 예제 1: $5n^2 + 3n \in \Theta(n^2)$?
  - $n \geq n0$인 모든 정수 n에 대해서 $c \times n^2 \leq 5n^2 + 3n \leq d \times n^2$ 이 성립하는데 세 변을 n^2으로 나누면 $c \leq 5 + \frac{3}{n} \leq d$가 되는데, $n0 = 1$을 선택하면 식의 중간 변은 (1) 5보다 작아질 수 없고 (2) 8보다 커질 수 없음. 따라서 $c = 5$, $d = 9$를 선택하면 $5n^2 + 3n \in \Theta(n^2)$ 이라고 결론지을 수 있음
  - NOTE: g(n)은 복잡도 함수 n^2보다 작아질 수 없고 n^2보다 커질 수 없으므로 **g(n)의 차수는 n^2**

알고리즘 복잡도의 점근적 표기법 (10/11)

- Θ(Theta) **표기법** contd.
  - 예제 1: $5n^2 + 3n \in \Theta(n^2)$?
    - g(n): 5n2 +3n, 5·f(n): $5n^2$, 9·f(n): $9n^2$에 대한 그래프



모든 $n \geq n0$ (= 1)에 대해서 $5n^2 + 3n$은 그래프 상에서 $3n^2$ 위에 위치(아무리 수행 시간이 빨라도 $3n^2$과 같거나 느림) 하며 $9n^2$ 아래에 위치(아무리 수행 시간이 느려도 $9n^2$과 같거나 빠름)

알고리즘 복잡도의 점근적 표기법 (11/11)

참고: 복잡도 카테고리의 주요 성질

- $g(n) \in O(f(n))$ if and only if $f(n) \in \Omega(g(n))$
- $g(n) \in \Theta(f(n))$ if and only if $f(n) \in \Theta(g(n))$
- $b > 1$이고 $a > 1$이면, **$\log_a n \in \Theta(\log_b n)$은 항상 성립** → 로그(logarithm) 복잡도 함수는 모두 같은 카테고리에 속함
- 지수(exponential) 복잡도 함수는 모두 같은 카테고리에 속하지 않음
  - 임의의 서로 다른 a와 b에 대해서 **$a^n \notin \Theta(b^n)$**
  - 만약 $b > a > 0$ 이면, **$a^n \in o(b^n)$**
- n!은 어떤 지수 (a^n) 복잡도 함수보다도 느림
  - **$a^n \in o(n!)$**
- 복잡도 카테고리들은 다음 순서로 나열할 수 있음
  - $\Theta(\lg n), \Theta(n), \Theta(n \lg n), \Theta(n^2), \Theta(n^j), \Theta(n^k), \Theta(a^n), \Theta(b^n), \Theta(n!)$, where $k>j>2$ and $b>a>1$