알고리즘: 입력을 출력으로 바꿔주는 잘 만들어진 계산적 절차.

인스턴스: 입력되는 인풋들.

Correct Algorithm: 모든 input instance에 대해 Correct output을 내주는 Algorithm

삽입정렬

INSERTION-SORT(A)	times	T(n) best-case	T(n) worst-case
for j ← 2 to length[A]	n	〃	〃
do key ← A[j]	$n-1$	〃	〃
i ← j−1	$n-1$	〃	〃
while i>0 and A[i]>key	$\sum_{j=2}^{n} t_j$	$n-1$	$\frac{n(n+1)}{2}$
do A[i+1] ← A[i]	$\sum_{j=2}^{n} (t_j-1)$	0	$\frac{n(n-1)}{2}$
i ← i−1	$\sum_{j=2}^{n} (t_j-1)$	0	$\frac{n(n-1)}{2}$
A[i+1] ← key	$n-1$	〃	〃
		$O(n)$	$O(n^2)$

while의 최악의 경우: $1+2+3+\cdots+n = \frac{n(n+1)}{2}$

합병정렬

MERGE-SORT(A, p, r)

if $p < r$

then $q \leftarrow \lfloor (p+r)/2 \rfloor$ — devide $O(1)$

MERGE-SORT(A, p, q) — conquer $T(n/2)$

MERGE-SORT(A, q+1, r) — conquer $T(n/2)$

MERGE(A, p, q, r) — combine $O(n)$

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

Asymtotic Notation : 점근 표기법, 함수 증가 양상.

Asymtotic efficiency : 알고리즘의 성능 대변

O-Notation

정의) 모든 $n \geq n_0 > 0$에 대하여 $0 \leq f(n) \leq C \cdot g(n)$인 양의 상수 C와 n_0가 존재하면 $f(n) = O(g(n))$ 이다.

$g(n)$은 $f(n)$의 Asymtotic upper bound

- example

$f(n) = n^2 - 2n$

$0 \leq n^2 - 2n \leq c \cdot n^2$, $n \geq n_0 > 0$

→ C, n_0가 존재하는가?

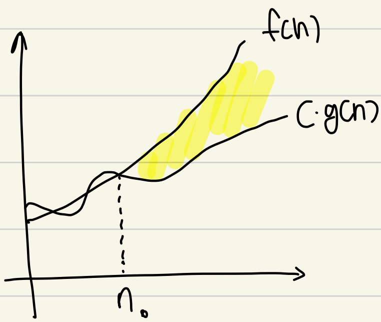
$C = 3$, $n_0 = 2$

$\therefore O(n^2)$

Ω - Notation

정의) 모든 $n \ge n_0 > 0$ 에 대하여 $0 \le c \cdot g(n) \le f(n)$ 인 양의상수 c와 n_0가 존재하면 $f(n) = \Omega(g(n))$ 이다.

$g(n)$은 $f(n)$의 Asymtotic lower bound



이미 Θ면 O도되고 Ω도 되는건가?

Θ - Notation

정의) 모든 $n \ge n_0 > 0$ 에 대하여 $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ 인 상수 c_1, c_2, n_0가 존재하면

$f(n) = \Theta(g(n))$ 이다.

recurrence

example) merge-sort $T(n) = \begin{cases} \Theta(1) & (n=1) \\ 2T(n/2) + \underbrace{\Theta(n)}_{\text{MERGE } T(n)} & (n>1) \end{cases}$

Substitution Method : 대체법

- 솔루션의 형태 guess
- 수학적 귀납법 사용.

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

guess : $T(n) = O(n\lg n) \leq cn\lg n$

prove by induction :

base $n_0 = 2$

$T(2) = 2T(1) + 2 \leq 2c\lg 2$

$n < K$ 라고 가정.

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

$T(K/2) \leq c\frac{K}{2}\lg\frac{K}{2}$

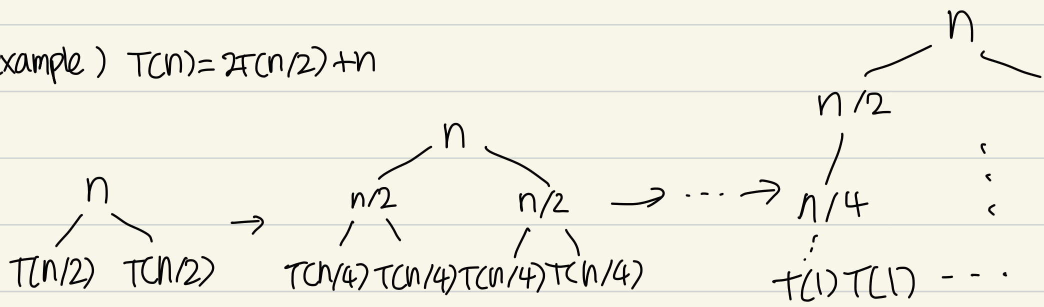
$$T(K) \leq cK\lg\frac{K}{2} + K$$
$$\leq cK(\lg K - 1) + K$$
$$\leq cK\lg K + K(1-c)$$

$c \geq 1$, $T(K) \leq cK\lg K$

Q.E.D.

