03장. 점화식과 점근적 복잡도 분석

Youn-Hee Han
LINK@KOREATECH
http://link.koreatech.ac.kr

사실을 많이 아는 것보다는
이론적 틀이 중요하고,
기억력보다는
생각하는 법이 더 중요하다.

-제임스 왓슨

학습 목표

◈ 재귀 알고리즘과 점화식의 관계를 이해한다.

◈ 점화식의 점근적 분석을 이해한다.

01. 점화식

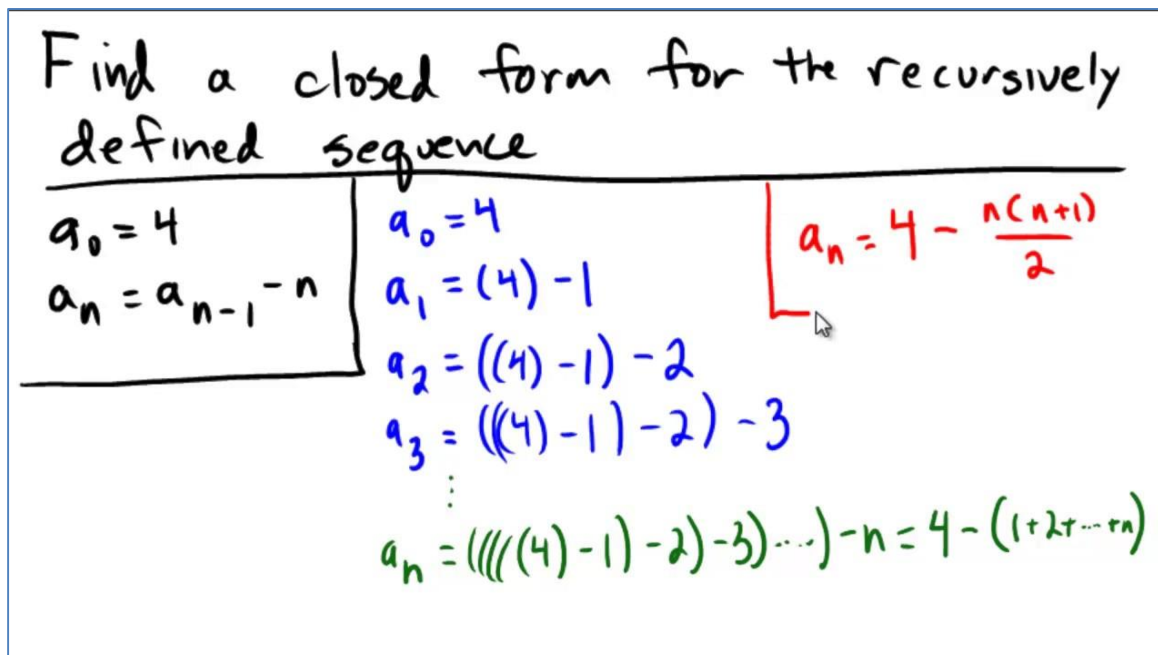
점화식의 이해

◈ 점화식(재귀식, recurrent relation)

- 수열에서 이웃하는 두개의 항 사이에 성립하는 관계를 나타낸 식
- 반대 개념의 식: 닫힌 식(Closed Form)

◈ 예

- $f(n) = f(n-1) + f(n-2)$
- $f(n) = nf(n-1)$
- $f(n) = f\left(\frac{n}{2}\right) + n$
- $a_n = a_{n-1} + 2$



병합 정렬의 수행 시간

◈ 병합정렬 의사코드

– 수행 시간의 점화식

$$T(n) = 2T\left(\frac{n}{2}\right) + \text{오버헤드}$$

- 크기가 n인 병합 정렬 시간은 크기가 $\frac{n}{2}$인 병합 정렬을 2번 수행하는 시간과 나머지 오버헤드를 더한 시간

알고리즘 2-1 | **병합 정렬**

```
mergeSort(A[], p, r)        ▷ A[p … r]을 정렬한다.
{
    if (p < r) then {
        ❶ q ← ⌊(p+r)/2⌋;           ▷ 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.
}
```

02. 점화식의 점근적 분석 방법

점화식의 점근적 분석 방법

◈ 반복 대치

– 더 작은 문제에 대한 함수로 반복해서 대치해 나가는 해법

◈ 추정 후 증명

– 결론을 추정하고 수학적 귀납법을 이용하여 증명하는 방법

◈ 마스터 정리

– 특정한 모양을 가진 점화식(재귀식)의 복잡도를 곧바로 산출

반복 대치

```
factorial(n)
{
        if (n=1) return 1 ;
        return n*factorial(n-1) ;
}
```

