

관계 중심의 사고법

쉽게 배우는 알고리즘

3장. 점화식과 알고리즘 복잡도 분석

Recurrence and Asymptotic Complexity Analysis

점화식

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

병합정렬의 수행시간

```

mergeSort(A[ ], p, r)
{
    if (p < r) then {
        q ← (p+q)/2; ----- ① ▷ p, q의 중간 지점 계산
        mergeSort(A, p, q); ----- ② ▷ 전반부 정렬
        mergeSort(A, q+1, r); ----- ③ ▷ 후반부 정렬
        merge(A, p, q, r); ----- ④ ▷ 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: $T(n) = 2T(n/2) + \text{오버헤드}$

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

점화식의 점근적 분석 방법

1. 반복대치 (Iteration)

- 더 작은 문제에 대한 함수로 반복해서 대치해 나가는 해법

2. 추정후 증명 (Guess & Verification)

- 결론을 추정하고 수학적 귀납법으로 이용하여 증명하는 방법

3. 마스터 정리 (Master Theorem)

- 형식에 맞는 점화식의 복잡도를 바로 알 수 있다

Assumption

1. For all $T(n)$, n is positive integers
2. All functions are monotonically nondecreasing
 - $T(n) \leq T(m) \forall n < m$
3. If we need, we can WLOG assume $n = a^k$ for any polynomial asymptotic function

1. 반복대치

$$T(n) = T(n-1) + n$$

$$T(1) = 1$$

$$T(n) = T(n-1) + n$$

$$= (T(n-2) + (n-1)) + n$$

$$= (T(n-3) + (n-2)) + (n-1) + n$$

...

$$= T(1) + 2 + 3 + \dots + n$$

$$= 1 + 2 + \dots + n$$

$$= n(n+1)/2$$

$$= \Theta(n^2)$$

반복대치

$$T(n) = 2T(n/2) + n$$

Assume $n = 2^k$

$$T(1) = 1$$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \\ &= 2^2(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n \\ &\dots \\ &= 2^kT(n/2^k) + kn \\ &= n + n \log n \\ &= \Theta(n \log n) \end{aligned}$$

다른 예: 반복대치

$$T(n) = n + 3T(\frac{n}{4})$$

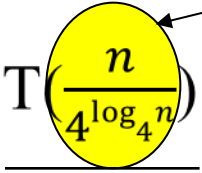
Assume $n = 4^k$

$$\begin{aligned} T(n) &= n + 3T(\frac{n}{4}) \\ &= n + 3(\frac{n}{4} + 3T(\frac{n}{4^2})) = n + \frac{3}{4}n + 3^2T(\frac{n}{4^2}) \\ &= n + \frac{3}{4}n + 3^2(\frac{n}{4^2} + 3T(\frac{n}{4^3})) = n + \frac{3}{4}n + (\frac{3}{4})^2n + 3^3T(\frac{n}{4^3}) \end{aligned}$$

$$\dots$$

$$= n + \frac{3}{4}n + (\frac{3}{4})^2n + \dots + 3^{\log_4 n}T(\frac{n}{4^{\log_4 n}})$$

$$\leq n \sum_{i=0}^{\infty} (\frac{3}{4})^i + n^{\log_4 3} \Theta(1)$$



$$= 4n + o(n)$$

$$= \Theta(n)$$

여기서 $o(n)$ 은 집합이 아니고
 $o(n)$ 에 속하는 어떤 함수를 의미함
이것은 집합

$$\therefore T(n) = O(n)$$

2. 추정후 증명

$$T(n) = 2T(n/2) + n$$

추정: $T(n) = O(n \log n)$, 즉 $T(n) \leq cn \log n$

<증명>

$$\begin{aligned}
 T(n) &= 2T(n/2) + n \\
 &\leq 2c(n/2)\log(n/2) + n && \text{귀납적 대치(inductive substitution)} \\
 &= cn \log n - cn \log 2 + n \\
 &= cn \log n + (-c \log 2 + 1)n \\
 &\leq cn \log n && \leftarrow \text{이를 만족하는 } c \text{가 존재한다} \\
