4절 차수

- 아래 두 알고리즘 중에서 어떤 알고리즘 선택?
  - 알고리즘 A의 시간 복잡도: $100n$
  - 알고리즘 B의 시간 복잡도: $0.01n^2$

- $0.01n^2$ 과 $100n$ 중에 누구의 복잡도가 더 커보임?

- 정답: n의 크기에 따라 달라짐.
    - $n \leq 10,000$: 알고리즘 B 선택
    - $n > 10,000$: 알고리즘 A 선택

- 이유:

$$
\begin{aligned}
0.01n^2 > 100n \quad &\Longleftrightarrow \quad n^2 > 10000n \\
&\Longleftrightarrow \quad n > 10000
\end{aligned}
$$

"궁극적으로 더 빠름"

- '$n > 10,000$인 임의의 양의 정수 n에 대해 $0.01n^2$ 이 $100n$ 보다 크다'를 다르게 표현하면 다음과 같음.
    - $0.01n^2$ 이 $100n$ 보다 궁극적으로 크다

- 다음 성질을 갖는 정수 $N \geq 0$이 존재할 때 $f(n)$이 $g(n)$ 보다 궁극적으로 크다 라고 말함:
    - $n > N$인 임의의 양의 정수 n에 대해 $f(n) > g(n)$.

- 시간 복잡도의 기준으로 볼 경우:
    - $g(n)$이 $f(n)$ 보다 궁극적으로 빠르다 $\iff$ $f(n)$이 $g(n)$ 보다 궁극적으로 크다

차수(Θ, 쎄타)의 직관적 이해

$\Theta(n)$: 1차 시간 복잡도

- $100n$
- $0.001n + 100$
- 등등

$\Theta(n^2)$: 2차 시간 복잡도

- $5n^2$
- $0.1n^2 + n + 100$
- 등등

$\Theta(n^3)$: 3차 시간 복잡도

- $7n^3$
- n^3
$+ 5n^2$
$+ 100n$
$+ 2$
- 등등

고차항의 지배력

- 예제: $0.1n^2 + n + 100$에서 2차 항 $0.1n^2$이 함수 전체를 지배함

n	$0.1n^2$	$0.1n^2 + n + 100$
10	10	120
20	40	160
50	250	400
100	1,000	1,200
1,000	100,000	101,100

복잡도 카테고리의 직관적 이해

- 1차, 2차, 3차 등의 시간복잡도를 갖는 함수들의 집합을 복잡도 카테고리라고 함.

매우 효율적인 알고리즘의 복잡도 예제

$\Theta(1)$: 상수 복잡도

- 0
- 1
- 1000
- 1억 등등 모든 상수

$\Theta(\lg n)$: 로그 복잡도

- $\lg n$
- $2 \lg n$
- $\frac{1}{2} \cdot \lg n + 3$

$\Theta(n)$: 1차 복잡도

- n
- $100n$
- $0.001n$
  $+ 10000$

$\Theta(n \lg n)$: 엔 로그 엔(n log n) 복잡도

경우에 따라 괜찮은 알고리즘의 복잡도 예제

$\Theta(n^2)$: 2차 복잡도

- n^2
- $5n^2$
- $0.1n^2 + n + 100$

$\Theta(n^3)$: 3차 복잡도

- n^3
- $0.001n^3$
$+ 5n^2$
$+ 2n$
$+ 7$
- $100n^3$
$+ n$
$+ 100$

사실상 사용할 수 없는 알고리즘의 복잡도 예제

$\Theta(2^n)$: 지수 복잡도

$\Theta(n!)$: 팩토리얼 복잡도

복잡도 함수의 증가율

시간복잡도별 실행시간 비교

- 가정: 단위연산 실행시간 = 1 ns (원서 오류 주의: 약 0.230 초)

