Ch.01 알고리즘이란

학습목표

- ✓ 알고리즘의 필요성과 자료구조와의 관계를 이해한다.
- ✓ 이 책에서 사용하는 알고리즘 표기법을 이해한다.

알고리즘은 작업 과정의 묘사

- 어떤 작업을 수행하기 위한 과정을 애매하지 않게 기술한 것
- 어떤 작업을 수행하기 위해 입력을 받아 출력을 만들어내는 과정을 애매하지 않게 기술한 것

바람직한 알고리즘

- **명확해야 한다**
  - 이해하기 쉽고 가능하면 간명하도록
  - 지나친 기호적 표현은 오히려 명확성을 떨어뜨림
  - 명확성을 해치지 않으면 일반언어의 사용도 무방

- **효율적이어야 한다**
  - 같은 문제를 해결하는 알고리즘들의 수행 시간이 수백만 배 이상 차이날 수 있다

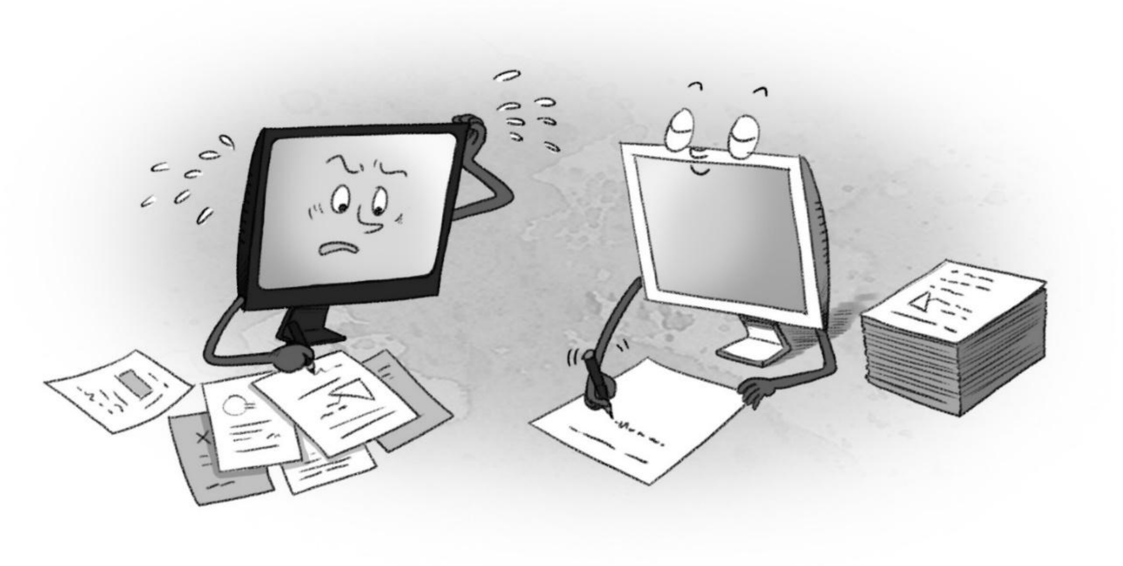


그림 1-1 같은 문제를 해결하는 데 100년이 걸리는 알고리즘 vs. 1분이 걸리는 알고리즘

입출력과 알고리즘 예

- **문제**
  - 100명의 학생의 시험점수의 최대값을 찾으라
- **입력**
  - 100개의 점수
- **출력**
  - 입력된 100개의 점수들 중 최댓값
- **알고리즘**

maxScore $(x[\,], n)$:

$x[1 \ldots n]$의 값을 차례대로 보면서 최댓값을 계산한다

return 위에서 찾은 최댓값

알고리즘은 생각하는 방법의 훈련

- 문제 자체를 해결하는 알고리즘을 배운다
- 그 과정에 깃든 '생각하는 방법'을 배우는 것이 더 중요하다
- 미래에 다른 문제를 해결하는 생각의 빌딩블록을 제공한다