Recursion-Tree Method : 재귀식 T(n)의 O-Notation을 합리적으로 추론하는 법.

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



$$T(n) = 2^0 \cdot n/2^0 + 2^1 \cdot n/2^1 + 2^2 \cdot n/2^2 + \cdots + \underbrace{\boxed{2^k}}_{\text{총노드수}} \cdot \underbrace{\boxed{n/2^k}}_{\text{노드값}}$$

$$= n \times \lg n$$

$$= O(n \lg n)$$

$n/2^k = 1$

$n = 2^k$

$k = \lg n$

example) $T(n) = 3T(n/4) + \Theta(n^2)$

1. Recursion-Tree Method

트리	노드 개수
n^2	1
$\vert$	$\vert$
$(n/4)^2$	3
$\vert$	$\vert$
$(n/4^2)^2$	3^2
$\vdots$	$\vdots$
$(n/4^k)^2 = 1$	$3^k = n^{\log_4 3}$

$k = \log_4 n$

$$
\begin{aligned}
T(n) &= 3^0 \cdot (n/4^0)^2 + 3^1 \cdot (n/4^1)^2 + \cdots + 3^{k-1} \cdot (n/4)^{k-1} + 3^k \cdot 1 \\
&= \sum_{i=0}^{k-1} 3^i \cdot (n/4^i)^2 + n^{\log_4 3} \\
&\le \sum_{i=0}^{\infty} 3^i \cdot n^2 / 4^{2i} + n^{\log_4 3} \\
&\le \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i \cdot n^2 + n^{\log_4 3} \\
&\le \frac{1}{1-\frac{3}{16}} \cdot n^2 + n^{\log_4 3}
\end{aligned}
$$

$T(n) = O(n^2)$

2. Substitution Method

ample:

$$T(n) = 3T(n/4) + n^2$$

guess:

$$T(n) = O(n^2) \leq cn^2$$

prove by induction:

Base $n_0 = 4$

$$T(4) = 3T(1) + 16 \leq 16c$$

$$3d + 1 \leq c$$

$n < k$라고 가정

$$T(k) = 3T(k/4) + k^2$$

$$T(k/4) \leq c \cdot \frac{k^2}{16}$$

$$T(k) \leq c \cdot \frac{3}{16} k^2 + k^2$$

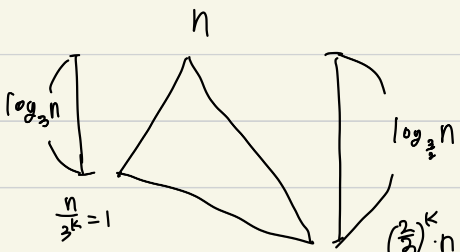
$$\leq \left(\frac{3}{16}c + 1\right) k^2$$

$\frac{3}{16}c + 1 > 0$, $T(k) \leq ck^2$ Q.E.D

맞나?

example) $T(n) = T(n/3) + T(2n/3) + \Theta(n)$

1. Recursion-Tree Method



$\frac{n}{3^k} = 1$ $\quad\quad$ $\left(\frac{2}{3}\right)^k \cdot n = 1$

$K = \log_3 n$ $\quad\quad$ $K = \log_{\frac{3}{2}} n$

각 레벨의 노드의 합 = n

$$n \cdot \log_3 n \le T(n) \le n \cdot \log_{\frac{3}{2}} n$$

$$n \cdot \frac{\lg n}{\lg 3} \le T(n) \le n \frac{\lg n}{\lg 3 - \lg 2}$$

$$\frac{1}{\lg 3} n \lg n \le T(n) \le \frac{1}{\lg 3 - 1} n \lg n$$

$$\therefore O(n \lg n)$$

2. Substitution Method

ample:

$$T(n) = T(n/3) + T(2n/3) + \theta(n)$$

guess:

$$T(n) = O(n\lg n) \le c \cdot n\lg n$$

prove by induction:

base $n_0 = 3$, $T(3) = T(1) + T(2) + 3 \le c \cdot 3\lg 3$

$n < k$ 일 때 $T(n) = O(n\lg n)$이 참이라고 가정.