n	$\lg n$	n	$n \lg n$	n^2	n^3	2^n
10	0.003 μs	0.01 μs	0.033 μs	0.10 μs	1.0 μs	1 μs
20	0.004 μs	0.02 μs	0.086 μs	0.40 μs	8.0 μs	1 ms
30	0.005 μs	0.03 μs	0.147 μs	0.90 μs	27.0 μs	1 초
40	0.005 μs	0.04 μs	0.213 μs	1.60 μs	64.0 μs	18.3 분
50	0.006 μs	0.05 μs	0.282 μs	2.50 μs	125.0 μs	13 일
10^2	0.007 μs	0.10 μs	0.664 μs	10.00 μs	1.0 ms	4×10^{13} 년
10^3	0.010 μs	1.00 μs	9.966 μs	1.00 ms	1.0 초	
10^4	0.013 μs	10.00 μs	130.000 μs	100.00 ms	16.7 분	
10^5	0.017 μs	0.10 ms	1.670 ms	10.00 초	11.6 일	
10^6	0.020 μs	1.00 ms	19.930 ms	16.70 초	31.7 년	
10^7	0.023 μs	0.01 초	0.230 초	1.16 일	31, 709 년	
10^8	0.027 μs	0.10 초	2.660 초	115.70 일	3.17×10^7 년	
10^9	0.030 μs	1.00 초	29.900 초	31.70 년		

차수 정의

- 차수(Θ)를 엄밀하게 정의하려면 "큰 O(big O)"와 "Ω(Omega, 오메가)" 개념 필요

'큰 O' 표기법

- 다음 성질을 갖는 양의 실수 c와 음이 아닌 정수 N이 존재할 때 $g(n) \in O(f(n))$ 성립:
  - $n \geq N$인 임의의 정수 n에 대해 $g(n) \leq c \cdot f(n)$

- $g(n) \in O(f(n))$ 읽는 방법:
    - $g(n)$은 $f(n)$의 큰 O이다.
    - $g(n)$의 점근적 상한은 $f(n)$이다.

- 의미: 입력크기 n에 대해 시간 복잡도 $g(n)$의 수행시간은 궁극적으로 $f(n)$보다 나쁘지는 않다.

'큰 O' 표기법 예제

- $n^2 + 10n \in O(n^2)$
  - $n \geq 10$인 경우: $n^2 + 10n$
    $\leq 2n^2$
  - 그러므로 $c = 2$와 $N = 10$ 선택

- $n^2 + 10n \in O(n^2)$:
  - $n \geq 1$인 경우: $n^2 + 10n$
    $\leq n^2$
    $+ 10n^2$
    $= 11n^2$
  - 그러므로 $c = 11$와 $N = 1$ 선택

- $2n^2$ 과 $n^2 + 10n$ 의 비교

- $5n^2 \in O(n^2)$
  - $n \geq 0$인 경우: $5n^2 \leq 5n^2$
  - 그러므로 $c = 5$와 $N = 0$ 선택

- $T(n) = n(n-1)/2$
  $\in O(n^2)$
  - $n \geq 0$인 경우: $n(n-1)/2$
    $\leq n^2/2$
  - 그러므로 $c = 1/2$과 $N = 0$ 선택

- $n^2 \in O(n^2 + 10n)$
  - $n \geq 0$인 경우: $n^2 \leq 1 \times (n^2 + 10n)$
  - 그러므로 $c = 1$과 $N = 0$ 선택

- $n \in O(n^2)$
    - $n \geq 1$인 경우: $n \leq 1 \times n^2$이 성립한다.
    - 그러므로 $c = 1$과 $N = 1$ 선택

- $n^3 \notin O(n^2)$
    - c와 N을 아무리 크게 지정하더라도, N 보다 큰 어떤 수 n에 대해 $n^3 > c \cdot n^2$이 성립한다.
    - 예를 들어, $n > c$로 잡으면 됨.

- $O(n^2)$: 특정 양의 실수 c에 대해 $c\,n^2$ 보다 궁극적으로 작은 값을 가지는 함수들의 집합

Ω 표기법

- 다음 성질을 갖는 양의 실수 c와 음이 아닌 정수 N이 존재할 때 $g(n) \in \Omega(f(n))$ 성립:
  - $n \geq N$인 임의의 정수 n에 대해 $g(n) \geq c \cdot f(n)$

- $g(n) \in \Omega(f(n))$ 읽는 방법:
  - $g(n)$은 $f(n)$의 오메가이다.
  - $g(n)$의 점근적 하한은 $f(n)$이다.

