**時間複雜度 — 漸進函數**

相信有在學習程式的各位一定常常聽到「時間複雜度」一詞，礙於之前對於Big-O, Omega, theta的概念一直處於一知半解的狀態，導致目前大概是學幾次就忘幾次，決定透過這次把它好好記下來。

由於這學期有選修演算法相關的課程，想透過這次機會邊上課學習邊做筆記。還希望各位讀者看過這篇過後，可以理解其中的意含。

**先前對於Ｂｉｇ－Ｏ的記憶：**

*Big-O：時間複雜度****最糟****的狀況。*

*Omega：時間複雜度****最好****的狀況。*

*Theta：時間複雜度****平均的狀況？？***

如果各位對於時間複雜度的概念，就像我一樣的話，很希望這篇可以帶給各位一些新的觀念與想法。那就讓我們開始吧~~

**本篇章節：**

*1. 漸進符號的功能*

*2. 介紹各個漸進符號 （Big-O, Omega, Theta, Little-O, Little-Omega）*

**1. 漸進符號的功能（Asymptotic notation）**

**為甚麼會需要漸進符號呢？**

—> 用來粗略估計時間複雜度的工具/手段。

簡單的定義：

**當 *n -> 無限大****, T(n)函數所估計的時間複雜度。------------------------------------------------*＊這邊的T(n)不是什麼可怕的東西，只是一個**函數的關係**而已。＊函數不可怕，其實只要想成是「***存在某種關係」***就對了。  
------------------------------------------------例如：  
 今天某某店全面打九折，所以他們的收銀機可能會出現下方的函式。  
   
 當看到某商品售價500元，我們很直觀地想說，打9折，不就是乘上0.9就好了嗎  
 心中算盤開始運作 500 \* 0.9 = 450元 那我們把它轉換成數學的樣子來呈現：  
 **f(x) = 0.9 \* x** **售價(y) = f(x) = 0.9 \* x**，其中 x 代表商品的原價。  
 **而f(x)所代表的是：「商品原價 x 」跟「售價 y」有9折的「關係」。**

如果還***不太清楚粗略估計是什麼***的話，沒關係，請看下方的舉例。

先用個生活上的範例來舉例。

*今天和朋友出國玩耍，免不了的一定會好奇一下對方帶了多少錢，所以就問說：「誒誒誒，你們這次出來玩，帶了多少錢啊？」*

*小城說：「 差不多 2萬。」*

*小春說：「 2萬 3千 512元！」*

從上面的例子來說，一般人應該會覺得小春回答的答案有點奇怪，太過精確了，其實提問者只想要知道大概的情況，聽到太精準還會覺得有點奇怪呢。

只想要**知道大概的情況**，就是漸進函數的精神！

如果還是不太理解我們再看一個例子。

下方為Insertion Sort的時間複雜度。

***Worst Case： T(n) = a n² + b n + c （ 其中 a, b, c 皆為常數 ）***

*老闆比爾問說：「最近專案使用的 Insertion sort，時間複雜度為多少？」*

*小澤豪不猶豫地說：「n²」*

*小御自信滿滿的說：「an² + bn + c」*

其實兩個答案都是對的，只是如果像小御一樣回答得這麼精確，就會像前一個例子，小春回答有相同的感覺，有點太多餘了。

**而漸進函數的出現就是想要粗略的預估演算法所花費的時間。**

最後一個是我之前常常出現的問題

*老師出的考卷：請問 4n³ + 98n² – 20*n*+ 14，請算出其 Θ()為多少？*

*這題多數人可能很快就會在答案卷上面寫：Θ*(n³)。

如果是我的話，可能靠著之前練習題目所培養的手感，毫不猶豫填入 n³，但是其實不太清楚為什麼會寫這個答案，下次只要題目變個形式又會出錯了。

所以就讓我們擺脫之前似懂非懂的狀態，來真正理解其中的意涵吧！

首先，我們先來介紹一下這些符號

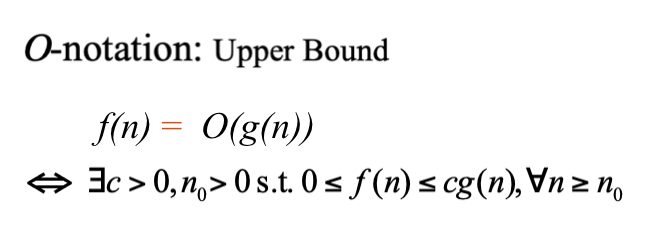
因為這些符號在演算法中佔有十分重要的地位，若行有餘力，請好好將這些符號的定義記起來（多寫幾遍差不多可以記住了，因為每個符號間的關係只有些微的差異），在未來解題時會是個有效的工具！