$$T(k) = T(k/3) + T(2k/3) + k$$

$$T(k/3) \le c \cdot (k/3)\lg(k/3)$$

$$T(2k/3) \le c(2k/3)\lg(2k/3)$$

$$T(k) \le c \cdot (k/3)\lg(k/3) + c(2k/3)\lg(2k/3) + k$$

$$\le c\left[(k/3)\lg k - (k/3)\lg 3 + (2k/3)\lg k - (2k/3)\lg \tfrac{3}{2}\right] + k$$

$$\le c\left[k\lg k - (k/3)\lg 3 - (2k/3)\lg 3 + (2k/3)\right] + k$$

$$\le ck\lg k - \left[ck(\lg 3 - 2/3) - k\right]$$

$$c(\lg 3 - 2/3) - 1 \ge 0$$

$$c \ge \frac{1}{\lg 3 - 2/3}, \quad T(k) \le ck\lg k \quad \text{Q.E.D.}$$

힙 정렬

최대힙: 루트가 최대.

최소힙: 루트가 최소.

length[A] : A의 길이

heap-size[A]: 힙 데이터의 수 $\le$ length[A]

※ 완전 이진트리 노드 i에 대해서

PARENT(i): return $\lfloor i/2 \rfloor$

LEFT(i): return $2i$

RIGHT(i): return $2i+1$

n개의 노드를 가진 완전 이진트리 높이

$2^{h+1}-1=n$

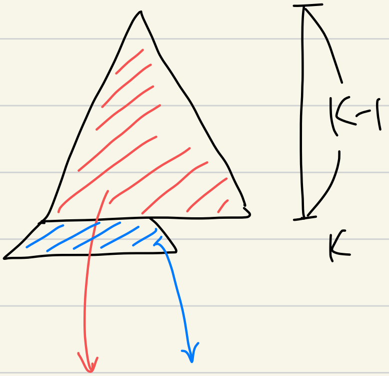
$h=O(\lg n)$

→ A에서 i노드를 힙구조에 맞게 재배치. 최대힙

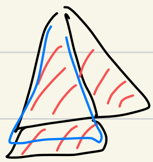
```
MAX-HEAPIFY(A, i)
    l ← LEFT(i)
    r ← RIGHT(i)
    if l ≤ heap-size[A] and A[l] > A[i]
        then largest ← l
        else largest ← i
    if r ≤ heap-size[A] and A[R] > A[i]
        then largest ← r
    if largest ≠ i
        then exchange A[i] ↔ A[largest]
             MAX-HEAPIFY(A, largest)
```

MAX-HEAPIFY의 시간복잡도

최악의 경우의수는 왼쪽 트리. 완전 이진트리는 왼쪽부터 채워지기 때문



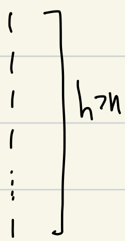
$$n = 2^K - 1 + 2^K/2 \fallingdotseq \frac{3}{2} 2^K$$



↳ 전체의 2/3.

$$\therefore T(n) = T(2n/3) + \Theta(1)$$

1. Recursion-Tree Method



$$2^{h+1} - 1 = n$$

$$h+1 = \lg n$$

$$h = O(\lg n)$$

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

2. Substitution method

ample:

$$T(n) = T(2n/3) + \Theta(1)$$

guess:

$$T(n) = O(\lg n) \le c \cdot \lg n$$

prove by induction:

base $n_0 = 3$, $T(3) = T(2) + 1 \le c \lg 3$

$\frac{d+1}{\lg 3} \le c$, d는 상수.

$n < K$ 라고 가정

$$T(K) = T(2K/3) + 1$$

$$T(2K/3) \le c \cdot \lg(2K/3) + 1$$

$$\le c(\lg K + 1 - \lg 3) + 1$$

$$\le c \lg K - c(\lg 3 - 1) + 1$$

$$c(\lg 3 - 1) \ge 1$$

$c \ge \frac{1}{\lg 3 - 1}$, $T(K) \le c \cdot \lg K$ Q.E.D.

→ A전체를 힙구조로 재배치

```
BUILD-MAX-HEAP(A)
    heap-size[A] ← length[A]
    for i ← ⌊length[A]/2⌋ downto 1
        do MAX-HEAPIFY(A, i)
```

BUILD-MAX-HEAP의 시간복잡도.

$height = \lfloor \lg n \rfloor$

→ 레벨별 최대 노드 갯수 : $\left\lceil \frac{n}{2^{h+1}} \right\rceil$

레벨	높이	노드갯수
0	h	$2^0 = \frac{2^{h+1}}{2^{h+1}} = \frac{n+1}{2^{h+1}}$
1	$h-1$	$2^1 = \frac{2^{h+1}}{2^h} = \frac{n+1}{2^h}$
2	$h-2$	$2^2 = \frac{2^{h+1}}{2^{h-1}} = \frac{n+1}{2^{h-1}}$
⋮	⋮	⋮
$h-1$	1	$2^{h-1} = \frac{2^{h+1}}{2^2} = \frac{n+1}{2^2}$
h	0	$2^h = \frac{2^{h+1}}{2} = \frac{n+1}{2}$

$$T(n) = \sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \cdot \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right)$$

$$= O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) = O\left(n \cdot \frac{\frac{1}{2}}{(1-\frac{1}{2})^2}\right)$$

$$= O(n)$$

$$2^{h+1} - 1 = n$$

$$2^{h+1} = n+1$$

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

```
HEAP-SORT(A)
    BUILD-MAX-HEAP(A)
    for i ← length[A] downto 2
        do exchange A[1] ↔ A[i]
            heap-size[A] = heap-size[A] - 1
            MAX-HEAPIFY(A, 1)
```

