Growth of Functions

3.2 Asymptotic notation

Θ-notation: $f(n) = \Theta(g(n))$，$g(n)$ is an asymptotically tight bound for $f(n)$。

$\Theta(g(n)) = \{f(n)|$ 存在大於零的常數 c_1，c_2，以及 n_0 使得 $0 \le c_1\, g(n) \le f(n) \le c_2\, g(n)$ 對於所有的 $n \ge n_0$ 都成立$\}$

範例: 證明 $3n^2 - 6n = \Theta(n^2)$。

證明:

為了證明上面的式子，我們必須找到 c_1, c_2，和 n_0 符合下面的不等式：

$$c_1 n^2 \le 3n^2 - 6n \le c_2 n^2， \quad (\text{對所有 } n \ge n_0)$$

同除以 n^2 得到

$$c_1 \le 3 - 6/n \le c_2$$

很明顯地，只要選擇 $c_1=2$，$c_2=3$ 以及 $n_0=6$ 我們就可以證明 $3n^2 - 6n = \Theta(n^2)$。

得證

註： $f(n) = \Theta(g(n))$ 若且唯若 $g(n) = \Theta(f(n))$，例如: $n^2 = \Theta(3n^2 - 6n)$

O-notation: $f(n) = O(g(n))$，$g(n)$ is an asymptotically upper bound for $f(n)$。

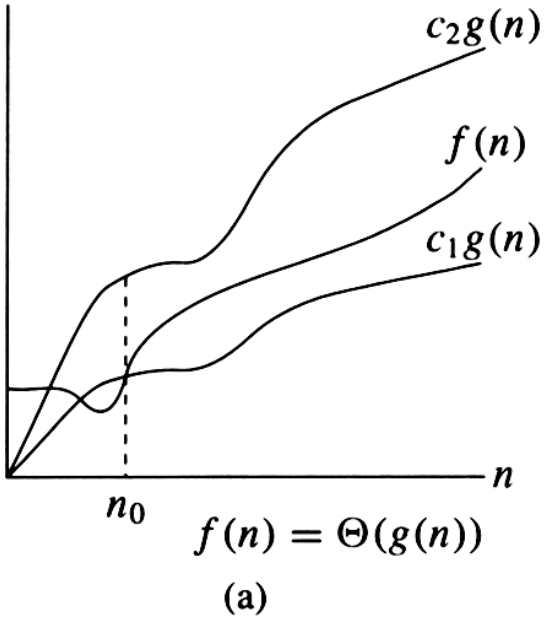
$O(g(n)) = \{f(n)|$ 存在大於零的常數 c and n_0 使得 $0 \leq f(n) \leq c_2\, g(n)$ 對於所有的 $n \geq n_0$ 都成立$\}$

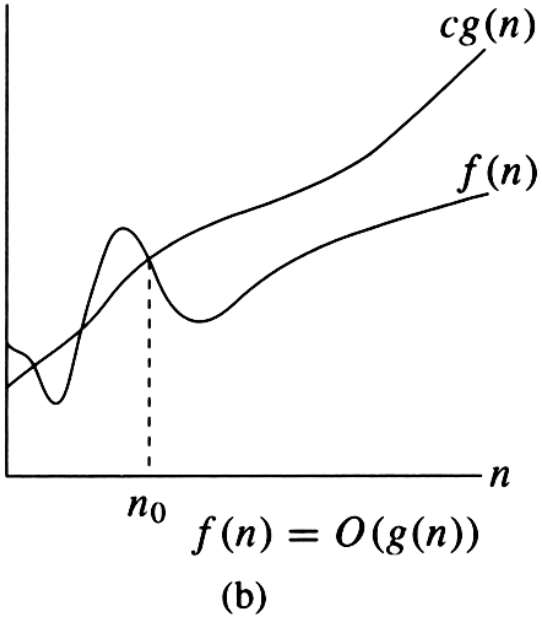
- $\Theta(g(n)) \subseteq O(g(n))$
- $f(n) = \Theta(g(n))$ 意味著 $f(n) = O(g(n))$
- $6n = O(n)$，$6n = O(n^2)$
- “執行時間為 $O(n^2)$” 表示 “在最糟的情況下執行時間為 $O(n^2)$”

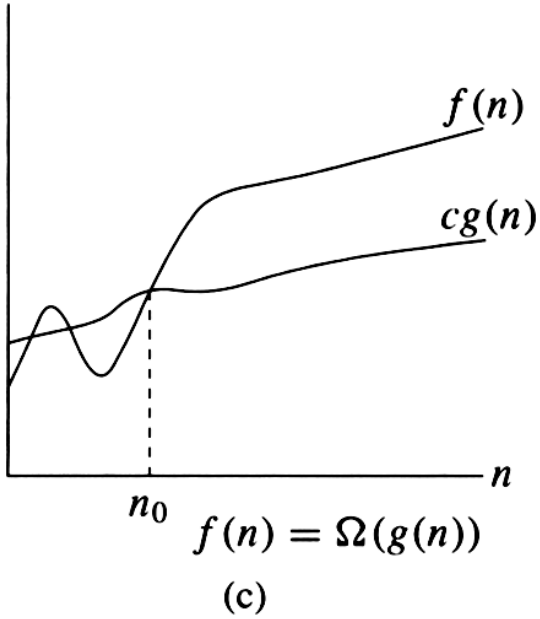
Ω-notation: $f(n) = \Omega(g(n))$，$g(n)$ is an asymptotically lower bound for $f(n)$。