그림 1-2 알고리즘은 생각하는 방법을 훈련하는 도구

알고리즘은 자료구조의 확장

선행 과목

- 프로그래밍, 자료구조

자료구조

- 건축의 건축 자재나 모듈 같은 것
- 자동차 제작의 부품이나 모듈 같은 것

그림 1-3 부품(자료구조) 선택의 중요성

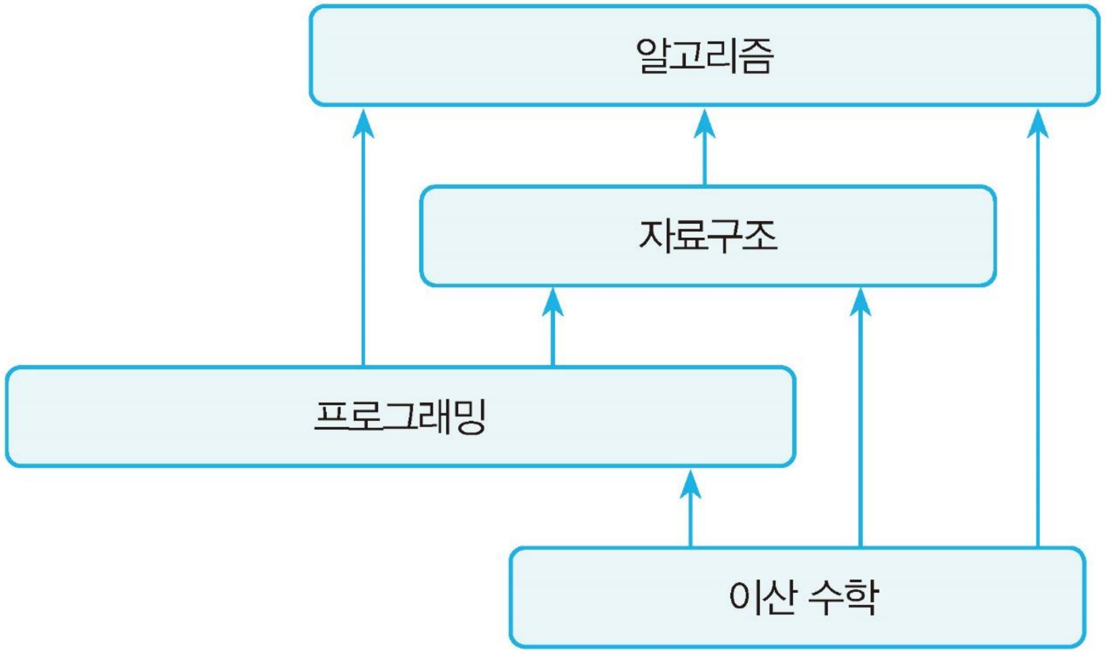


그림 1-4 알고리즘, 자료구조, 프로그래밍, 이산 수학의 관계

알고리즘 표기법

- **자연어를 이용한 서술적 표현**
  - 서술적일 뿐만 아니라 쓰는 사람에 따라 일관성이나 명확성을 유지하기 어려움
  - 누구라도 쉽게 이해하고 쓸 수 있어야 하는 알고리즘을 표현하는 데는 한계가 있음
- **순서도를 이용한 도식화**
  - 명령의 흐름을 쉽게 파악할 수 있지만 복잡한 알고리즘을 표현하는 데는 한계가 있음
- **프로그래밍 언어를 이용한 구체화**
  - 추가로 구체화할 필요가 없으나 해당 언어를 모르면 이해하기 어려움
  - 다른 프로그래밍 언어로 프로그램을 개발하는 경우에는 다른 프로그래밍 언어로 변환해야 하므로 범용성이 떨어짐
- **가상코드를 이용한 추상화**
  - 가상코드Pseudo-Code는 직접 실행할 수는 없지만 일반적인 프로그래밍 언어와 형태가 유사해 프로그래밍 언어로 구체화하기가 쉬움

이 책에서 사용하는 알고리즘 표기법

알고리즘 4-4 버블 정렬 2

```
bubbleSort(A[], n): ❶   ▷ A[0...n-1]을 정렬한다.
    sorted ← FALSE ❷
    for last ← n-1 downto 1
❸       sorted ← TRUE
        for i ← 0 to last-1
            if (A[i] > A[i+1])
❹               A[i] ↔ A[i+1]   ❺ ▷ 원소 교환
                sorted ← FALSE
        if (sorted = TRUE) return
```

❶ 함수 시작

❷ 문장 뒤 세미콜론 생략

❸ for 루프에 속하는 문장 들여쓰기

❹ If 뒤의 then 생략

❺ 주석

Ch.02 알고리즘 설계와 분석의 기초

학습목표

- ✓ 알고리즘을 설계하고 분석하는 몇 가지 기초 개념을 이해한다.
- ✓ 가장 기초적인 알고리즘의 수행 시간을 분석할 수 있도록 한다.
- ✓ 점근적 표기법을 이해한다.

알고리즘 분석의 필요성

- **같은 문제에서도 효율 면에서 아주 큰 차이가 나는 다양한 알고리즘이 존재**
  - 정렬에서, 입력의 크기가 n일 때,
    최악의 경우 n^2에 비례하는 시간을 소모하는 알고리즘도 있고
    nlogn에 비례하는 알고리즘도 있음

- **알고리즘을 학습하는 과정에서 얻을 수 있는 여러 가지 기법과 생각하는 방법 훈련 가능**

알고리즘의 수행 시간

- **입력의 크기 n에 대해 시간이 얼마나 걸리는지로 표현한다**
- **입력의 크기는 대부분 자명함**
  - 정렬: 정렬할 원소의 수
  - 색인: 색인에 포함된 원소의 수

  …

알고리즘 수행 시간의 예

- 알고리즘 예 1

```
sample1(A[], n):
    k = ⌊n/2⌋
    return A[k]
```

상수 시간

- 알고리즘 예 2

```
sample2(A[], n):
    sum ← 0
    for i ← 1 to n
        sum ← sum + A[i]
    return sum
```

n에 비례하는 시간

- 알고리즘 예 3

```
sample3(A[], n):
    sum ← 0
    for i ← 1 to n
        for j ← 1 to n
            sum ← sum + A[i] * A[j]
    return sum
```

n^2에 비례하는 시간

알고리즘 수행 시간의 예

■ 알고리즘 예 4

```
sample4(A[], n):
    sum ← 0
    for i ← 1 to n
        for j ← 1 to n
            ❶ k ← A[1...n]에서 임의로 ⌊n/2⌋개를 뽑을 때 이들 중 최댓값
            ❷ sum ← sum + k
    return sum
```

— n^3에 비례하는 시간

■ 알고리즘 예 5

```
sample5(A[], n):
    sum ← 0
    for i ← 1 to n-1
        for j ← i + 1 to n
            ❶ sum ← sum + A[i] * A[j]
    return sum
```