定義這些符號時，會有些枯燥，還請各位讀者心平氣和地把它看完。

**2. 各個漸進符號的介紹 （Big-O, Omega, Theta, Little-O, Little-Omega）**

**1). 𝑂 (g(n))：Big-O notation（上界）**

我們最熟悉的Big-O，中文的角度可以說他是該函數的**寬鬆的上界**。其定義如下（看到英文，不要害怕，在下一張的圖片有附上自己轉換的中文定義，可以搭配著看）：



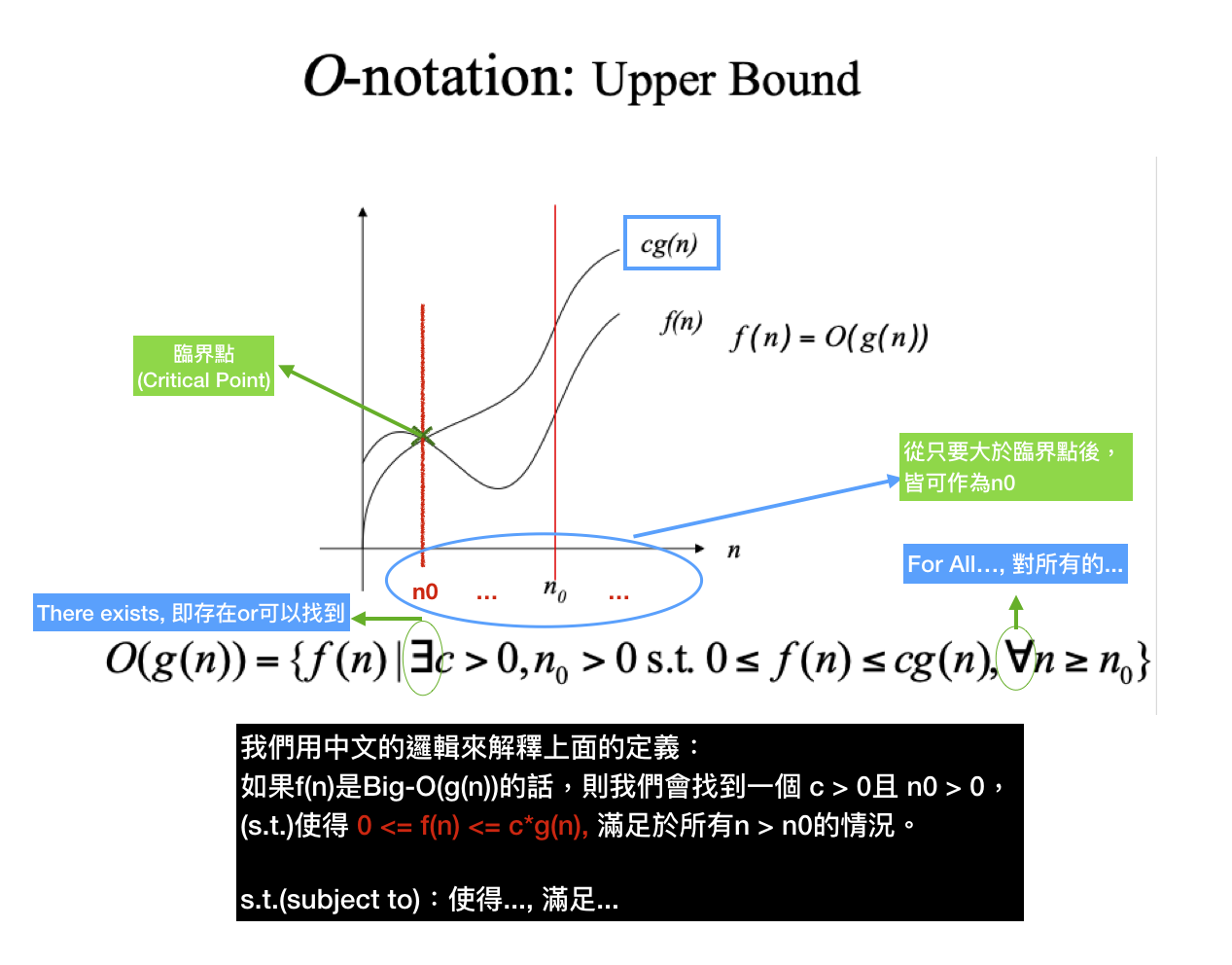
注意！！上圖中的等號，代表的是 is 的意思，

而非數學中「等於」的概念。

*f(n) = 𝑂 (g(n))*

*中文：****g(n)*是*f(n)的上界****，或是，****f(n)的上界函數是 g(n)***

*英文：g(n)****“is”*** *an upper bound of f(n)*



**看完圖片可能會產生的疑問：**

***1. 為什麼是 0 <= f(n)？*** *Ans: 因為我們的 f(n)代表的是演算法的執行時間（個人偏好想成「執行步驟」），常理說是不可能會出現「負」的執行步驟才是。*

*其中 n代表的是 輸入的****資料個數****，也不可能會出現「負」的資料個數。*

***2. n0是什麼？*** *Ans: 由上圖來看，我們有兩個函數( f(n), c\*g(n) )，他們有兩個不同的圖形，從圖上來看在****臨界點之前，f(n) > c\*g(n)；****而在臨界點後，Big-O的定義才開始****恆成立****，而那些位於臨界點之後，皆可成為 n0。*