퀵 정렬

```
QUICKSORT(A, p, r)
    if p < r
        then q ← PARTITION(A, p, r)
             QUICKSORT(A, p, q-1)
             QUICKSORT(A, q+1, r)

PARTITION(A, p, r)
    x ← A[r]
    i ← p-1
    for j ← p to r-1
        do if A[j] ≤ x
            then i ← i+1
                exchange A[i] ↔ A[j]
    exchange A[i+1] ↔ A[r]
    return i+1
```

Worst-case : $T(n) = T(n-1) + n = O(n^2)$

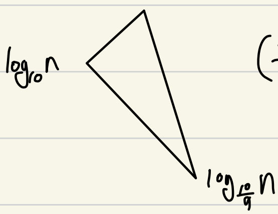
pivot 기준 한쪽으로 치우침　　파티션

Best-case : $T(n) = 2T(n/2) + n = O(n\lg n)$

Balanced-case : $T(n) = T(9n/10) + T(n/10) + n = O(n\lg n)$

$\frac{n}{10^k} = 1$

$n = 10^k$



$\left(\frac{9}{10}\right)^k n = 1$

$n = \left(\frac{10}{9}\right)^k$

$$n\log_{10} n \le T(n) \le n\log_{\frac{10}{9}} n$$

$$n\frac{\lg n}{\lg 10} \le T(n) \le n \cdot \frac{\lg n}{\lg 10 - \lg 9}$$

C_1　　C_1

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

Finding Median

모든 원소는 유니크하다 가정.

Input: Array A

Output: median

$\frac{n}{4} \le rank(x) \le \frac{3}{4}n$

A_L	x	A_R

$A_L < x < A_R$

몇번?

$|A_L|$이 $\lfloor \frac{n}{2} \rfloor$ 보다 작은 경우 → median은 A_R에 있음. $index = r - |A_L| - 1$

$|A_L|$이 $\lfloor \frac{n}{2} \rfloor$ 보다 큰 경우 → median은 A_L에 있음, $index = r$

FindElement(r, A)

$O(1)$ 1. A를 5개씩 그룹화.

$O(n)$ 2. 각 그룹의 median 들을 B에, Bsize= $\frac{n}{5}$

$T(\frac{n}{5})$ 3. FindElement($\frac{n}{10}$, A)

$O(n)$ 4. A를 A_L과 A_R로 파티셔닝

5. if $|A_L| == r-1$ then return x

6. else if $|A_L| > r-1$

$T(\frac{7}{10}n)$ 7. then FindElement(r, A_L)

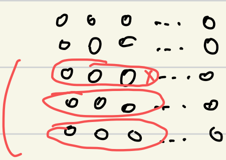
8. else

9 then FindElement($r-|A_L|-1$, A_R)

$$T(n) = T(\frac{n}{5}) + T(\frac{7}{10}n) + O(n)$$

$\frac{7}{10}n$인 이유

$3 \times \frac{1}{10}n$



최소 $\frac{7}{10}n$이 확실하기 때문에 최악의 경우 $O(\frac{7}{10}n)$

$$n$$

$$\frac{2}{10}n \qquad \frac{7}{10}n$$

$$\frac{4}{100}n \qquad \frac{14}{100}n \qquad \frac{14}{100}n \qquad \frac{49}{100}n$$

$\Rightarrow \frac{9^0}{10^0}$

$\vdots$

$\Rightarrow \frac{9}{10}n$

$\vdots$

$\Rightarrow \frac{81}{100}n$

$\vdots$

$\Rightarrow \frac{9^k}{10^k}n$

$$O\left(\sum_{h=0}^{\log n} \frac{9^h}{10^h} n \cdot h\right)$$

$$O\left(n \sum \left(\frac{9}{10}\right)^h \cdot h\right)$$

$$O\left(n \cdot \frac{\frac{9}{10}}{\left(1-\frac{9}{10}\right)^2}\right)$$

$$= O(n)$$

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

※ $T(n) = T(a) + T(b) + f(n)$

if $a+b<n \rightarrow T(n) = O(n)$

n보다 작은건가?
f(n)보다 작은건가?