— n^2에 비례하는 시간

■ 팩토리얼

```
factorial(n):
    if (n=1) return 1
  ❶ return n * factorial(n-1)
```

— n에 비례하는 시간

재귀와 귀납적 사고

- **재귀=자기호출(recursion)**
- **재귀적 구조**
  - 어떤 문제 안에 크기만 다를 뿐 성격이 똑같은 작은 문제(들)가 포함되어 있는 것
  - 예1: factorial
    - $N! = N \times (N-1)!$
  - 예2: 수열의 점화식
    - $a_n = a_{n-1} + 2$

그림 2-1 자기호출의 개념이 담긴 마트료시카

재귀의 예: 병합 정렬

알고리즘 2-1 **병합 정렬**

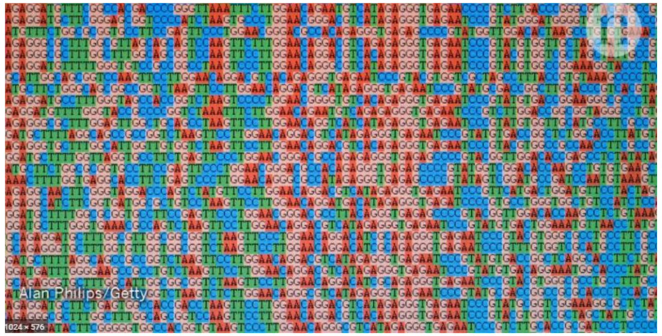
mergeSort(A[], p, r): ▷ $A[p...r]$을 정렬한다.
  if ($p < r$)
    ❶ $q \leftarrow \lfloor (p+r)/2 \rfloor$ ▷ p, r의 중간 지점 계산
    ❷ mergeSort(A, p, q) ▷ 전반부 정렬
    ❸ mergeSort(A, q+1, r) ▷ 후반부 정렬
    ❹ merge(A, p, q, r) ▷ 병합

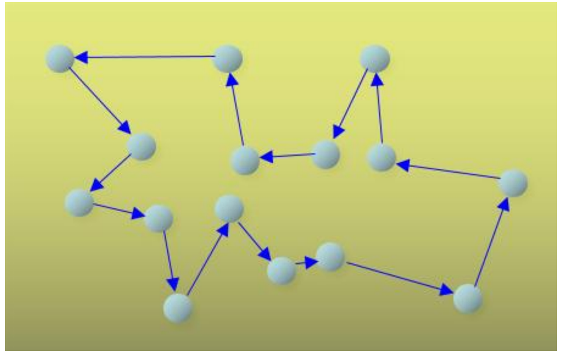
merge(A[], p, q, r):
  정렬되어 있는 두 배열 $A[p...q]$와 $A[q+1...r]$을 합쳐
  정렬된 하나의 배열 $A[p...r]$을 만든다.

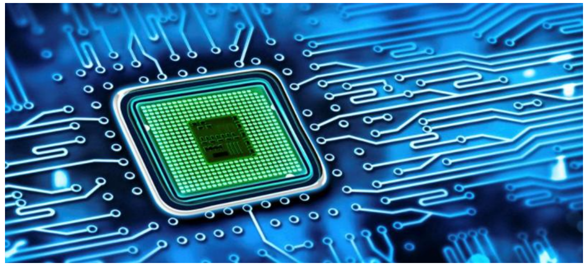
- ✓ ❷, ❸은 재귀호출
- ✓ ❶, ❹는 재귀적 관계를 드러내기 위한 오버헤드

다양한 알고리즘의 적용 주제들

- **자동차 네비게이션**
- **스케줄링**
  - TSP, 차량 라우팅, 작업 공정, ...
- **인간 게놈 프로젝트**
  - 매칭, 최적의 진화 계통도, ...
- **검색**
  - 데이터베이스, 웹페이지들, ...
- **자원의 배치**
- **반도체 설계**
- **...**







