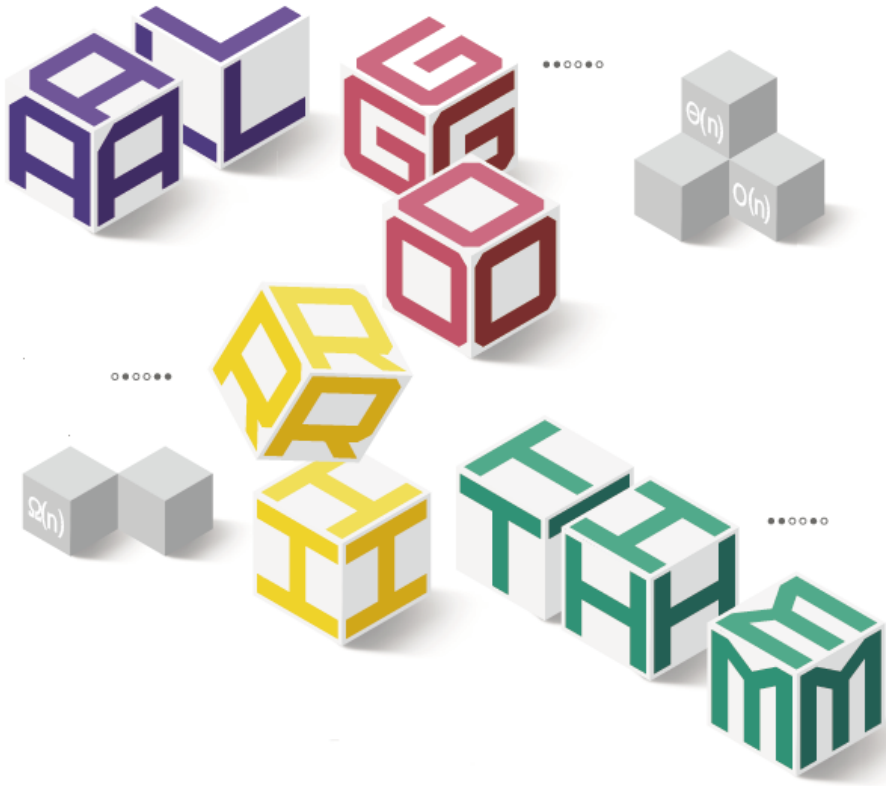


관계 중심의 사고법

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
  - $a_n = a_{n\text{-}1} + 2$
  - $f(n) = n\,f(n-1)$
  - $f(n) = f(n-1) + f(n-2)$
  - $f(n) = f(n/2) + n$ $\qquad = f\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$

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: $T(n) = 2T(n/2) +$ 오버헤드

✓크기가 n인 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

a는 양의 정수

1. 반복대치

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

$$\begin{aligned}
T(n) &= T(n-1) + n \\
&= (T(n-2) + (n-1)) + n \\
&= (T(n-3) + (n-2)) + (n-1) + n \\
&\cdots \\
&= T(1) + 2 + 3 + \ldots + n \\
&= 1 + 2 + \ldots + n \\
&= n(n+1)/2 \\
&= \Theta(n^2)
\end{aligned}$$

반복대치

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

Assume $n = 2^k$

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

다른 예: 반복대치

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

Assume $n = 4^k$

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

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

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

$$\cdots$$

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

1

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

$\Theta(1)$

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

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

$$= \Theta(n)$$

이것은 집합

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

2. 추정후 증명

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

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

<증명>

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

귀납적 대치(inductive substitution)

← 이를 만족하는 c가 존재한다

Choose e.g., $c = 2$, $n_0 = 4$(적당히)

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

추정후 증명: 다른 예

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

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

<증명>

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

≤ 0

< 0

Choose $c = 5$

조심! c 값의 일관성

앞에서

$\cdots$

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

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

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

$\not\leq dn\log n$ (X) ← 새로운 상수 도입!

추정후 증명: 직관과 배치되는 예

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

추정: $T(n) = O(n)$, 즉 $T(n) \le cn$

<증명>

$$\begin{aligned} T(n) &= 2T(n/2) + 1 \\ &\le 2c(n/2) + 1 \qquad \longleftarrow \text{ 귀납적 대치} \\ &= cn + 1 \\ &\not\le cn \qquad \longleftarrow \text{ 더 이상 진행 불가!} \end{aligned}$$

직관과 배치되지만...