◈ 반복 대치 예 1: $n!$

- $T(n)$: 오른쪽 알고리즘으로 $n!$을 구하는 데 소요되는 시간
- $T(n)$의 점화식

$$T(n) = T(n-1) + c$$
$$T(1) \le c$$

  - c: 재귀 호출을 제외한 나머지 수행시간
- $T(n)$ 전개(반복 대치)를 통한 Closed Form 및 점근적 복잡도 산출

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

따라서, $\boldsymbol{T(n) = O(n)}$

반복 대치

◈ 반복 대치 예 2: 병합정렬

– $T(n)$: 입력의 크기가 n일 때 복잡도
  - 비교 연산에 대한 연산 수행 횟수

– $T(n)$의 점화식
  - $n = 2^k$이라 가정해도 일반성을 잃지 않음

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$
$$T(1) = 1$$

알고리즘 2–1 | 병합 정렬

```
mergeSort(A[], p, r)        ▷ A[p … r]을 정렬한다.
{
    if (p < r) then {
        ❶ q ← ⌊(p+r)/2⌋;           ▷ 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
    }
}
```

$T(n) = 1$

```
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.
}
```

merge 함수의 복잡도: $T(n) = n-1$

반복 대치

◈ 반복 대치 예 2: 병합정렬

– $T(n)$ 전개(반복 대치)를 통한 Closed Form 및 점근적 복잡도 산출

- $n = 2^k$이라 가정 ($k = 1, 2, 3, \ldots$)
  → $k = logn$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$
$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + n \\ &= 2\left(2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n = 2^2T\left(\frac{n}{2^2}\right) + 2n \\ &= 2^2\left(2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + n = 2^3T\left(\frac{n}{2^3}\right) + 3n \\ &\ldots \\ &= 2^kT\left(\frac{n}{2^k}\right) + kn = nT(1) + nlogn = n + nlogn \end{aligned}$$

$n + nlogn = O(nlogn)$ → 따라서, $\boldsymbol{T(n) = O(nlogn)}$

반복 대치

◈ 반복 대치를 위한 중요 가정: $n = 2^k$

– $n = 2^k$ 이라 가정해도 $T(n)$의 점근적 복잡도 산출에 대하여 일반성을 잃지 않는 이유
  - 어떠한 n이라도 n과 $2n$ 사이에 2의 멱수(2^k)가 존재함
    - ➢ 즉, $n \le 2^k < 2n$인 2^k가 존재
  - 만약 임의의 상수 r에 대해 $\boldsymbol{T(n) = O(n^r)}$이라고 가정
  - 이때, 입력의 크기가 2배가 되면

$$\boldsymbol{T(2n) = O\big((2n)^r\big) = O(2^r n^r) = O(n^r)}$$

  이므로 입력의 크기가 2배가 되도 점근적 복잡도는 동일
  - 입력의 크기 n에 대하여 $\boldsymbol{T(n)}$은 다음 식을 만족하므로 단조 증가 함수

$$\boldsymbol{T(n)(= O(n^r)) \le T\big(2^k\big) < T(2n)(= O(n^r))}$$

  - 따라서, $T\big(2^k\big)$의 점근적 복잡도도 $\boldsymbol{O(n^r)}$
  - 따라서, $n = 2^k$이라 가정해도 점근적 복잡도 산출 결과는 일반적인 n에 대한 점근적 복잡도 산출 결과와 동일

추정 후 증명

◈ 추정 후 증명 예 1: 병합정렬

- [예제 3-1] 입력의 크기가 n이고 복잡도가 $T(n) \le 2T\left(\frac{n}{2}\right) + n$ 일 때, 점근적 복잡도는 $T(n) = O(nlogn)$이다. 즉, 충분히 큰 n에 대하여 $T(n) \le cnlogn$을 만족하는 양의 실수 c가 존재한다.

  [증명]

  - 경계 조건($n \ge 2$(즉, $n_0 = 2$)): $T(2) \le c_1 2log2$인 양의 실수 c_1은 분명히 존재한다.
  - 귀납적 가정: $T\left(\frac{n}{2}\right) \le c_2 \frac{n}{2} log \frac{n}{2}$인 양의 실수 c_2가 존재한다.
  - 귀납적 전개:

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + n \\ &\le 2c_2 \frac{n}{2} log \frac{n}{2} + n \\ &= c_2 nlogn - c_2 nlog2 + n \\ &= c_2 nlogn + (-c_2 log2 + 1)n \\ &\le c_2 nlogn \text{ for a positive number } c_2 \ge \frac{1}{log2} \end{aligned}$$

  즉, 임의의 n에 대해 $T(n) \le c_2 nlogn$인 양의 실수 c_2가 존재한다.
  따라서, 최종적인 c를 $c = \max\{c_1, c_2\}$로 정하면 $T(n) = O(nlogn)$임이 증명된다.