알고리즘의 분석

- **크기가 작은 문제**
  - 알고리즘의 효율성이 중요하지 않다
  - 비효율적인 알고리즘도 무방

- **크기가 충분히 큰 문제**
  - 알고리즘의 효율성이 중요하다
  - 비효율적인 알고리즘은 치명적

- **입력의 크기가 충분히 큰 경우에 대한 분석을 점근적 분석이라 한다**

알고리즘의 분석

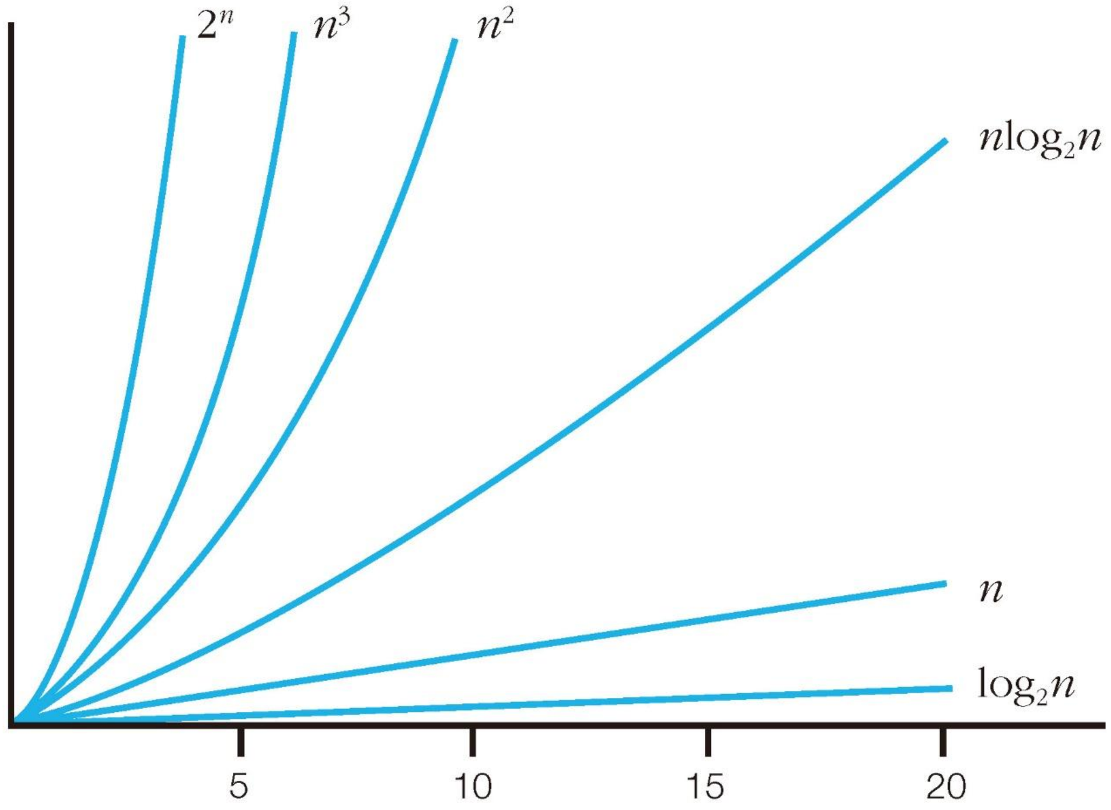


그림 2-3 여러 함수의 증가율 비교

표 2-1 두 함수 10n과 $2n^2$의 증가율 비교

n	10n	$2n^2$
1	10	2
2	20	8
10	100	200
100	1,000	20,000
1,000	10,000	2,000,000
10,000	100,000	200,000,000
100,000	1,000,000	20,000,000,000
1,000,000	10,000,000	2,000,000,000,000
10,000,000	100,000,000	200,000,000,000,000
...	...	...
10^{10}	10^{11}	$2 \cdot 10^{20}$
10^{20}	10^{21}	$2 \cdot 10^{40}$

대표적인 점근적 표기법

- O(빅오)-표기: 점근적 상한
- Ω(오메가)-표기: 점근적 하한
- Θ(세타)-표기: 점근적 동일

그림 2–4 대표적인 점근적 표기법

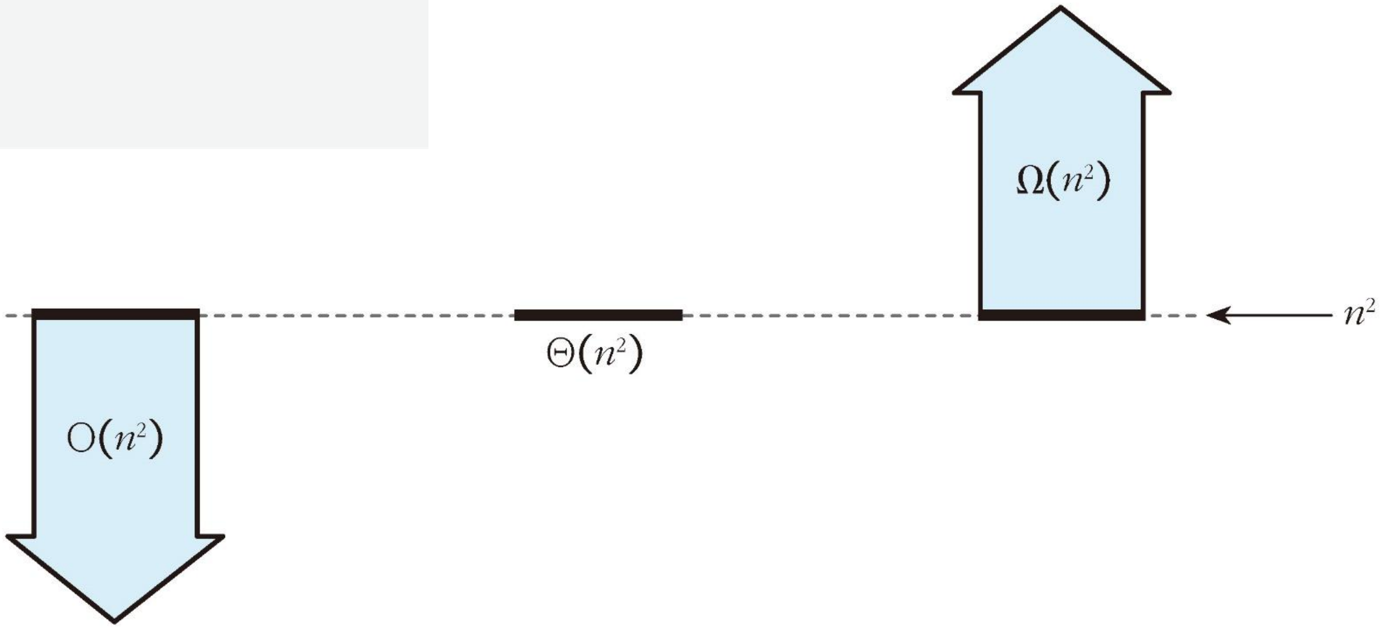


그림 2–5 대표적인 점근적 표기의 시각적 표현

$O(g(n))$ – big Oh

기껏해야(at most) $g(n)$의 비율로 증가하는 함수

e.g., $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$, …

- 수학적 표기
  - $O(g(n)) = \{ f(n) \mid \exists c > 0, n_0 \geq 0 \text{ s.t.} \forall n \geq n_0, f(n) \leq cg(n) \}$
    $f(n) \in O(g(n))$을 관행적으로 $f(n) = O(g(n))$라고 쓴다.