추정: $T(n) \le cn-2$ ← 왜 $T(n) \le cn+2$ 가 아니고?

<증명>

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

직관에 따르면...

추정: $T(n) \le cn+2$

<증명>

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

Usually it is straightforward to verify a claim for boundary cases.

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

Guess $T(n) \le cn\log n$ ⟵ $O(\ )$

$\ge cn\log n$ ⟵ $\Omega(\ )$

$\rightarrow \mathrm{T}(10) = 10\mathrm{T}(1) + 10 = 20 \le c10\log 10$

$\ge c10\log 10$

The common practice don't 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,

$$T(1) = 1$$

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

where

a, b are positive constants

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

$$T(n) = f(n) + aT(\frac{n}{b})$$
$$= f(n) + a(f(\frac{n}{b}) + aT(\frac{n}{b^2}))$$
$$= f(n) + a(f(\frac{n}{b}) + a(f(\frac{n}{b^2}) + aT(\frac{n}{b^3}))$$
$$\cdots$$
$$= \sum_{i=0}^{k-1} a^i f(\frac{n}{b^i}) + a^k T(\frac{n}{b^k})$$
$$= \sum_{i=0}^{k-1} a^i f(\frac{n}{b^i}) + n^{\log_b a}$$

Assume $n = b^k$

1

$a^k = a^{\log_b n} = n^{\log_b a}$

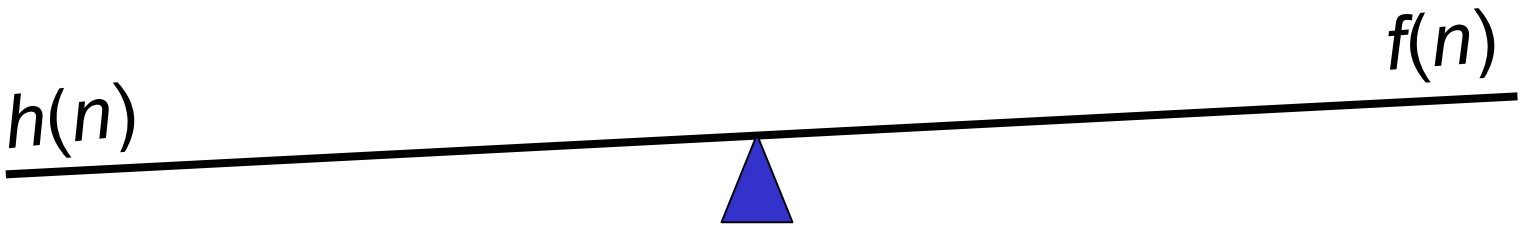
- Particular solution
- Cost of all overheads

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

마스터 정리의 직관적 의미

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

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 정리

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

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

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

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

$$\mathrm{T}(n) = \Theta(\mathrm{f}(n))$$

③ $\frac{f(n)}{h(n)} = \Theta(1)$이면

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

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 $2f\left(\frac{n}{4}\right) = \frac{n}{2} < n = f(n)$
  - $T(n) = \Theta(f(n)) = \Theta(n)$
- $T(n) = 2T(\frac{n}{2}) + n$
  - 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

Matrix multiplication

- 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

$$\left(\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right) = \left(\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right)\left(\begin{array}{c|c} e & g \\ \hline f & h \end{array}\right)$$

where
$$r = ae + bf$$
$$s = ag + bh$$
$$t = ce + df$$
$$u = cg + dh$$

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

○ Strassen's algorithm

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

$$P_1 = a(g-h) \qquad r = P_5 + P_4 - P_2 + P_6$$
$$P_2 = (a+b)h \qquad s = P_1 + P_2$$
$$P_3 = (c+d)e \qquad t = P_3 + P_4$$
$$P_4 = d(f-e) \qquad u = P_5 + P_1 - P_3 - P_7$$
$$P_5 = (a+d)(e+h)$$
$$P_6 = (b-d)(f+h)$$
$$P_7 = (a-c)(e+g)$$

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

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