 &&& \text{Choose e.g., } c = 2, n_0 = 2
 \end{aligned}$$

Reminder: $O(n \log n) = \{f(n) \mid \exists c > 0, n_0 \geq 0 \text{ s.t. } \forall n \geq n_0, f(n) \leq cn \log n\}$

추정후 증명: 다른 예

$$T(n) = 2T(\frac{n}{2} + 17) + n$$

추정: $T(n) = O(n \log n)$, 즉 $T(n) \leq cn \log n$

<증명>

$$\begin{aligned} T(n) &= 2T(\frac{n}{2} + 17) + n \\ &\leq 2c(\frac{n}{2} + 17)\log(\frac{n}{2} + 17) + n && \longleftarrow \frac{n}{2} + 17 < n, 34 < n \\ &= c(n+34)\log(\frac{n}{2} + 17) + n \\ &\leq c(n+34)\log\frac{3n}{4} + n && \longleftarrow \frac{n}{2} + 17 \leq \frac{3n}{4}, 68 \leq n \\ &= cn \log n + cn \log \frac{3}{4} + 34c \log \frac{3n}{4} + n \\ &= cn \log n + n(c \log \frac{3}{4} + 1) + 34c \log \frac{3n}{4} \\ &\leq cn \log n \text{ for sufficiently large } n \end{aligned}$$

≤ 0

< 0

Choose $c = 5$

조심! c 값의 일관성

앞에서

...

$$= c(n+34)\log\left(\frac{n}{2}+17\right) + n$$

$$\leq c(n+34)\log n + n$$

$$= cn\log n + 34c\log n + n$$

$$\leq dn\log n \text{ (X)} \quad \longleftarrow \text{새로운 상수 도입!}$$

추정후 증명: 직관과 배치되는 예

$$T(n) = 2T(n/2) + 1$$

추정: $T(n) = O(n)$, 즉 $T(n) \leq cn$

<증명>

$$T(n) = 2T(n/2) + 1$$

$$\leq \underline{2c(n/2)} + 1$$

← 귀납적 대치

$$= cn + 1$$

$$\underline{\leq cn}$$

← 더 이상 진행 불가!

1차이로 불가능

직관과 배치되지만...

추정: $T(n) \leq cn - 2$

← 왜 $T(n) \leq cn + 2$ 가 아니고?

<증명>

$$T(n) = 2T(n/2) + 1$$

$$\leq 2(\underline{c(n/2) - 2}) + 1 \quad \leftarrow \text{귀납적 대치}$$

$$= cn - 3$$

$$\leq cn - 2$$

직관에 따르면...

추정: $T(n) \leq cn+2$

<증명>

$$\begin{aligned}
 T(n) &= 2T(n/2) + 1 \\
 &\leq 2(\underline{c(n/2) + 2}) + 1 \quad \longleftarrow \text{귀납적 대치} \\
 &= cn + 5 \\
 &\not\leq cn + 2
 \end{aligned}$$

We can always verify any claim for a small enough n (e.g., $n = 2$)

e.g. $T(n) = 10T(n/10) + n, T(1) = 1$

$$\begin{array}{ll} \text{Guess } T(n) \leq cn \log n & \longleftarrow O() \\ \geq cn \log n & \longleftarrow \Omega() \end{array}$$

$$\begin{aligned} \rightarrow T(10) &= 10T(1) + 10 = 20 \leq c10 \log 10 \\ &\geq c10 \log 10 \end{aligned}$$

Thus we don't have to explicitly prove the boundary cases
in Guess & Verification

3. 마스터 정리

- $T(n) = aT(\frac{n}{b}) + f(n)$ 와 같은 모양을 가진 점화식은 마스터 정리에 의해 바로 결과를 알 수 있다

배경

Given a recurrence,

$$T(1) = 1$$

$$T(n) = aT(\frac{n}{b}) + f(n) \text{ for } n > 1,$$

where

a, b are positive constants

$f(n) = O(g(n))$ for some polynomial $g(n)$

Assume $n = b^k$

