1.1. $h(n) = n^{\log_4 4} = n$, $f(n) = b$ $\Rightarrow \frac{f(n)}{h(n)} = O(\frac{1}{n})$

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

1.2. $T(n) = 3T(n/3) + n\log n$ (Assume : $n = 3^k$)

$= 3(3T(n/3^2) + \frac{n}{3}\log\frac{n}{3}) + n\log n = 3^2T(n/3^2) + n\log(n \times \frac{n}{3})$

$= 3^2(3T(n/3^3) + \frac{n}{3^2}\log\frac{n}{3^2}) + n\log(n \times \frac{n}{3}) = 3^3T(n/3^3) + n\log(n \times \frac{n}{3} \times \frac{n}{3^2})$

$= 3^kT(n/3^k) + n\log\frac{n^k}{3^{k(k-1)/2}}$

$= nT(1) + n\log(n^k/n^{(k-1)/2})$

$= nT(1) + \frac{k+1}{2}\cdot n\log n = nT(1) + (\frac{\log_3 n + 1}{2})\cdot n\log n$ ← T(1)은 상수이고, 최고차항: $n(\log n)^2$

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

$\Rightarrow T(n) = \Omega(n(\log n)^2)$, $T(n) = O(n(\log n)^2)$

1.3. $h(n) = n^{\log_5 5} = n$, $f(n) = 3n$ $\Rightarrow \frac{f(n)}{h(n)} = O(1)$

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

1.4. Guess 1 : $T(n) = \Omega(n\log n)$, i.e., $T(n) \geq c_1 n\log n$

Proof 1 : $T(n) = T(n/4) + T(3n/4) + \Theta(n)$

$\geq c_1(\frac{n}{4}\log\frac{n}{4} + \frac{3n}{4}\log\frac{3n}{4}) + \Theta(n)$

$= c_1 n\log n + (\frac{c_1}{4}\log\frac{3^3}{4^4})n + \Theta(n)$

$\geq c_1 n\log n$ (for some $c_1 > 0$ s.t. $\Theta(n)$ dominates $\frac{c_1}{4}\log\frac{3^3}{4^4}$)

Guess 2 : $T(n) = O(n\log n)$, i.e., $T(n) \leq c_2 n\log n$

Proof 2 : $T(n) = T(n/4) + T(3n/4) + \Theta(n)$

$\leq c_2(\frac{n}{4}\log\frac{n}{4} + \frac{3n}{4}\log\frac{3n}{4}) + \Theta(n)$

$= c_2 n\log n + (\frac{c_2}{4}\log\frac{3^3}{4^4})n + \Theta(n)$

$\leq c_2 n\log n$ (for some $c_2 > 0$ s.t. $\frac{c_2}{4}\log\frac{3^3}{4^4}n$ dominates $\Theta(n)$)

$\Rightarrow T(n) = \Omega(n\log n)$, $T(n) = O(n\log n)$

1.5. Guess 1 : $T(n) = \Omega(n\log n)$, i.e., $T(n) \geq c_1 n\log 3n$

Proof 1 : $T(n) = 3T(n/3 + 9) + n$

$\geq 3c_1(\frac{n}{3}+9)\log(n+27) + n$

$= c_1(n+27)\log(n+27) + n$

$\geq c_1(n+27)\log n + n$

$= c_1 n\log n + n + 27c_1\log n$

$\geq c_1 n\log n$

Guess 2 : $T(n) = O(n\log n)$, i.e., $T(n) \leq c_2 n\log n$

Proof 2 : $T(n) = 3T(n/3 + 9) + n$

$\leq 3c_2(\frac{n}{3}+9)\log(\frac{n}{3}+9) + n$

$= c_2(n+27)\log(\frac{n}{3}+9) + n$

$\leq c_2(n+27)\log(\frac{n}{2}) + n$

$= c_2 n\log n - c_2 n\log 2 + 27c_2\log(\frac{n}{2}) + n$

$= c_2 n\log n + [n(1-c_2\log 2) + 27c_2\log(\frac{n}{2})]$

$\leq c_2 n\log n$ (for some $c_2 > 0$ s.t. $n(1-c_2\log 2)$ dominates $27c_2\log\frac{n}{2}$)

$\Rightarrow T(n) = \Omega(n\log n)$, $T(n) = O(n\log n)$

2. $T(n) = T(\lfloor \frac{n}{4} \rfloor) + T(n - 2\lfloor \frac{n}{4} \rfloor) + \theta(n)$

$\Rightarrow T(n) = T(n/4) + T(n/2) + \theta(n)$ (Assume : $r-p+1 = n = 4^k$)

$\leq \frac{3c_1}{4}n + \theta(n) = c_1 n + \theta(n) - \frac{c_1}{4}n$ (Guess : $T(n) = O(n)$, i.e., $T(n) \leq c_1 n$)

$\leq c_1 n$ (for some $c_1 > 0$ s.t. $\frac{c_1}{4}n$ dominates $\theta(n)$)

