20190650 김관호 HW 1

1.(a) Let $f(n) = 10n^3 + 1024n^2 \log_2^{10} n + 3n \log_{10} n - 144$, $g(n) = n^3$

By definition of Big-O, when $c = 100$, $n_0 = 2^{10}$

$$10n_0^3 + 1024 n_0^2 \log_2^{10} n_0 + 3n_0 \log_{10} n^0 - 144 \le 100 \cdot n_0^3$$

thus, there exist constants $c > 0$, $n_0 \ge 0$ for all $n \ge n_0$ we have $f(n) \le c \cdot g(n)$

so, $10n^3 + 1024n^2 \log_2^{10} n + 3n \log_{10} n - 144 = O(n^3)$

1.(b)

To see if $n \log n = o(n^{1+c})$,

Let's see where does $\lim_{n\to\infty} \frac{n \log n}{n^{1+c}}$ goes.

If that goes to zero, $n \log n = o(n^{1+c})$

$$\lim_{n\to\infty} \frac{n \log n}{n^{1+c}} = \lim_{n\to\infty} \frac{\log n}{n^c}.$$

Since we know the commonsense rules In Lecture NOTE that any polynomial dominates any logarithm, even for tiny number $c > 0$ n^c dominates $\log n$

So, result of $\lim_{n\to\infty} \frac{\log n}{n^c}$ is zero.

$\therefore$ $n \log n = o(n^{1+c})$ for $c > 0$

2.

By using commonsense rules in <u>Lecture Note</u>, we can simplify any $t(n)$ easy to compare their asymptotic growths.

Let simplify each $t(n)$.

$t_1(n) = \frac{n^2}{\log_2^9 n} \Rightarrow n^2$, $t_2(n) = 2^{\log_2 n} \Rightarrow n$

$t_3(n) = \frac{n!}{n^{1024}} \Rightarrow n!$ (factorial dominates exponential)

$t_4(n) = 1024 n \log_{10} n \Rightarrow n \log n$

$t_5(n) = n^{1.5} \log_2 n^{1024} \Rightarrow n^{1.5} \log n$

$t_6(n) = 2^n$, $t_7(n) = 10^{1.9 \log_{10} n} \Rightarrow n^{1.9}$

(In $t_2(n)$, $t_7(n)$ $x^{\log_x n}$ can be $n^{\log_x x} = n$)

So, By comparing all of these let's rank ascending order.

$(n, n\log n, n^{1.5}\log n, n^{1.9}, n^2, 2^n, n!)$

$\Rightarrow (t_2(n), t_4(n), t_5(n), t_7(n), t_1(n), t_6(n), t_3(n))$

3.

$$\begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

We need to calculate F_n in $O(\log n)$ using above matrix expression.

In simple way, It goes $O(n)$ when calculate $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$ for n times. $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^0 \to \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^1 \to \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 \to \ldots$ n times

So, It's the key for problem that reducing calculate time for $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$.

And so on, $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$ can be calculated by $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2}$, when n is odd, just multiply $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ one more time to $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2}$.

In order to calculate $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$, Using half exponent reduce It's processing time to $\underline{\underline{\log_2 n}}$ because we just need to know $\underline{\underline{\text{half}}}$ of each $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$ up to $n = 1$.

$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2}$

Thus, here is algorithm to calculate F_n in $O(\log n)$.

1. If $n=1$, return $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$.
2. calculate $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2}$ for even number n.
   If n is odd calculate $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{\lfloor n/2 \rfloor} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{\lfloor n/2 \rfloor} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$
3. recursively call calculate $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2}$ up to $n=1$
4. calculate $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix}$

   And we get F_n.

In this algorithm, calculating $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n$ by multipling $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n/2}$ takes $\log_2 n$ times. Because size of exponent is going to be half of it's previous size until size gets 1.