2장. 점화식과 점근적 복잡도 분석

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

학습목표

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

점화식의 이해

- 점화식
  - 어떤 함수를 자신보다 더 작은 변수에 대한 함수와의 관계로 표현한 것
- 예
  - $a_n = a_{n-1} + 2$
  - $f(n) = n\,f(n-1)$
  - $f(n) = f(n-1) + f(n-2)$
  - $f(n) = f(\frac{n}{2}) + n$

Mergesort의 수행시간

```
mergeSort(A[ ], p, r)
{
    if (p < r) then { ---------------------------- ①
        q ← (p+q)/2;  ---------------------- ②  ▷ p, q의 중간 지점 계산
        mergeSort(A, p, q);  --------------- ③  ▷ 전반부 정렬
        mergeSort(A, q+1, r); -------------- ④  ▷ 후반부 정렬
        merge(A, p, q, r);  ---------------- ⑤  ▷ 병합
    }
}
merge(A[ ], p, q, r)
{
    정렬되어 있는 두 배열 A[p ... q]와 A[q+1 ... r]을 합하여
    정렬된 하나의 배열 A[p ... r]을 만든다.
}
```

수행시간의 점화식: $T(n) = 2T(n/2) +$ 오버헤드

✓크기가 n인 병합정렬 시간은 크기가 $n/2$인 병합정렬을 2번 하고 나머지 오버헤드를 더한 시간이다

점화식의 점근적 분석 방법

- 반복대치
  - 더 작은 문제에 대한 함수로 반복해서 대치해 나가는 해법
- 추정 후 증명
  - 결론을 추정하고 수학적 귀납법으로 이용하여 증명하는 방법
- 마스터 정리
  - 형식에 맞는 점화식의 복잡도를 바로 알 수 있다

반복대치

- 예) $n!$을 구하는 알고리즘의 복잡도

```
factorial(n)
{
    if (n == 1) return 1;  ---------- ①
    return n * factorial(n – 1);  ------- ②
}
```

$T(n) = T(n-1) + c$ // c = ① 수행 시간 + ②의 곱셈 한 번 수행 시간
$T(1) \leq c$

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

따라서 $T(n) = O(n)$

반복대치

- 예) Mergesort의 경우

$T(n) = 2T(\frac{n}{2}) + n$ // 여기서 n은 ⑤에 필요한 n-1과 // ①에 필요한 한 번의 비교를 합한 것

$T(1) = 1$

$$
\begin{aligned}
T(n) &= 2T(\tfrac{n}{2}) + n \\
&= 2(2T(\tfrac{n}{2^2}) + \tfrac{n}{2}) + n = 2^2T(\tfrac{n}{2^2}) + 2n \\
&= 2^2(2T(\tfrac{n}{2^3}) + \tfrac{n}{2^2}) + 2n = 2^3T(\tfrac{n}{2^3}) + 3n \\
&\cdots \\
&= 2^kT(\tfrac{n}{2^k}) + kn = nT(1) + kn \\
&= n + n\log n \qquad // T(1) = 1\text{이고 } k = \log_2 n \\
&= O(n\log n)
\end{aligned}
$$

추정 후 증명(Guess & Verification)

식의 모양을 보고 점근적 복잡도를 추정한 다음 그것이 옳음을 귀납적으로 증명하는 방식

예1) Mergesort의 점화식 $T(n) \le 2T(\frac{n}{2}) + n$ 의 점근적 복잡도는 $T(n) = O(n\log n)$이다. 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 대하여 $T(\frac{n}{2}) \le c\frac{n}{2}\log\frac{n}{2}$ 을 만족한다고 가정하면,

$$
\begin{aligned}
T(n) &\le 2T(\tfrac{n}{2}) + n \\
&\le 2c(\tfrac{n}{2})\log(\tfrac{n}{2}) + n \qquad \text{// 귀납적 가정을 이용} \\
&= cn\log n - cn\log 2 + n \\
&= cn\log n + (-c\log 2 + 1)n \\
&\le cn\log n
\end{aligned}
$$

} // 이를 만족하는 $c \ge \frac{1}{bg2}$가 존재한다.