- 직관적 의미
  - $f(n) = O(g(n)) \Rightarrow f$는 g 보다 빠르게 증가하지 않는다.
    상수 비율의 차이는 무시

- 예
  - $O(n^2) = \{3n^2 + 2n,\ 7n^2 - 100n,\ n\log n + 5n,\ 3n,\ \ldots\}$

$O(g(n))$ – big Oh

예제 2-1

$5n^2=O(n^2)$임을 보여라.

예제 2-2

$5n^2+3=O(n^2)$임을 보여라.

예제 2-3

$\frac{n^2}{2}-5=O(n^2)$임을 보여라.

$\Omega(g(n))$ – big Omega

적어도(at least) $g(n)$의 비율로 증가하는 함수
$O(g(n))$과 대칭적

- 수학적 표기
  - $\Omega(g(n)) = \{ f(n) \mid \exists c > 0, n_0 \geq 0 \text{ s.t.} \forall n \geq n_0, f(n) \geq cg(n) \}$
- 직관적 의미
  - $f(n) = \Omega(g(n)) \Rightarrow f$ 는 g 보다 느리게 증가하지 않는다.
- 예
  - $\Omega(n^2) = \{3n^2 + 2n, 7n^2 - 100n, n^3 + n\log n + 5n, 2^n + 3n, \ldots\}$

$\Omega(g(n))$ – big Omega

예제 2-5

$5n^2 = \Omega(n^2)$임을 보여라.

예제 2-6

$5n^2 + 3 = \Omega(n^2)$임을 보여라.

예제 2-7

$\frac{n^2}{2} - 5 = \Omega(n^2)$임을 보여라.

예제 2-8

$5n^3 + 3 = \Omega(n^2)$임을 보여라.

$\Theta(g(n))$ – big Theta

$\boldsymbol{g(n)}$의 비율로 증가하는 함수

- 수학적 표기
  - $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$
- 직관적 의미
  - $f(n) = \Theta(\boldsymbol{g}(n)) \Rightarrow \boldsymbol{f}$는 $\boldsymbol{g}$와 같은 정도로 증가한다
- 예
  - $\Theta(n^2) = \{7n^2 + 9n + 4,\ 15n^2 - 100n,\ 2n^2 - 1000n,\ \ldots\}$

$\Theta(g(n))$ – big Theta

예제 2-9

$5n^2=\Theta(n^2)$임을 보여라.

예제 2-10

$5n^2+3=\Theta(n^2)$임을 보여라.

예제 2-11

$\frac{n^2}{2}-5=\Theta(n^2)$임을 보여라.

$o(g(n))$ – little oh

$\boldsymbol{g(n)}$보다 느린 비율로 증가하는 함수

- 수학적 표기
  - $o(\mathbf{g}(n)) = \{\, \boldsymbol{f}(n) \mid \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\}$
- 직관적 의미
  - $\boldsymbol{f}(n) = o(\mathbf{g}(n)) \Rightarrow \boldsymbol{f}$ 는 $\boldsymbol{g}$보다 느리게 증가한다
- 예
  - $o(n^2) = \{9n + 4,\ 100n\log n + 25n,\ 2n - 1000,\ 5n^{1.99} + 17n + 4,\ \ldots\}$

$o(g(n))$ – little oh

예제 2-12

$n^2-5=o(n^3)$임을 보여라.

$\omega(g(n))$ – little omega

$g(n)$보다 빠른 비율로 증가하는 함수

- 수학적 표기
  - $\omega(g(n)) = \{ f(n) \mid \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \}$
- 직관적 의미
  - $f(n) = \omega(g(n)) \Rightarrow f$ 는 g보다 빠르게 증가한다.
- 예
  - $\omega(n^2) = \{3n^2\log 3n + 2n,\ 7n^3 - 100n,\ n^4 + n\log n + 5n,\ 2^n + 3n,\ 0.5n^{2.01} - 59n - 45,\ \ldots\}$

$\omega(g(n))$ – little omega

예제 2-13

$\frac{n^3}{4} = \omega(n^2)$임을 보여라.

각 점근적 표기법의 직관적 의미

- $O(g(n))$
  - Tight *or* loose upper bound
- $\Omega(g(n))$
  - Tight *or* loose lower bound
- $\Theta(g(n))$
  - Tight bound
- $o(g(\mathrm{n}))$
  - Loose upper bound
- $\omega(g(n))$
  - Loose lower bound