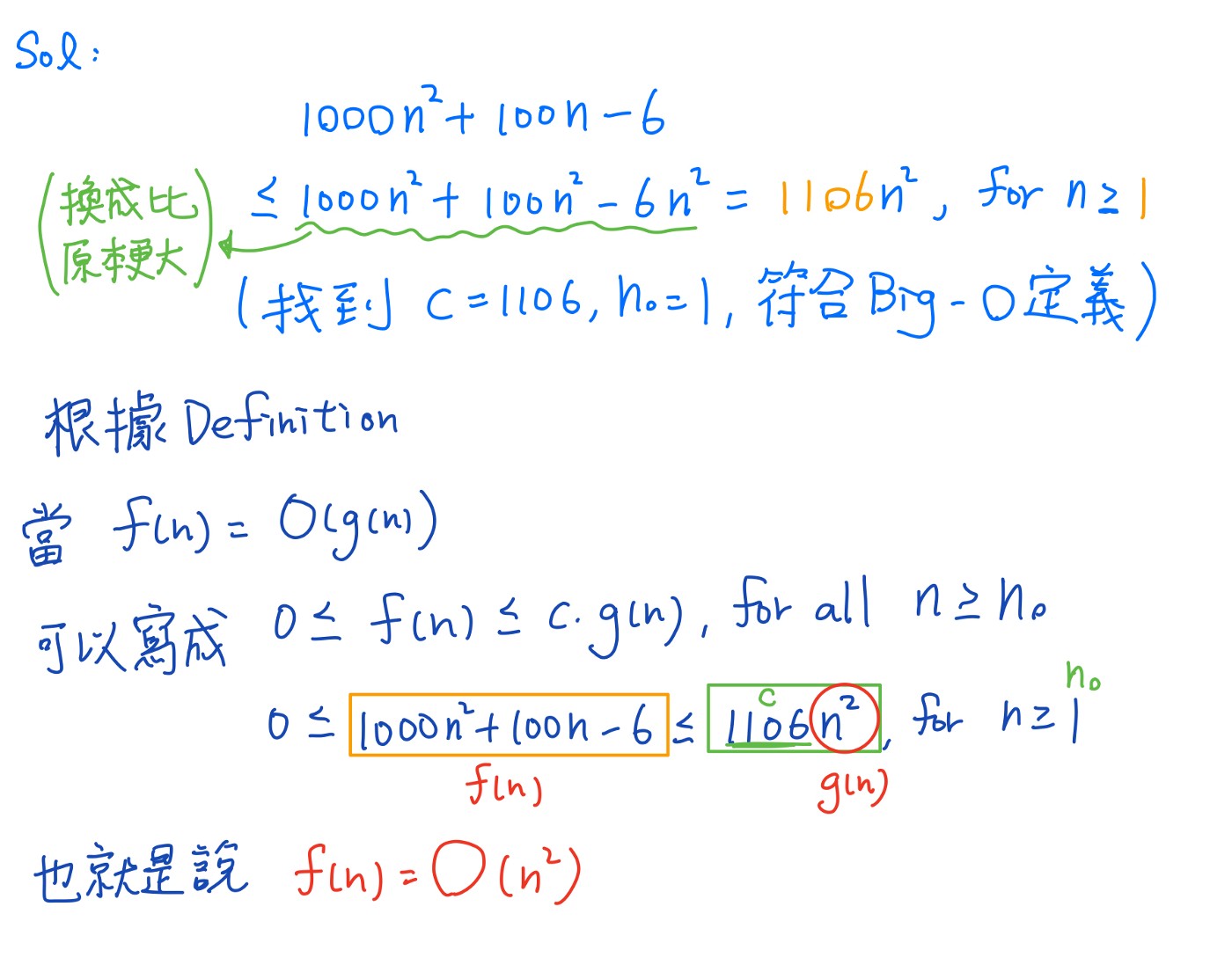
*＊可以想成只有在 n0成立的條件下，「定義」才會成立！！*

***3. g(n)是什麼？*** *Ans: g(n)可以想成是符合那些定義條件中的某個代表。（看完以下範例後，最後有作個解釋）*

***4. c \* g(n)中的 c是什麼？*** *Ans: c是 const（常數）可以想成是為了成為代表，需要多加配件來鞏固自己的地位。（看完以下範例後，最後有作個解釋）*

直接用例子來看：

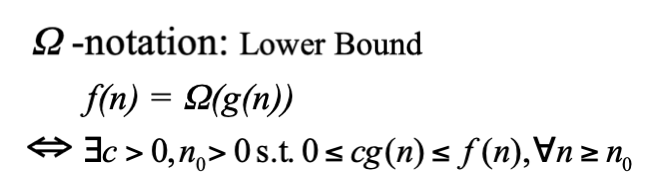
Example:  
 1000n^2 + 100n - 6 = O(n^2)