$$\begin{aligned} T(n) &= f(n) + aT\left(\frac{n}{b}\right) \\ &= f(n) + a\left(f\left(\frac{n}{b}\right) + aT\left(\frac{n}{b^2}\right)\right) \\ &= f(n) + a\left(f\left(\frac{n}{b}\right) + a\left(f\left(\frac{n}{b^2}\right) + aT\left(\frac{n}{b^3}\right)\right)\right) \\ &\dots \\ &= \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right) + a^k T\left(\frac{n}{b^k}\right) \\ &= \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right) + n^{\log_b a} \end{aligned}$$

- Particular solution
- Cost of all overheads

- Homogeneous solution
- Cost of solving the boundary
subproblems of size 1
- Time complexity 분석의 잣대가 됨

마스터 정리

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

① 어떤 양의 상수 ε 에 대하여 $\frac{f(n)}{h(n)} = O(\frac{1}{n^\varepsilon})$ 이면

$$T(n) = \Theta(h(n))$$

② 어떤 양의 상수 ε 에 대하여 $\frac{f(n)}{h(n)} = \Omega(n^\varepsilon)$ 이고, 충분히 큰 모든 n 에 대해 $af(\frac{n}{b}) < f(n)$ 이면

$$T(n) = \Theta(f(n))$$

③ $\frac{f(n)}{h(n)} = \Theta(1)$ 이면

$$T(n) = \Theta(h(n) \log n)$$

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

- $T(n) = 2T(\frac{n}{3}) + c$
 - $a=2, b=3, h(n) = n^{\log_3 2}, f(n) = c$
 - $T(n) = \Theta(h(n)) = \Theta(n^{\log_3 2})$
- $T(n) = 2T(\frac{n}{4}) + n$
 - $a=2, b=4, h(n) = n^{\log_4 2}, f(n) = n$ and $2f(\frac{n}{4}) = \frac{n}{2} < n = f(n)$
 - $T(n) = \Theta(f(n)) = \Theta(n)$
- $T(n) = 2T(\frac{n}{2}) + n$
 - $a=2, b=2, h(n) = n^{\log_2 2} = n, f(n) = n$
 - $T(n) = \Theta(h(n) \log n) = \Theta(n \log n)$

Strassen Algorithm

- Want to multiply two $n \times n$ matrices $A \times B$

- $C = AB$, $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj} \rightarrow \Theta(n^3)$

- Divide the matrices into four $\frac{n}{2} \times \frac{n}{2}$ matrices

Then $C = AB$ can be rewritten as

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

$$\begin{aligned} \text{where } r &= ae + bf \\ s &= ag + bh \\ t &= ce + df \\ u &= cg + dh \end{aligned}$$

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) \text{ by master's theorem } T(n) = \Theta(n^3). \quad n^{\log_2 8} = n^3$$

- Strassen's algorithm

$$P_i = A_i B_i = (\alpha_{i1}a + \alpha_{i2}b + \alpha_{i3}c + \alpha_{i4}d)(\beta_{i1}e + \beta_{i2}f + \beta_{i3}g + \beta_{i4}h)$$

$$\begin{aligned} P_1 &= a(g-h) & r &= P_5 + P_4 - P_2 + P_6 \\ P_2 &= (a+b)h & s &= P_1 + P_2 \\ P_3 &= (c+d)e & t &= P_3 + P_4 \\ P_4 &= d(f-e) & u &= P_5 + P_1 - P_3 - P_7 \\ P_5 &= (a+d)(e+h) \\ P_6 &= (b-d)(f+h) \\ P_7 &= (a-c)(e+g) \end{aligned}$$

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

$$\text{by master theorem } T(n) = \Theta(n^{\log_2 7})$$