추정 후 증명

◈ 추정 후 증명 예 2

– [예제 3-2] 입력의 크기가 n일 때 복잡도가 $T(n) \le 2T\left(\frac{n}{2}+10\right)+n$인 점화 관계식을 가질 때, 점근적 복잡도는 $T(n) = O(nlogn)$이다. 즉, 충분히 큰 n에 대하여 $T(n) \le cnlogn$을 만족하는 양의 실수 c가 존재한다.

[증명]

- $n \ge 40$[$(즉, n_0 = 40)$]으로 가정하며, 점근적 복잡도 분석에서 이러한 가정은 문제없다. 이 때, 경계 조건을 위해서는 $\frac{n}{2}+10$ 식을 고려하여 $T(30)$에 대해 검토한다.

- 경계 조건: $T(30) \le 30c_1log30$인 양의 실수 c_1는 분명히 존재한다.

- 귀납적 가정: $T\left(\frac{n}{2}+10\right) \le c_2\left(\frac{n}{2}+10\right)log\left(\frac{n}{2}+10\right)$인 양의 실수 c_2가 존재한다.

추정 후 증명

◈ 추정 후 증명 예 2

- [예제 3-2] 입력의 크기가 n일 때 복잡도가 $T(n) \le 2T\left(\frac{n}{2}+10\right)+n$인 점화 관계식을 가질 때, 점근적 복잡도는 $T(n)=O(nlogn)$이다. 즉, 충분히 큰 n에 대하여 $T(n) \le cnlogn$을 만족하는 양의 실수 c가 존재한다.
  - [증명]
    - 귀납적 전개:

$$
\begin{aligned}
T(n) &\le 2T\left(\frac{n}{2}+10\right)+n \\
&\le 2c_2\left(\frac{n}{2}+10\right)log\left(\frac{n}{2}+10\right)+n \\
&= c_2nlog\left(\frac{n}{2}+10\right)+20c_2log\left(\frac{n}{2}+10\right)+n \\
&\le c_2nlog\left(\frac{3n}{4}\right)+20c_2log\left(\frac{3n}{4}\right) \qquad (\because n \ge 40) \\
&= c_2nlogn+[c_2(log3-log4)+1]n+20c_2log\left(\frac{3n}{4}\right) \\
&\le c_2nlogn \text{ for a large positive number } c_2
\end{aligned}
$$

즉, 임의의 n에 대해 $T(n) \le c_2nlogn$인 양의 실수 c_2가 존재한다. 따라서, 최종적인 c를 $c=\max\{c_1,c_2\}$로 정하면 $T(n)=O(nlogn)$임이 증명된다.

- [예제 3-3] 생략

마스터 정리

◈ 마스터 정리

– 복잡도가 다음 형태의 재귀식으로 표현될 때, 점근적 복잡도를 곧 바로 산출할 수 있는 정리

$$\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n) \quad \text{for } a \geq 1 \text{ and } b > 1}$$

- 즉, 입력의 크기가 n인 문제를 풀기 위해, 입력의 크기가 $\frac{n}{b}$인 문제를 a개 풀고, 나머지 $f(n)$의 오버헤드가 필요한 알고리즘에 대한 복잡도

– [정리 3-1]

- 추가적인 함수 $h(n)$ 정의 $\boxed{h(n) = n^{\log_b a}}$
- 1) 어떤 양의 상수 ε에 대하여 $\frac{f(n)}{h(n)} = O\left(\frac{1}{n^{\varepsilon}}\right)$이면 $T(n) = \Theta(h(n))$이다.
- 2) 어떤 양의 상수 ε에 대하여 $\frac{f(n)}{h(n)} = \Omega(n^{\varepsilon})$이고, 어떤 상수 c와 충분히 큰 모든 n에 대해 $af\left(\frac{n}{b}\right) \leq cf(n)$이면 $T(n) = \Theta(f(n))$이다.
- 3) $\frac{f(n)}{h(n)} = \Theta(1)$이면 $T(n) = \Theta(h(n)logn)$이다.