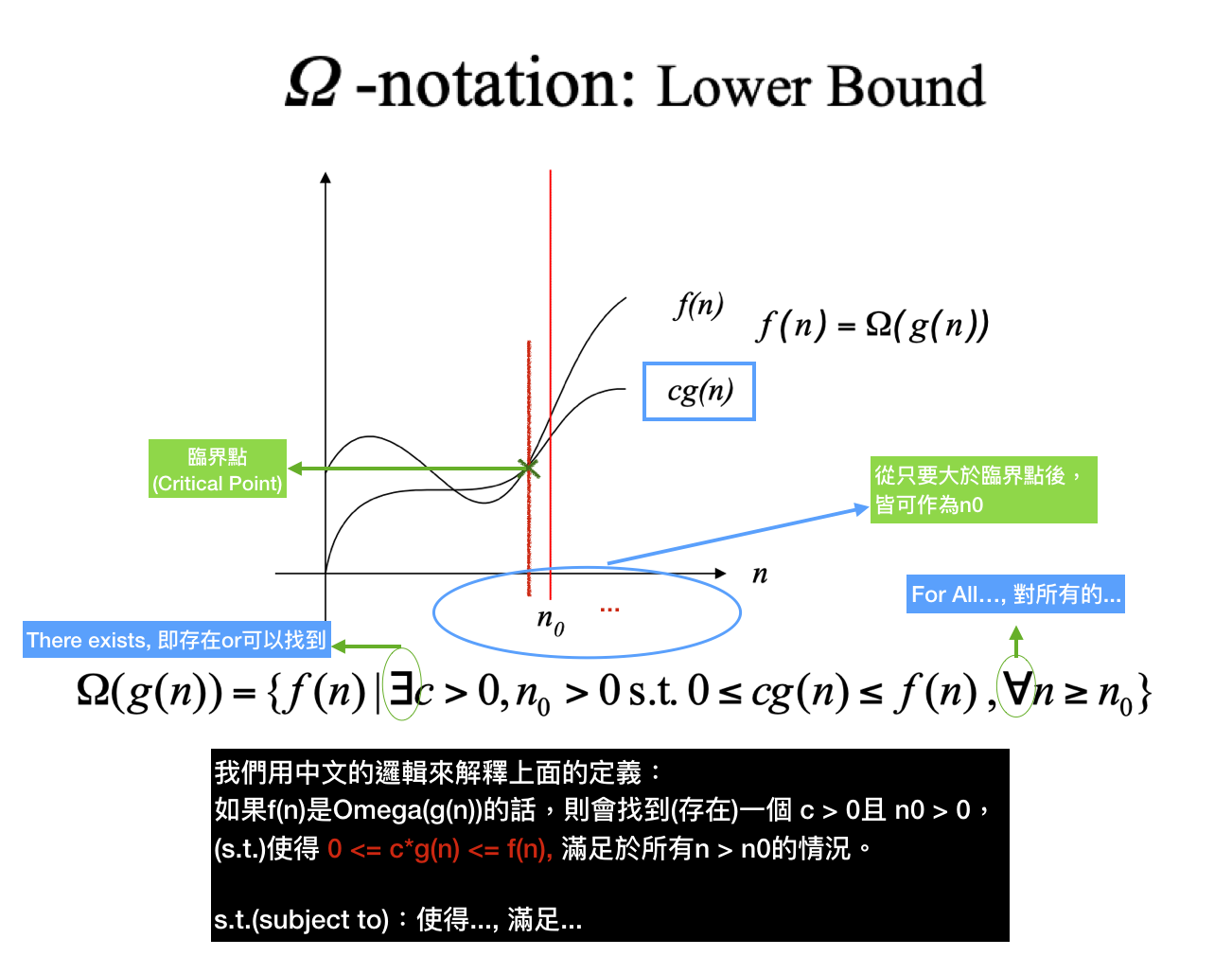


以剛剛的範例來看，我要找的g(n)只要**符合定義要的條件即可。**可想而知，g(n)一定不可能只會有一個，還可能會有n³, n⁴等等，比f(n)大的函數，大有人在，而我們所選的g(n)，只是從茫茫人海中找一個出來罷了。（所以在這邊稱選到的g(n)為「代表」）

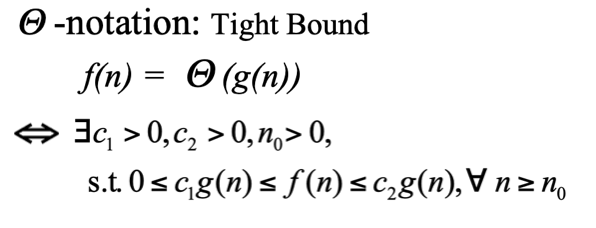
下面的部分可能就不會像Big-O講得一樣詳細，因為大部分的概念都很雷同，還請各位讀者多多包涵～～