- 의미: 입력크기 n에 대해 시간 복잡도 $g(n)$의 수행시간은 궁극적으로 $f(n)$보다 효율적이지 못하다.

Ω 표기법 예제

- $n^2 + 10n \in \Omega(n^2)$
  - $n \geq 0$인 경우: $n^2 + 10n$
    $$\geq n^2$$
  - 그러므로 $c = 1$과 $N = 0$ 선택

- $5n^2 \in \Omega(n^2)$
  - $n \geq 0$인 경우: $5n^2 \geq 1 \cdot n^2$
  - 그러므로, $c = 1$과 $N = 0$ 선택

- $T(n) = n(n-1)/2$
  $\in \Omega(n^2)$
  - $n \geq 2$인 경우: $\frac{n(n-1)}{2}$
    $\geq \frac{1}{4}n^2$
  - 그러므로 $c = 1/4$과 $N = 2$ 선택

- $n^3 \in \Omega(n^2)$
  - $n \geq 1$인 경우: $n^3 \geq 1 \cdot n^2$
  - 그러므로, $c = 1$과 $N = 1$ 선택

- $n \notin \Omega(n^2)$
    - c를 아무리 작게, N을 아무리 크게 지정하더라도, $n \le c \cdot n^2$을 만족시키는 $n \ge N$이 존재.
    - 예를 들어, $n \ge 1/c$로 잡으면 됨.

- $\Omega(n^2)$: 특정 양의 실수 c에 대해 $c\,n^2$ 보다 궁극적으로 큰 값을 가지는 함수들의 집합

Θ 표기법

- $g(n) \in \Theta(f(n))$ 읽는 방법:
    - $g(n)$의 $f(n)$의 **차수**이다.

Θ 표기법 예제

- $T(n) = n(n-1)/2$:
  $\in \Theta(n^2)$
  - $n \geq 2$인 경우: $n(n-1)/2 \geq \frac{1}{4}n^2$
  - $n \geq 0$인 경우: $n(n-1)/2 \leq \frac{1}{2}n^2$
  - 그러므로, $c = \frac{1}{4}, d = \frac{1}{2}, N = 2.$

작은 o(small o) 표기법

- 임의의 양의 실수 c에 대해 다음 성질을 갖는 음이 아닌 정수 N이 존재할 때 $g(n) \in o(f(n))$ 성립:
  - $n \geq N$인 임의의 정수 n에 대해 $g(n) \leq c \cdot f(n)$

- $g(n) \in o(f(n))$ 읽는 방법:
  - $g(n)$은 $f(n)$의 '작은 오(small o)이다.

- 의미
    - $g(n)$이 $f(n)$에 비해 궁극적으로 하찮을 만큼 작다.
    - 알고리즘 분석적 측면: 복잡도 $g(n)$이 복잡도 $f(n)$ 보다 궁극적으로 훨씬 좋다.
        - 이유: $c > 0$이 아무리 작더라도, n이 충분히 크면 $g(n) < f(n)$ 성립하기 때문.

큰 O vs 작은 o

- 큰 O: 하나의 양의 실수 c에 대해서 부등식 성립
- 작은 o: 모든 양의 실수 c에 대해서 부등식 성립

작은 o 표기법 예제

- $n \in o(n^2)$
  - $c > 0$가 주어졌을 때, $n \geq 1/c$인 모든 n에 대해 $n \leq c \cdot n^2$ 성립.

- $n \notin o(5n)$
  - $c < 1/5$인 경우, 임의의 음이 아닌 정수 n에 대해 $n > c \cdot 5n$ 성립.

- $n^2 \notin o(5n)$
  - n 이기 때문.
    $\notin o(5n$
    $)$

작은 o 특성

- $g(n) \in o(f(n))$ 이면 다음도 성립:

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

- 증명: 생략.