마스터 정리

◈ 마스터 정리

- 복잡도가 다음 형태의 재귀식으로 표현될 때, 점근적 복잡도를 곧 바로 산출할 수 있는 정리

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \quad \text{for } a \geq 1 \text{ and } b > 1$$

  - 즉, 입력의 크기가 n인 문제를 풀기 위해, 입력의 크기가 $\frac{n}{b}$인 문제를 a개 풀고, 나머지 $f(n)$의 오버헤드가 필요한 알고리즘에 대한 복잡도

- [마스터 정리의 근사 버전]
  - 추가적인 함수 $h(n)$ 정의: $h(n) = n^{\log_b a}$
  - 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 $af\left(\frac{n}{b}\right) \leq f(n)$이면 $T(n) = \Theta(f(n))$이다.
  - 3) $\frac{f(n)}{h(n)} = \Theta(1)$이면 $T(n) = \Theta(h(n)logn)$이다.

마스터 정리

◈ 마스터 정리 활용 예 1

– [예제 3-4] $T(n) = 2T\left(\frac{n}{3}\right) + c$ [c는 상수]

$\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n) \text{ for } a \geq 1 \text{ and } b > 1}$ $\boxed{h(n) = n^{\log_b a}}$

- $a = 2,\ b = 3,\ f(n) = c$

- [마스터 정리의 근사 버전]에 의해
  - ➢ $h(n) = n^{\log_b a} = n^{\log_3 2}$
  - ➢ $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{c}{n^{\log_3 2}} = 0$ 이므로 $T(n) = \Theta(h(n)) = \Theta(n^{\log_3 2})$이다.

마스터 정리

◈ 마스터 정리 활용 예 2

– [예제 3-5] $T(n) = 2T\left(\frac{n}{4}\right) + n$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ for $a \geq 1$ and $b > 1$

$h(n) = n^{\log_b a}$

- $a = 2$, $b = 4$, $f(n) = n$

- [마스터 정리의 근사 버전]에 의해
  - $h(n) = n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$
  - $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{n}{\sqrt{n}} = \infty$ 이고, 충분히 큰 모든 n에 대해 $2f\left(\frac{n}{4}\right) = 2 \cdot \frac{n}{4} = \frac{n}{2} \leq f(n)$이므로, $T(n) = \Theta(f(n)) = \Theta(n)$이다.

마스터 정리

◈ 마스터 정리 활용 예 3

– [예제 3-6] $T(n) = 2T\left(\frac{n}{2}\right) + n$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ for $a \geq 1$ and $b > 1$

$h(n) = n^{\log_b a}$

- $a = 2$, $b = 2$, $f(n) = n$
- [마스터 정리의 근사 버전]에 의해
  - ➢ $h(n) = n^{\log_b a} = n^{\log_2 2} = n$
  - ➢ $\frac{f(n)}{h(n)} = \Theta(1)$이므로, $T(n) = \Theta(h(n)logn) = \Theta(nlogn)$이다.

마스터 정리

◈ 마스터 정리 활용 예 4

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ for $a \geq 1$ and $b > 1$

$h(n) = n^{\log_b a}$

- $a = 1$, $b = 2$, $f(n) = 1$
- [마스터 정리의 근사 버전]에 의해
  - ➢ $h(n) = n^{\log_b a} = n^{\log_2 1} = n^0 = 1$
  - ➢ $\frac{f(n)}{h(n)} = \Theta(1)$이므로, $T(n) = \Theta(h(n)logn) = \Theta(logn)$이다.

마스터 정리

◈ 마스터 정리 활용 예 5

$$T(n) = 3T\left(\frac{n}{2}\right) + n$$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ for $a \geq 1$ and $b > 1$

$h(n) = n^{\log_b a}$

- $a = 3$, $b = 2$, $f(n) = n$

- [마스터 정리의 근사 버전]에 의해
  - $h(n) = n^{\log_b a} = n^{\log_2 3}$
  - $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{n}{n^{\log_2 3}} = 0$ 이므로 $T(n) = \Theta\big(h(n)\big) = \Theta\big(n^{\log_2 3}\big) \approx \Theta(n^{1.58})$이다.

마스터 정리

◈ 마스터 정리 활용 예 6

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ **for** $a \geq 1$ **and** $b > 1$

$h(n) = n^{\log_b a}$

- $a = 7, b = 2, f(n) = n^2$
- **[마스터 정리의 근사 버전]에 의해**
  - $h(n) = n^{\log_b a} = n^{\log_2 7}$
  - $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{n^2}{n^{\log_2 7}} = \lim_{n\to\infty} \frac{n^{\log_2 4}}{n^{\log_2 7}} = 0$ **이므로** $T(n) = \Theta(h(n)) = \Theta\left(n^{\log_2 7}\right) \approx \Theta(n^{2.81})$**이다.**

Questions & Answers