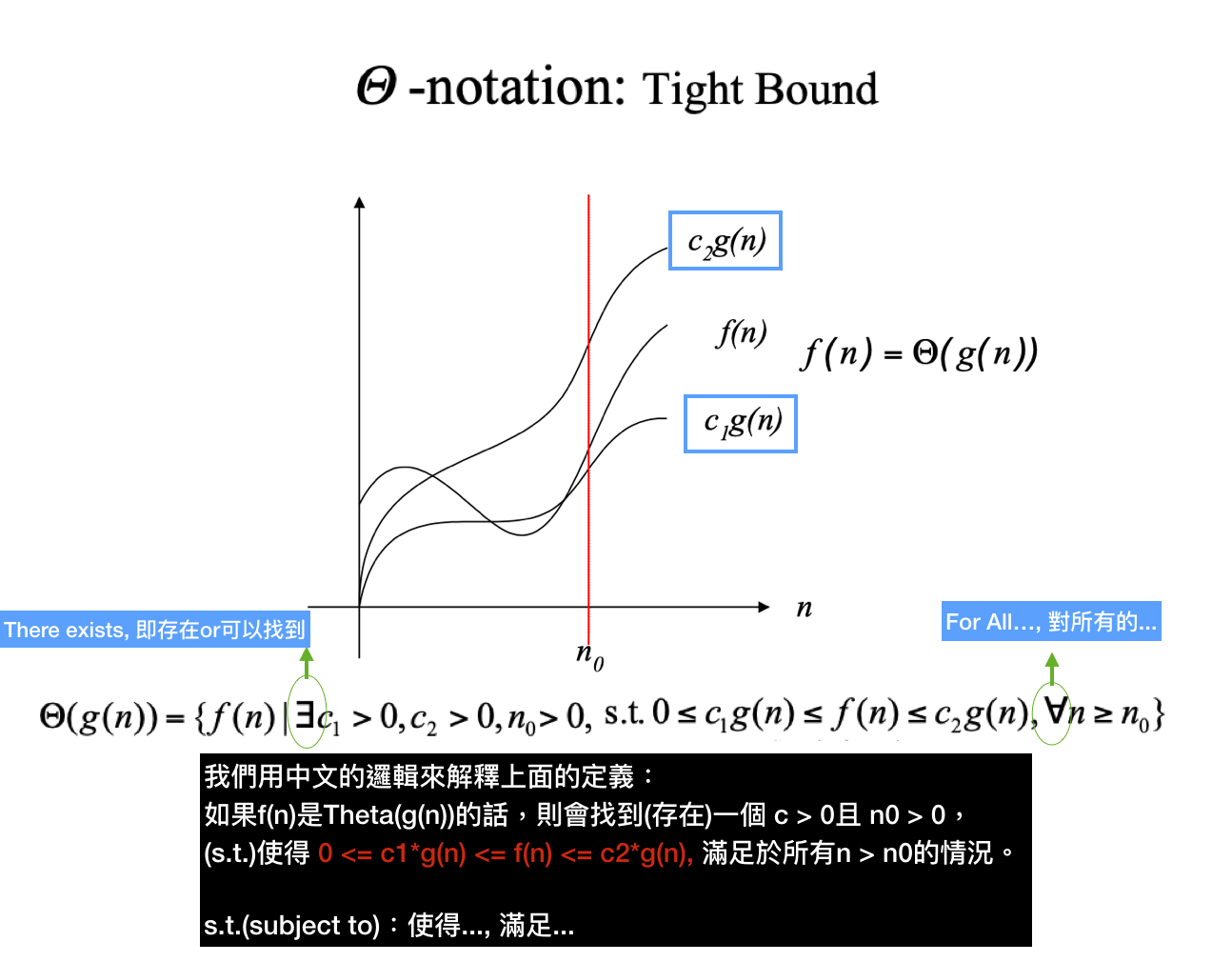
**2). Ω (g(n))：Omega notation（下界）**





**3). Θ (g(n))：Theta notation（嚴格上下界）**

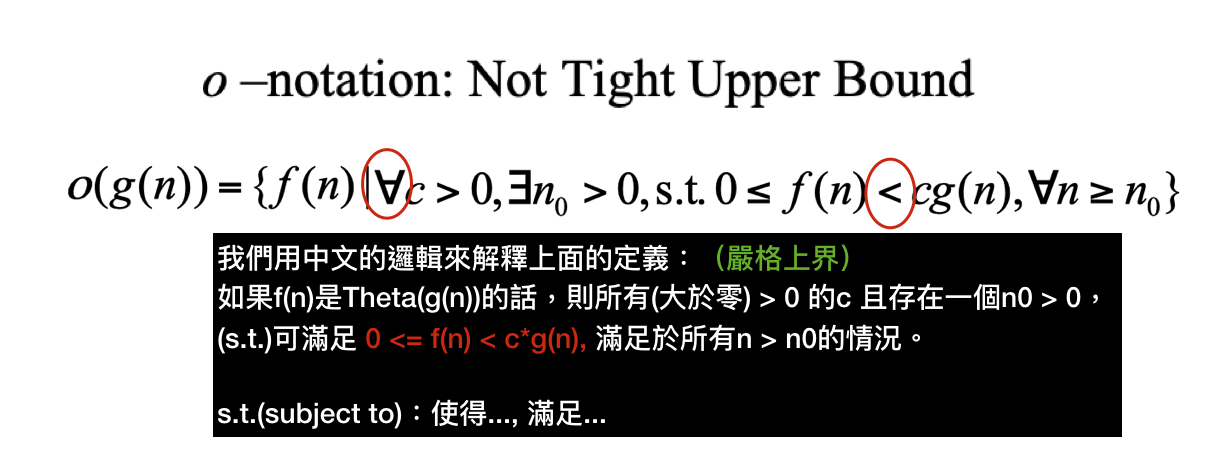




**4). 𝑜 (g(n))：Little-O notation（嚴格上界）**

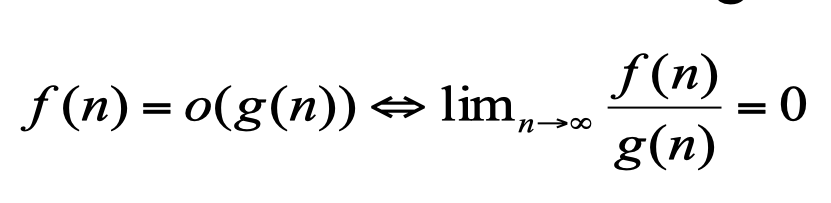
這邊的概念，是從Big-O出來作延伸，跟Big-O不同的是，**Little-o的所需的條件更多**。

**定義一：**



**定義二：**

**如果 f(n) 是 little-o(g(n)) 的話，則當n -> 無限大，我們取兩個函數間的「比值」會趨近於0。（請參考下方圖片）＊＊用直觀一點的想法來說，如果取比值後會等於0，則代表我的f(n) < g(n)，  
( g(n)的增長速度比f(n)快很多 )，也就是說，g(n)會成為f(n)的嚴格上界。**