그림 2-7 O와 o의 차이

각 점근적 표기법의 직관적 의미

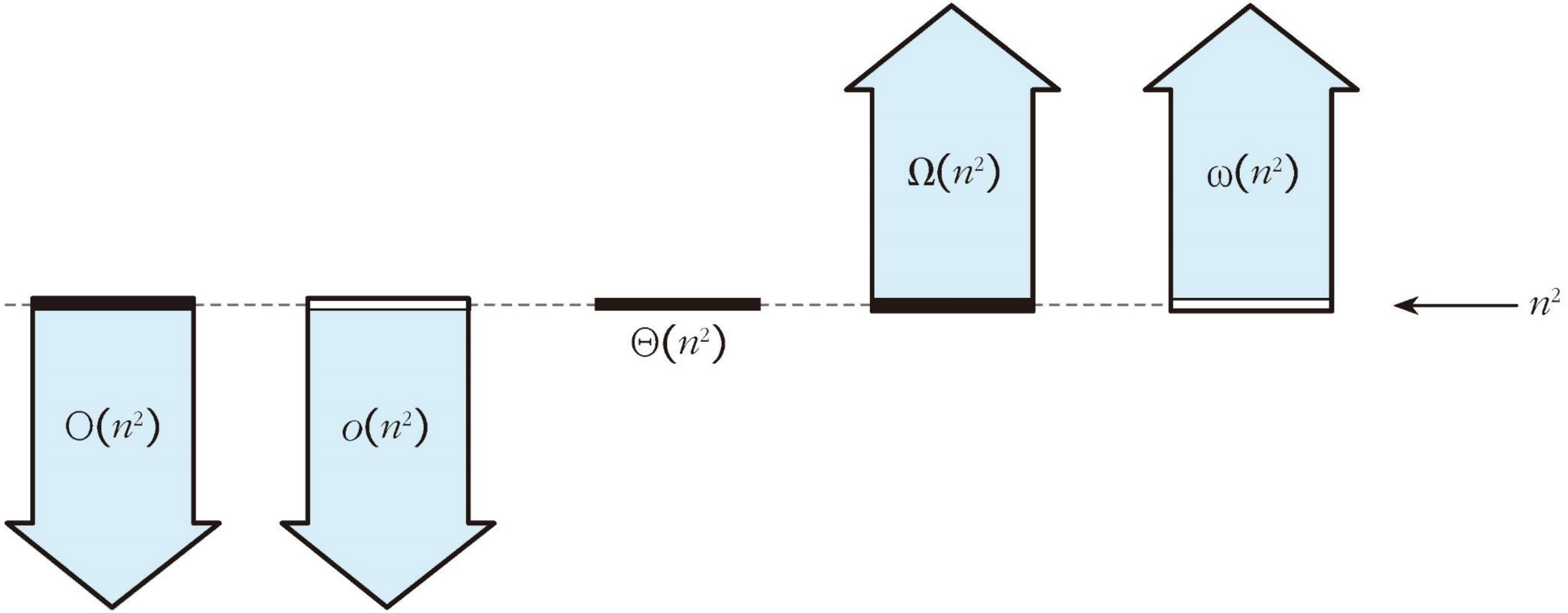


그림 2-6 **점근적 표기의 시각적 표현**

Ch.03 점화식과 알고리즘 복잡도 분석

학습목표

- ✓ 재귀 알고리즘과 점화식의 관계를 이해한다.
- ✓ 점화식의 점근적 분석을 이해한다.

점화식

- 어떤 함수를 자신보다 더 작은 변수에 대한 함수와의 관계로 표현한 것

예

$$a_n = a_{n-1} + 2$$
$$f(n) = n\,f(n-1)$$
$$f(n) = f(n-1) + f(n-2)$$
$$f(n) = f(n/2) + n$$

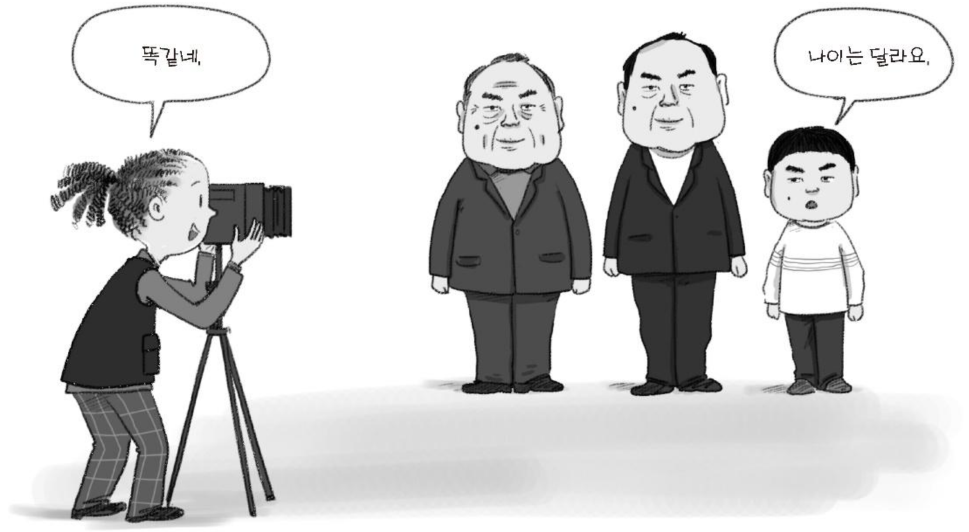


그림 3-1 점화식의 일상 비유

병합 정렬의 수행 시간

```
mergeSort(A[], p, r):        ▷ A[p...r]을 정렬한다.
    ❶ if (p < r)
        ❷ q ← ⌊(p+r)/2⌋         ▷ p, r의 중간 지점 계산
        ❸ mergeSort(A, p, q)     ▷ 전반부 정렬
        ❹ mergeSort(A, q+1, r)   ▷ 후반부 정렬
        ❺ merge(A, p, q, r)      ▷ 병합
```