$T(n) \geq \frac{3c_2}{4}n + \theta(n) = c_2 n + \theta(n) - \frac{c_2}{4}n$ (Guess : $T(n) = \Omega(n)$, i.e., $T(n) \geq c_2 n$)

$\geq c_2 n$ (for some $c_2 > 0$ s.t. $\theta(n)$ dominates $-\frac{c_2}{4}n$)

$\Rightarrow T(n) = \theta(n)$

3. $T(n) = T(\frac{n}{3}+5) + T(\frac{2n}{3}+7)$

Guess : $T(n) = O(n)$, i.e., $T(n) \leq c(n-13)$

$\Rightarrow T(n) \leq c(\frac{n}{3}-8) + c(\frac{2n}{3}-6)$

$= c(n-14)$

$\leq c(n-13)$

$= O(n)$

4. 크기 n인 배열 a를 sorting할때, iteration 횟수를 i라 지정 (작은값을 앞으로 가져오는 정렬 사용)

① $i = 0$: no loof → true

② $i = k$: $a[0] \le a[1] \le \cdots \le a[k-1] \le a[x]$ $(k-1 < x < n)$

③ $i = k+1$: $(k+1)$루프에서 $a[k] \cdots a[n-1]$ 중 최솟값을 찾아 $a[k]$로 자리를 바꿔줌.

$a[k] \le a[x]$ $(k < x < n)$이고 k루프에서 $a[k-1] \le a[x]$ $(k-1 < x < n)$

이었기에 $a[0] \le a[1] \le \cdots \le a[k]$ 이 성립한다.

⇒ 따라서 Selection Sort 알고리즘은 제대로 Sorting 하고 있다.

5. quickSort(A[], p, r) :

```
if(p < r)
    q ← partition(A, p, r)
    quickSort(A, p, q-1)
    quickSort(A, q+1, r)
```

partition(A[], p, r) :

```
pivot = linearSelect(A, p, r, ⌊(r-p+2)/2⌋)
A[r] ⟷ A[pivot]
x ← A[r]
i ← p-1
for j ← p to r-1
    if (A[j] < x)
        A[++i] ⟷ A[j]
A[i+1] ⟷ A[r]
return i+1
```

linearSelect(A, p, r, i):

1. If #elements ≤ 5, find i^{th} smallest naively and finish
2. Divide the whole set into $\lceil n/5 \rceil$ groups each having 5 elements
   (If #elements is not an exact multiple of 5, one group has fewer than 5 elements)
3. Find median in each group (3^{rd} if #elements is 5)
   Let these medians be $m_1, m_2, \cdots, m_{\lceil n/5 \rceil}$
4. Find the median M of medians $m_1, m_2, \cdots, m_{\lceil n/5 \rceil}$, recursively.
   (If #elements is even, choose any of the two medians) ← call linearSelect()
5. Using M as the pivot, partition the set $A[p \dots r]$
6. Choose the appropriate part and recursively repeat steps 1 ~ 6 ← call linearSelect()

⇒ linearSelect의 worst-case running time : $T_1(n) \le T(\lceil n/5 \rceil) + T(7n/10 + 2) + \Theta(n)$

$$\Rightarrow T_1(n) \le T\left(\frac{n}{5}+1\right) + T\left(\frac{7n}{10}+2\right) + \Theta(n)$$

$$\le c_1\left(\frac{n}{5}+1\right) + c\left(\frac{7n}{10}+2\right) + \Theta(n) \quad (\text{Assume } T_1(k) \le ck \ \forall k,\ n_0 \le k < n)$$

$$= cn - \frac{cn}{10} + 3c + \Theta(n)$$

$$\le cn \quad \left(c > 0 \text{ s.t. } -\frac{cn}{10} \text{ dominates } 3c + \Theta(n)\right)$$

$$= O(n)$$

⇒ quickSort의 worst-case running time : $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

$$\Rightarrow \frac{\Theta(n)}{n} = \Theta(1)$$

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

6. mergeSort(A[], p, r):

```
if (p<r)
    q ← ⌊(p+r-1)/16⌋
    for i = 0 to 14
        mergeSort(A, p, q)
        tmp = p
        p = q+1
        q = 2q - tmp + 1
    mergeSort(A, p, r)
```

merge(A[], p, q, r):

16개의 sorted subarrays의 첫 원소들 중 최소 원소를 찾아서 앞에서 부터 배치한다.

$\Rightarrow T(n) = 16T(n/16) + \theta(n) \quad \leftarrow \frac{\theta(n)}{n} = \theta(1)$

$= \theta(n\log n)$

7.1. $T(n) = 1+2+2^2+\cdots+2^{k-1}$ (Assume: $n=2^k$)

$= 2^k - 1 = n-1$

$= \theta(n)$

7.2. $T(n) = \sum_{i=0}^{k-1}\left(\frac{n}{2^i}\times 2^i\right)$ (Assume: $n=2^k$)

$= nk = n\log_2 n$

$= \theta(n\log n)$