***可能這邊有些人會又個疑問說為什麼用這個方法，不能也是Big-O呢？*** *Ans: 因為在我們先前的定義中，Big-O中有****包含「等於」的概念****，所以當「函數 f(n)等於 函數 g(n)」的狀態，取比值後會是一個「常數」，如此一來對嚴謹的數學來說定義不夠明確。*

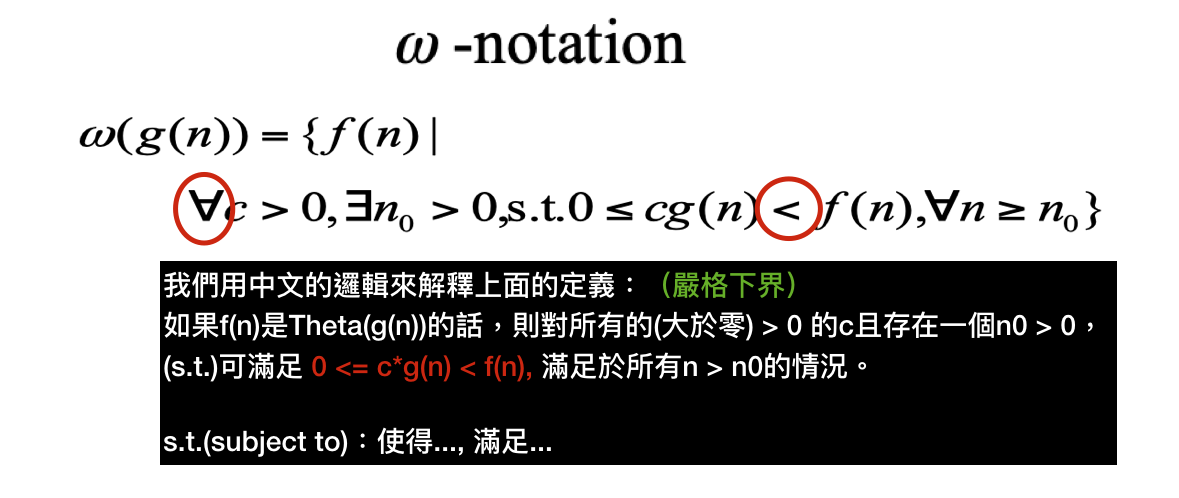
取極限的計算，需用到羅必達法則。

（如果對於羅必達法則還不是很熟練的朋友可以透過這次學習的機會把他了解清楚～～）

**5). ω (g(n))：Little-Omega notation（嚴格下界）**

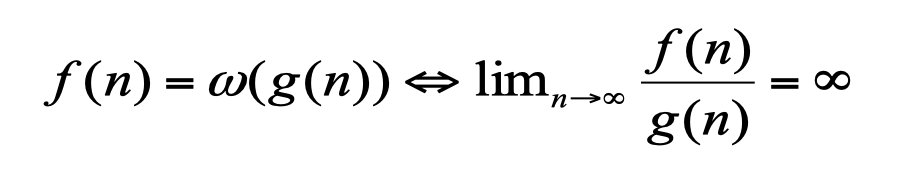
這邊的概念，是從Omega出來作延伸，跟Omega不同的是， Little-Omega所需的條件更多。