$\Omega(g(n)) = \{f(n)|$ 存在大於零的常數 c 和 n_0 使得 $0 \leq cg(n) \leq f(n)$ 對於所有 $n \geq n_0$ 都成立$\}$

註： $f(n) = \Theta(g(n))$ 若且唯若 $(f(n)=O(g(n))) \ \& \ (f(n)=\Omega(g(n)))$







tight bound

upper bound

lower bound

***o*-notation**: $f(n) = o(g(n))$ (little-oh of g of n)

$o(g(n)) = \{f(n)|$ 對於任何大於零的常數 c，都存在一個常數 $n_0 > 0$ 使得 $0 \le f(n) < cg(n)$ 對於所有的 $n \ge n_0$ 都成立$\}$

- $2n = o(n^2)$，但是 $2n^2 \neq o(n^2)$
- $f(n) = o(g(n))$ 也可以被定義成 $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

ω-*notation*: $f(n) = \omega(g(n))$ (little-omega of g of n)

$\omega(g(n)) = \{f(n)|$ 對於任何大於零的常數 c，都存在一個常數 $n_0 > 0$ 使得 $0 \leq cg(n) < f(n)$ 對於所有 $n \geq n_0$ 都成立$\}$

- $2n^2 = \omega(n)$，但是 $2n^2 \neq \omega(n^2)$
- $f(n) = \omega(g(n))$ 若且唯若 $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

Comparison of functions

- 函數: ω Ω Θ O o
  實數: $>$ $\geq$ $=$ $\leq$ $<$

- Transitivity，Reflexivity，Symmetry

- 任兩實數皆可互想比較大小(trichotomy)，但是任兩函數並不一定能夠互相比較。
  - 例如： $f(n)=n$ and $g(n)=n^{1+\sin n}$

Appendix A: Summation formulas

$$\sum_{k=1}^{n}(ca_k + b_{k)} = c\sum_{k=1}^{n} a_k + \sum_{k=1}^{n} b_k$$

$$\sum_{k=1}^{n} k = \frac{1}{2} n(n+1) = \Theta(n^2) \qquad \sum_{k=0}^{n} x^k = (x^{n+1} - 1)/(x-1)$$

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = \log_e n + O(1) \quad \text{(Harmonic series)}$$

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

$$\sum_{k=1}^{n-1} \frac{1}{k(k+1)} = \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1}\right) = 1 - \frac{1}{n}$$

$$\lg \prod_{k=1}^{n} a_k = \sum_{k=1}^{n} \lg a_k$$

Exercises

Problem 1:

為了解決一個問題而設計程式時，分析該演算法的執行時間複雜度是個很重要的依據。線性時間的演算法通常要比二次方時間的演算法受大家歡迎。

通常，問題的大小 *n* 可以決定演算法的執行時間，也許是要被排序的數字個數，或是多邊形的點的數目，等等。由於要算出一個演算法相對於 *n* 的執行時間公式不是很容易，如果能夠自動化那就太棒了。很不幸地，一般來說這是不太可能做到的，但是我們在這邊要考慮的程式是非常簡單的，所以把不可能變成了可能。我們的程式是根據下面的規則所建立的（BNF格式），其中 <***number***> 是大於等於零的整數。

Exercises

- <***Program***> ::= "BEGIN" <***Statementlist***> "END"
- <***Statementlist***> ::= <***Statement***> | <***Statement***> <***Statementlist***>
- <***Statement***> ::= < ***LOOP-Statement***> | <***OP-Statement***>
- <***LOOP-Statement***> ::= <***LOOP-Header***> <***Statementlist***> "END"
- <***LOOP-Header***> ::= "LOOP" <***number***> | "LOOP n"
- <***OP-Statement***> ::= "OP" <***number***>

程式的執行時間可以計算如下：*OP-statement* 的執行時間就跟它的參數一樣。被 <*LOOP-Statement*> 包起來的區段則是會執行很多次，有可能會執行常數次（如果給定的 LOOP 參數是常數），或是執行 *n* 次（如果給定的 LOOP 參數是 *n*）。一段 statement 的執行時間只要把構成那段 statement 的全部時間加總起來就是答案。因此程式的執行時間一般來說會跟 *n* 有關係。

Exercises

輸入：第一行會有一個整數 k 表示有幾個程式需要處理。接下來會有 k 個符合之前規則的程式。空白字元以及換行可能會出現在程式中的任何地方，但不會出現在關鍵字或是數字之間，比如 BEGIN, END, LOOP, OP。LOOP 的深度最大只會到 10 。

輸出：對於每個程式，第一行先輸出程式的編號，如輸出實例所示。接著要輸出程式的執行時間，這會是一個跟 n 有關的多項式，最大的 degree 只會到 10。用平常表示多項式的方法印出來，格式如下：

”Runtime = $a*n\text{^}10+b*n\text{^}9+\cdots+p*n\text{^}2+q*n+r$”，省略係數是 0 的項次。係數是 1 的話該係數不用寫出來（除了常數項）。如果執行時間是 0，則印出”Runtime = 0”。在每組測資之後印一個空行。

以下是一個輸出入的實例:

Sample Input	Sample Output
2 BEGIN LOOP n OP 4 LOOP 3 LOOP n OP 1 END OP 2 END OP 1 END OP 17 END BEGIN OP 1997 LOOP n LOOP n OP 1 END END END	Program #1 Runtime = 3*n^2+11*n+17 Program #2 Runtime = n^2+1997