- **주의**
  - $o(f(n)) \neq O(f(n)) - \Omega(f(n))$
- 다음 함수 $g(n)$에 대해 $g(n) \in O(n) - \Omega(n)$이지만 $g(n) \notin o(n)$임:

$$g(n) = \begin{cases} n & \text{if } n\%2 = 0 \\ 1 & \text{if } n\%2 = 1 \end{cases}$$

차수의 특성

- $g(n) \in O(f(n)) \quad \Longleftrightarrow \quad f(n) \in \Omega(g(n))$

- $g(n) \in \Theta(f(n)) \quad \Longleftrightarrow \quad f(n) \in \Theta(g(n))$

- 임의의 $a, b > 1$에 대해

$$\log_a n \in \Theta(\log_b n)$$

즉, 로그 함수는 모두 동일한 복잡도 카테고리에 속함.

- $b > a > 0$이면 다음 성립:

$$a^n \in o(b^n)$$

즉, 지수 함수는 밑수가 다르면 다른 복잡도 카테고리에 속함.

- 임의의 양의 실수 a에 대해 다음 성립:

$$a^n \in o(n!)$$

즉, $n!$은 어떠한 지수 복잡도함수보다 더 나쁘다(느리다).

- 많이 언급되는 복잡도 카테고리를 순서대로 나열하면 다음과 같음:

$$\Theta(\lg n) \quad \Theta(n) \quad \Theta(n \lg n) \quad \Theta(n^2) \quad \Theta(n^j) \quad \Theta(n^k) \quad \Theta(a^n) \quad \Theta(b^n) \quad \Theta(n!)$$

    - 단, $k > j > 2$이고 $b > a > 1$임.
    - $g(n)$이 $f(n)$의 카테고리 보다 왼편에 위치한 카테고리에 속한 경우 다음 성립:

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

- $c \geq 0, d > 0, g(n) \in O(f(n)), h(n) \in \Theta(f(n))$ 인 경우 다음 성립:

$$c \cdot g(n) + d \cdot h(n) \in \Theta(f(n))$$

예제

- $\Theta(\log_4 n) \in \Theta(\lg n)$
- $\lg n \in o(n)$

- $7n^2 \in \Theta(n^2)$
- $10n \lg n + 7n^2 \in \Theta(n^2)$
- $3 \lg n$

극한(limit)을 이용하여 차수를 구하는 방법

정리

- $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ 의 값이
    - 만약 $c > 0$이면, $g(n) \in \Theta(f(n))$,
    - 만약 0 이면, $g(n) \in o(f(n))$,
    - 만약 ∞, 즉, 발산하면, $f(n) \in o(g(n))$.

예제

- $\frac{n^2}{2} \in o(n^3)$:

$$\lim_{n \to \infty} \frac{n^2/2}{n^3} = \lim_{n \to \infty} \frac{1}{2n} = 0$$

- $b > a > 0$일 때, $a^n \in o(b^n)$:

$$\lim_{n\to\infty} \frac{a^n}{b^n} = \lim_{n\to\infty} \left(\frac{a}{b}\right)^n = 0$$

- $a > 0$일 때, $a^n \in o(n!)$
  - 증명 생략.

로피탈(L'Hopital)의 법칙

아래 조건이 성립한다고 가정하자.

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

그러면 다음이 성립한다.

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

예제

- 다음이 성립한다.

$$\lg n \in o(n)$$

- 이유

$$\lim_{n \to \infty} \frac{\lg n}{n} = \lim_{n \to \infty} \left(\frac{\frac{1}{n \ \ln 2}}{1} \right) = 0$$

예제

- 다음이 성립한다.

$$\log_a n \in \Theta(\log_b n)$$

- 이유

$$\lim_{n\to\infty} \frac{\log_a n}{\log_b n} = \lim_{n\to\infty} \left(\frac{\frac{1}{n \ \ln a}}{\frac{1}{n \cdot \ln b}} \right) = \frac{\log b}{\log a} > 0$$