**定義一：**



**定義二：**

**如果f(n)是litter-omega(g(n))的話，則當 n -> 無限大，我們取兩個函數間的「比值」會趨近無限。（請參考下圖）用直觀一點的想法來說，如果取比值後會等於0，則代表我的f(n) > g(n)，  
( f(n)的增長速度比g(n)快很多 )，也就是說，g(n)會成為f(n)的嚴格下界。**



***和剛剛相同的概念，為什麼用這個方法不能也是 Omega呢？***

*Ans: 因為在我們先前的定義中，Omega中有包含「等於」的概念，所以當處䱷「f(n)等於g(n)」的狀態，取比值後會是一個「常數」，如此一來對嚴謹的數學來說定義不夠明確。*

如果目前對於上述的概念，還沒有很了解的話，請看我[下一篇文章](https://mycollegenotebook.medium.com/%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6-%E5%87%BD%E6%95%B8%E9%96%93%E7%9A%84%E6%AF%94%E8%BC%83-3e9ce0788f0f)，這裡面比較多的應用。可以透過範例來理解這些符號之間的關係。

不知不覺已經打了這麼多了，為了不要增加各位讀者的負擔（我也好累）。各函數間比較的部分，我們下一次再來講吧。謝謝各位的閱讀～～

如有錯誤還懇請多多指教！！

在精進自我的道路上一起加油吧！

100

1

100

1

[More from Sharon Peng](https://mycollegenotebook.medium.com/?source=post_page-----397ca19cdc4c--------------------------------)

Follow

一起精進程式能力吧！！

[Mar 13, 2021](https://mycollegenotebook.medium.com/python-csv-ufeff-%E5%95%8F%E9%A1%8C-4de92dc5a002?source=post_page-----397ca19cdc4c----0----------------------------)

**[Python CSV — \ufeff 問題](https://mycollegenotebook.medium.com/python-csv-ufeff-%E5%95%8F%E9%A1%8C-4de92dc5a002?source=post_page-----397ca19cdc4c----0----------------------------)**

[這個問題已經困擾筆者我蠻多次了，想說記錄下來看能不能加深印象，也希望能幫助遇到相同困難的讀者。 怕讀者不知道下方讀取是什麼樣類型的檔案，為了方便理解，做個簡單的介紹。 下方輸入的csv檔案(0311\_output)是一個簡易的記帳表格，每一列各代表 編號，日期，價錢。 這邊先放上可能會出現\ufeff 的程式碼 印出來結果如下：](https://mycollegenotebook.medium.com/python-csv-ufeff-%E5%95%8F%E9%A1%8C-4de92dc5a002?source=post_page-----397ca19cdc4c----0----------------------------)

[Csv](https://medium.com/tag/csv?source=post_page-----397ca19cdc4c---------------csv-----------------)

[2 min read](https://mycollegenotebook.medium.com/python-csv-ufeff-%E5%95%8F%E9%A1%8C-4de92dc5a002?source=post_page-----397ca19cdc4c----0----------------------------)

Share your ideas with millions of readers.

[Write on Medium](https://medium.com/new-story?source=post_page_footer_cta_write-------------------------------------)

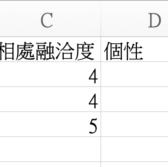
[Mar 4, 2021](https://mycollegenotebook.medium.com/python%E8%AE%80%E5%8F%96csv%E6%AA%94%E6%A1%88%E8%B3%87%E6%96%99-e74e0eb4ea26?source=post_page-----397ca19cdc4c----1----------------------------)

**[利用Python讀取CSV檔案資料](https://mycollegenotebook.medium.com/python%E8%AE%80%E5%8F%96csv%E6%AA%94%E6%A1%88%E8%B3%87%E6%96%99-e74e0eb4ea26?source=post_page-----397ca19cdc4c----1----------------------------)**

[這篇主要針對CSV檔案做讀取(Read)與寫入(Write)的介紹，也會介紹一些基本的概念。 如果是剛接觸程式不久的新手，聽到讀取檔案資料應該會很困惑吧，什麼叫做讀檔或是寫檔？（筆者我一開始沒接觸前也不知道會讀寫檔可以幹嘛？後來在實際應用過，才發現是個很好用的工具。 所以這篇就讓我介紹一下這項好用的工具吧！！ 什麼是讀寫檔？ 我們可以想得簡單一點，今天打開Word，然後直接在上面寫文字並存檔。其實即是「寫檔案(Write)」的動作。 讀檔(Read)：跟字面上的意思一樣，只能「讀取」檔案，不能對檔案做修改。可以想看看，有時開啟一些Word檔時，一開始可能會在上方顯示「受保護檢視」，如下圖，這代表說這份文件我們只能看（讀取），不能做任何更動。如果要更改的話，只要按下啟用編輯即可（這又回到剛剛所說的「寫檔案」的動作） 那可能有些人又會有個疑問，平常我在寫程式，或者是練題目的時候，幾乎都是用num = scanf(…)或是num = input()，等待系統自動輸入，或是自己做輸入，根本不需要讀檔啊。](https://mycollegenotebook.medium.com/python%E8%AE%80%E5%8F%96csv%E6%AA%94%E6%A1%88%E8%B3%87%E6%96%99-e74e0eb4ea26?source=post_page-----397ca19cdc4c----1----------------------------)