- **수행 시간의 점화식** $T(n) = 2T(\frac{n}{2}) + \text{후처리 시간}$
  - 크기가 n인 병합 정렬 시간은 크기가 $n/2$인 병합 정렬을 두 번하는 시간과 나머지 오버헤드를 더한 시간이다.
  - $T(n) = T(\frac{n}{4}) + c$

점화식의 점근적 분석 방법

- **반복 대치(Iteration)**
  - 더 작은 문제에 대한 함수로 반복해서 대치해 나가는 해법
- **추정후 증명(Guess & Verification)**
  - 결론을 추정하고 수학적 귀납법으로 이용하여 증명하는 방법
- **마스터 정리(Master Theorem)**
  - 형식에 맞는 점화식의 복잡도를 바로 알 수 있다

반복 대치

$T(n) = T(n-1) + c$ (c는 자기호출을 제외한 나머지의 수행 시간)

$$
\begin{aligned}
T(n) &= T(n-1) + c \\
&= T(n-2) + c + c = T(n-2) + 2c \\
&= T(n-3) + c + 2c = T(n-3) + 3c \\
&\cdots \\
&= T(1) + (n-1)c \\
&\le c + (n-1)c \\
&= cn
\end{aligned}
$$

따라서 $T(n) \le cn$이므로 $T(n) = O(n)$이다.

반복 대치

$$\boxed{T(n) \le 2T(\frac{n}{2}) + n}$$

$$\begin{aligned}
T(n) &\le 2T(\frac{n}{2}) + n \\
&\le 2(2T(\frac{n}{4}) + \frac{n}{2}) + n = 2^2 T(\frac{n}{2^2}) + 2n \\
&\le 2^2(2T(\frac{n}{2^3}) + \frac{n}{2^2}) + 2n = 2^3 T(\frac{n}{2^3}) + 3n \\
&\dots \\
&\le 2^k T(\frac{n}{2^k}) + kn = nT(1) + kn = n + n\log n \quad \leftarrow T(1)=1\text{이고 } k=\log_2 n\text{이기 때문} \\
&= O(n\log n)
\end{aligned}$$

반복 대치

$$\boxed{T(n) \le 2T(\frac{n}{2}) + O(n)}$$

$$\begin{aligned}
T(n) &\le 2T(\frac{n}{2}) + \mathrm{O}(n) \\
&\le 2(2T(\frac{n}{4}) + \frac{n}{2}) + \mathrm{O}(n) = 2^2 T(\frac{n}{2^2}) + 2\mathrm{O}(n) \\
&\le 2^2(2T(\frac{n}{2^3}) + \frac{n}{2^2}) + 2\mathrm{O}(n) = 2^3 T(\frac{n}{2^3}) + 3\mathrm{O}(n) \\
&\dots \\
&\le 2^k T(\frac{n}{2^k}) + k\,\mathrm{O}(n) = nT(1) + k\,\mathrm{O}(n) = nT(1) + \log n \cdot \mathrm{O}(n) \\
&\le nT(1) + \mathrm{O}(kn) = n + \mathrm{O}(n \log n) \\
&= \mathrm{O}(n \log n)
\end{aligned}$$

추정후 증명

예제 3-1

$T(n) \le 2T(\frac{n}{2}) + n$의 점근적 복잡도는 $T(n) = O(n\log n)$임을 보여라. 즉, 충분히 큰 n에 대하여 $T(n) \le cn\log n$인 양의 상수 c가 존재한다.

▶▶ **증명** 경계조건: $T(2) \le c2\log 2$를 만족하는 c가 존재한다.

귀납적 가정과 전개: $\frac{n}{2}$에 대해 $T(\frac{n}{2}) \le c(\frac{n}{2})\log\frac{n}{2}$을 만족한다고 가정하면,

$$
\begin{aligned}
T(n) &\le 2T(\frac{n}{2}) + n \\
① &\le 2c(\frac{n}{2})\log(\frac{n}{2}) + n \\
&= cn\log n - cn\log 2 + n \\
&= cn\log n + \underbrace{(-c\log 2 + 1)n}_{②} \\
③ &\le cn\log n
\end{aligned}
$$

추정후 증명

예제 3-2

$T(n) \le 2T(\frac{n}{2}+10)+n$의 점근적 복잡도는 $T(n)=O(n\log n)$임을 보여라. 즉, 충분히 큰 n에 대하여 $T(n) \le cn\log n$인 양의 상수 c가 존재한다.

▶▶ **증명**

$$
\begin{aligned}
T(n) &\le 2T(\frac{n}{2}+10)+n \\
❶ &\le 2c(\frac{n}{2}+10)\log(\frac{n}{2}+10)+n \\
&= cn\log(\frac{n}{2}+10)+20c\log(\frac{n}{2}+10)+n \\
❷ &\le cn\log\frac{3n}{4}+20c\log\frac{3n}{4}+n \\
&= cn\log n + cn(\log 3-\log 4)+20c\log\frac{3n}{4}+n \\
&= cn\log n + \underbrace{[c(\log 3-\log 4)+1]n+20c\log\frac{3n}{4}}_{❸} \\
❹ &\le cn\log n
\end{aligned}
$$

추정후 증명

예제 3-3

$T(n)=2T(\frac{n}{2})+1$의 점근적 복잡도는 $O(n)$임을 보여라.