추정후 증명

예2) $T(n) = 2T(\frac{n}{2} + 10) + n$의 점근적 복잡도는 $O(n\log n)$이다.
즉, 충분히 큰 n에 대하여 $T(n) \le cn\log n$ 인 양의 상수 c가 존재

$$
\begin{aligned}
T(n) &\le 2T(\tfrac{n}{2} + 10) + n \\
&\le 2c(\tfrac{n}{2} + 10)\log(\tfrac{n}{2} + 10) + \mathrm{n} \quad \text{// 귀납적 가정 이용.} \\
&\qquad\qquad \text{// 단, } \tfrac{n}{2} + 10 < n \text{, 따라서 } n > 20 \\
&= cn\log(\tfrac{n}{2} + 10) + 20c\log(\tfrac{n}{2} + 10) + n \\
&\le cn\log\tfrac{3n}{4} + 20c\log\tfrac{3n}{4} + n \qquad \text{// } \log(\tfrac{n}{2} + 10) \le \log\tfrac{3n}{4} \text{ 을 이용,} \\
&\text{// 그러려면 } \tfrac{n}{2} + 10 \le \tfrac{3n}{4} \text{ 이어야 하므로 n} \ge 40 \text{ 이어야 함.} \\
&= cn\log n + cn(\log 3 - \log 4) + 20c\log\tfrac{3n}{4} + n \\
&= cn\log n + [c(\log 3 - \log 4) + 1]n + 20c\log\tfrac{3n}{4} \qquad \text{// 빨간색 처리된} \\
&\text{// 부분이 0이하 이어야 증명이 완료됨. } n\text{이 충분히 크므로 OK!} \\
&\le cn\log n
\end{aligned}
$$

추정후 증명

예3) $T(n) = 2T(\frac{n}{2}) + 1$의 점근적 복잡도는 $O(n)$이다.
즉 $T(n) \leq cn$

<증명 실패의 경우>

$T(n) = 2T(\frac{n}{2}) + 1$
$\leq 2c(\frac{n}{2}) + 1$ // 귀납적 가정 이용
$= cn + 1$

더 이상 진행 불가. $cn+1 \leq cn$임을 증명할 수 없기 때문.

<성공적인 증명>

발상의 전환 필요! $T(n) \leq cn$ 대신 $T(n) \leq cn - 2$ 임을 증명하자.
그래도 여전히 $T(n) = O(n)$ 이기 때문이다.

$T(n) = 2T(\frac{n}{2}) + 1$
$\leq 2(c\frac{n}{2} - 2) + 1$ // 귀납적 가정 이용
$= cn - 3$
$\leq cn - 2$

마스터 정리Master Theorem

- $T(n) = aT(\frac{n}{b}) + f(n)$와 같은 모양을 가진 점화식은 마스터 정리에 의해 바로 결과를 알 수 있다.
  - 입력크기 n인 문제를 풀기 위해 입력 크기 $\frac{n}{b}$인 문제를 a개 풀고, 나머지 $f(n)$의 오버헤드가 필요한 알고리즘들에 해당.
- $n^{\log_b a} = h(n)$이라 하자

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 $\frac{f(n)}{h(n)} = \Omega(\frac{1}{n^\varepsilon})$이고, 어떤 상수 $c(<1)$와 충분히 큰 모든 n에 대해 $af(\frac{n}{b}) \leq cf(n)$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 의미

① $h(n)$이 더 무거우면 $h(n)$이 수행시간을 결정한다.
② $f(n)$이 더 무거우면 $f(n)$이 수행시간을 결정한다.
③ $h(n)$과 $f(n)$이 같은 무게이면 $h(n)$에 $\log n$을 곱한 것이 수행시간이 된다.

마스터 정리의 적용 예

- $T(n) = 2T(\frac{n}{3}) + c$
  - 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 정리 ①에 해당.
    - $h(n)$이 $f(n)$을 다항식 $n^{\log_3 2}$의 비율로 압도한다.
  - 따라서 $T(n) = \Theta(n^{\log_3 2})$

마스터 정리의 적용 예

- $T(n) = 2T(\frac{n}{4}) + n$
  - a=2, b=4, $h(n) = n^{\log_4 2}$, $f(n) = n$
  - $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{n}{\sqrt{n}} = \infty$이고, $af(\frac{n}{b}) = 2\frac{n}{4} = \frac{n}{2} < f(n)$ 이므로 마스터 정리 ②에 해당.
    - $f(n)$이 $h(n)$을 다항식 $\sqrt{n}$의 비율로 압도한다. 또한 $af(\frac{n}{b}) = 2\frac{n}{4} = \frac{n}{2} \leq \frac{1}{2}f(n)$ 이므로 c = $\frac{1}{2}$에 대해 $af(\frac{n}{b}) \leq cf(n)$을 만족한다.
  - 따라서 $T(n) = \Theta(n)$

마스터 정리의 적용 예

- $T(n) = 2T(\frac{n}{2}) + n$
  - a=2, b=2, $h(n) = n^{\log_2 2} = n$, $f(n) = n$
  - $\frac{f(n)}{h(n)} = \Theta(1)$이므로 마스터 정리 유형 ③에 속한다.
  - 따라서 $T(n) = \Theta(n \log n)$

Thank you