Assignment 1 Algorithms & Complexity

1. Design an algorithm to find all palindromes(迴文) of length ≥ 2. It does not need to be an optimal algorithm, as long as it can solve the problem.
2. Analyze the every-case (if exists), worst-case, average-case, and best-case time complexities of your algorithm.
3. Textbook exercises 1-15~18, 1-22.

Due date: two weeks.

Assignment 1 Algorithms & Complexity

1-15 Show directly that $f(n) = n^2 + 3n^3 \in (n^3)$. That is, use the definitions of O and Ω to show that f(n) is in both $O(n^3)$ and $\Omega(n^3)$.

1-16 Using the definitions of O and Ω, show that

$$6n^2 + 20n \in O(n^3), \text{ but } 6n^2 + 20n \notin \Omega(n^3).$$

1-17 Using the Properties of Order in Section 1.4.2, show that

$$5n^5 + 4n^4 + 6n^3 + 2n^2 + n + 7 \in \Theta(n^5).$$

1-18 Let $p(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$, where $a_k > 0$. Using the Properties of Order in Section 1.4.2, show that $p(n) \in \Theta(n^k)$.

1-22 Group the following functions by complexity category.

$$n \ln n \qquad (\lg n)^2 \qquad 5n^2 + 7n \qquad n^{5/2}$$

$$n! \qquad 2^{n!} \qquad 4^n \qquad n^n \qquad n^n + \ln n$$

$$5^{\lg n} \qquad \lg(n!) \qquad (\lg n)! \qquad \sqrt{n} \qquad e^n \qquad 8n + 12 \qquad 10^n + n^{20}$$

1 (constant) $< \textbf{log } \boldsymbol{n} < \boldsymbol{n} < \boldsymbol{n} \textbf{ log } \boldsymbol{n} < \boldsymbol{n^2} < \boldsymbol{n^3} < \boldsymbol{2^n} < \boldsymbol{3^n} \leq$ **n!**

Assignment 1 Algorithms & Complexity

名稱	公式	證明
和差	$\log_\alpha MN = \log_\alpha M + \log_\alpha N$	設$M=\beta^m$，$N=\beta^n$ $\log_\alpha MN = \log_\alpha \beta^m\beta^n$ $= \log_\alpha \beta^{m+n}$ $= (m+n)\log_\alpha \beta$ $= m\log_\alpha\beta + n\log_\alpha\beta$ $= \log_\alpha \beta^m + \log_\alpha \beta^n$ $= \log_\alpha M + \log_\alpha N$ $\log_\alpha \frac{M}{N} = \log_\alpha M + \log_\alpha \frac{1}{N}$ $= \log_\alpha M - \log_\alpha N$
基變換（換底公式）	$\log_\alpha x = \frac{\log_\beta x}{\log_\beta \alpha}$	設 $\log_\alpha x = t$ $\therefore x = \alpha^t$ 兩邊取對數，則有$\log_\beta x = \log_\beta \alpha^t$ 即 $\log_\beta x = t\log_\beta \alpha$ 又$\because \log_\alpha x = t$ $\therefore \log_\alpha x = \frac{\log_\beta x}{\log_\beta \alpha}$
指係(次方公式)	$\log_{\alpha^n} x^m = \frac{m}{n}\log_\alpha x$	$\log_{\alpha^n} x^m = \frac{\ln x^m}{\ln \alpha^n}$ $= \frac{m\ln x}{n\ln\alpha}$ $= \frac{m}{n}\log_\alpha x$

Source:wiki

Assignment 1 Algorithms & Complexity

還原	$\alpha^{\log_\alpha x} = x$ $= \log_\alpha \alpha^x$	
互換	$M^{\log_\alpha N} = N^{\log_\alpha M}$	設 $b = \log_\alpha N, c = \log_\alpha M$ 則有 $\alpha^c = M$ and $\alpha^b = N$. 公式左側是 $(\alpha^c)^b$ 公式右側是$(\alpha^b)^c$
倒數	$\log_\alpha \theta = \frac{\ln\theta}{\ln\alpha} = \frac{1}{\frac{\ln\alpha}{\ln\theta}} = \frac{1}{\log_\theta \alpha}$	
鏈式	$\log_\gamma \beta \log_\beta \alpha = \frac{\ln\alpha}{\ln\beta}\frac{\ln\beta}{\ln\gamma}$ $= \frac{\ln\alpha}{\ln\gamma}$ $= \log_\gamma \alpha$	

log log n = log (log n)

$\log^k n = (\log n)^k$

Source:wiki