▶▶ **실패하는 증명** 충분히 큰 n에 대해 $T(n)\le cn$인 양의 상수 c가 존재한다는 것을 증명하려 한다.

$$\begin{aligned} T(n) &= 2T(\frac{n}{2})+1 \\ &\le 2c(\frac{n}{2})+1 \quad \leftarrow \text{귀납적 가정 이용} \\ &= cn+1 \end{aligned}$$

여기서 더 이상 진행이 되지 않는다. $cn+1\le cn$임을 증명할 수 없기 때문이다.

▶▶ **성공하는 증명** 미묘한 차이가 앞을 가로막고 있다. 발상의 전환이 필요하다. $T(n)\le cn$임을 증명하는 대신 $T(n)\le cn-2$임을 증명할 수 있다면 여전히 $T(n)=O(n)$이다.

$$\begin{aligned} T(n) &= 2T(\frac{n}{2})+1 \\ &\le 2(c\frac{n}{2}-2)+1 \quad \leftarrow \text{귀납적 가정 이용} \\ &= cn-3 \\ &\le cn-2 \end{aligned}$$

마스터 정리

- $T(n) = aT(n/b) + f(n)$와 같은 점화식에서 $n^{\log_b a} = h(n)$이라 하자

정리 3-1 마스터 정리

① 어떤 양의 상수 ε에 대하여 $\frac{f(n)}{h(n)} = O(\frac{1}{n^{\varepsilon}})$이면, $T(n) = \Theta(h(n))$이다.

② 어떤 양의 상수 ε에 대하여 $\frac{f(n)}{h(n)} = \Omega(n^{\varepsilon})$이고, 어떤 상수 $c(<1)$와 충분히 큰 모든 n에 대해 $af(\frac{n}{b}) \le cf(n)$이면 $T(n) = \Theta(f(n))$이다.

③ $\frac{f(n)}{h(n)} = \Theta(1)$이면 $T(n) = \Theta(h(n)\log n)$이다.

마스터 정리의 직관적 이해

마스터 정리의 근사 버전

1. $\lim_{n\to\infty} \frac{f(n)}{h(n)} = 0$이면 $T(n) = \Theta(h(n))$이다.
2. $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \infty$이고, 충분히 큰 모든 n에 대해 $af(\frac{n}{b}) < f(n)$이면 $T(n) = \Theta(f(n))$이다.
3. $\frac{f(n)}{h(n)} = \Theta(1)$이면 $T(n) = \Theta(h(n)\log n)$이다(마스터 정리의 3항 그대로임).

① $h(n)$이 더 무거우면 $h(n)$이 수행 시간을 결정한다.

② $f(n)$이 더 무거우면 $f(n)$이 수행 시간을 결정한다.

③ $h(n)$과 $f(n)$이 같은 무게이면 $h(n)$에 $\log n$을 곱한 것이 수행 시간이 된다.

✓ 원 정리와 약간 차이가 있지만 직관적 이해를 위해서 도움이 된다.

마스터 정리의 근사 버전

마스터 정리의 근사 버전

❶ $\lim_{n\to\infty}\frac{f(n)}{h(n)}=0$이면 $T(n)=\Theta(h(n))$이다.

❷ $\lim_{n\to\infty}\frac{f(n)}{h(n)}=\infty$이고, 충분히 큰 모든 n에 대해 $af(\frac{n}{b})<f(n)$이면 $T(n)=\Theta(f(n))$이다.

❸ $\frac{f(n)}{h(n)}=\Theta(1)$이면 $T(n)=\Theta(h(n)\log n)$이다(마스터 정리의 3항 그대로임).

마스터 정리의 적용 예

예제 3-4

$T(n)=2T(\frac{n}{3})+c$ (c는 상수)

▶▶ **점근적 복잡도**

$a=2,\ b=3,\ h(n)=n^{\log_3 2},\ f(n)=c$

$\lim_{n\to\infty}\frac{f(n)}{h(n)}=\lim_{n\to\infty}\frac{c}{n^{\log_3 2}}=0$이므로 마스터 정리의 유형 ❶에 속한다($h(n)$이 $f(n)$을 다항식 $n^{\log_3 2}$의 비율로 압도한다). 따라서 $T(n)=\Theta(n^{\log_3 2})$이다.

마스터 정리의 적용 예

예제 3-5

$$T(n) = 2T(\frac{n}{4}) + n$$

▶▶ **점근적 복잡도**

$a=2,\ b=4,\ h(n)=n^{\log_4 2}=\sqrt{n},\ f(n)=n$

$\lim_{n\to\infty}\frac{f(n)}{h(n)}=\lim_{n\to\infty}\frac{n}{\sqrt{n}}=\infty$이고 $af(\frac{n}{b})=2\cdot\frac{n}{4}=\frac{n}{2}<f(n)$이므로 마스터 정리의 유형 ❷에 속한다. ($f(n)$이 $h(n)$을 다항식 $n^{\frac{1}{2}}$의 비율로 압도한다. 또 $af(\frac{n}{b})=2\cdot\frac{n}{4}=\frac{n}{2}\leq\frac{1}{2}f(n)$이므로 $c=\frac{1}{2}$에 대해 $af(\frac{n}{b})\leq cf(n)$을 만족한다.) 따라서 $T(n)=\Theta(n)$이다.

마스터 정리의 적용 예

예제 3-6

$$T(n)=2T(\frac{n}{2})+n$$

▶▶ **점근적 복잡도**

$a=b=2,\ h(n)=n^{\log_2 2}=n,\ f(n)=n$

$\frac{f(n)}{h(n)}=\Theta(1)$이므로 마스터 정리의 유형 ❸에 속한다. 따라서 $T(n)=\Theta(n\log n)$이다.