[Python](https://medium.com/tag/python?source=post_page-----397ca19cdc4c---------------python-----------------)

[5 min read](https://mycollegenotebook.medium.com/python%E8%AE%80%E5%8F%96csv%E6%AA%94%E6%A1%88%E8%B3%87%E6%96%99-e74e0eb4ea26?source=post_page-----397ca19cdc4c----1----------------------------)

[[](https://mycollegenotebook.medium.com/python%E8%AE%80%E5%8F%96csv%E6%AA%94%E6%A1%88%E8%B3%87%E6%96%99-e74e0eb4ea26?source=post_page-----397ca19cdc4c----1----------------------------)](https://mycollegenotebook.medium.com/python%E8%AE%80%E5%8F%96csv%E6%AA%94%E6%A1%88%E8%B3%87%E6%96%99-e74e0eb4ea26?source=post_page-----397ca19cdc4c----1----------------------------)

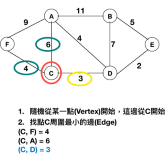
[Sep 30, 2020](https://mycollegenotebook.medium.com/prims-algorithm-47b7686de488?source=post_page-----397ca19cdc4c----2----------------------------)

**[Prim’s Algorithm](https://mycollegenotebook.medium.com/prims-algorithm-47b7686de488?source=post_page-----397ca19cdc4c----2----------------------------)**

[另一種找到最小生成樹(Minimum Spanning Tree)的方法。 ＊個人認為跟Kruskal’s Algorithm最不一樣的地方是，Prim’s 需要先有一個起始點，而Kruskal’s 是直接找最小邊(Edge)。 Prim’s Algorithm的步驟： 1. 隨機取一個點(Vertex)作為初始開始的起點。 2. 之後找尋此起點周圍最小的「邊(Edge)」，且取的邊不能形成Cycle(loop)。 3. 繼續延續連接後的點。 4. 重複步驟2，直到最後已經找不到邊可以取。 為了更詳細的說明，這邊也放了許多圖片，請大家耐心觀看。 下面圖片有很多圈圈，這邊做個簡單的說明。 綠色圈圈：可能的邊 黃色圈圈：最小的邊 紅色圈圈：新延伸到的點(Vertex) 圖例：](https://mycollegenotebook.medium.com/prims-algorithm-47b7686de488?source=post_page-----397ca19cdc4c----2----------------------------)

[Prims Algorithm](https://medium.com/tag/prims-algorithm?source=post_page-----397ca19cdc4c---------------prims_algorithm-----------------)

[2 min read](https://mycollegenotebook.medium.com/prims-algorithm-47b7686de488?source=post_page-----397ca19cdc4c----2----------------------------)

[[](https://mycollegenotebook.medium.com/prims-algorithm-47b7686de488?source=post_page-----397ca19cdc4c----2----------------------------)](https://mycollegenotebook.medium.com/prims-algorithm-47b7686de488?source=post_page-----397ca19cdc4c----2----------------------------)

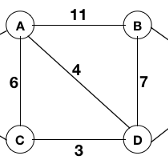
[Sep 30, 2020](https://mycollegenotebook.medium.com/kruskal-algorithm-deb0d64ce271?source=post_page-----397ca19cdc4c----3----------------------------)

**[Kruskal’s Algorithm](https://mycollegenotebook.medium.com/kruskal-algorithm-deb0d64ce271?source=post_page-----397ca19cdc4c----3----------------------------)**

[找到最小生成樹(Minimum Spanning Tree)的一種方法。 這部分先以，建立概念為主，之後有機會再把程式碼丟上來。 Kruskal’s Algorithm步驟如下： 1. 由小到大排序，所有「邊(Edge)」的權重(Weigh)。 2. 從小到大開始取那些「邊(Edge)」，前提是取到的Edge不能形成一個迴圈(loop, cycle)。 3. 重複步驟 2的動作，直到最後已經不能再取。 為了比較詳細的說明，下面投影片比較多。 圖例： 開始按照步驟進行](https://mycollegenotebook.medium.com/kruskal-algorithm-deb0d64ce271?source=post_page-----397ca19cdc4c----3----------------------------)

[Kruskals Algorithm](https://medium.com/tag/kruskals-algorithm?source=post_page-----397ca19cdc4c---------------kruskals_algorithm-----------------)

[2 min read](https://mycollegenotebook.medium.com/kruskal-algorithm-deb0d64ce271?source=post_page-----397ca19cdc4c----3----------------------------)

[[](https://mycollegenotebook.medium.com/kruskal-algorithm-deb0d64ce271?source=post_page-----397ca19cdc4c----3----------------------------)](https://mycollegenotebook.medium.com/kruskal-algorithm-deb0d64ce271?source=post_page-----397ca19cdc4c----3----------------------------)

[Sep 28, 2020](https://mycollegenotebook.medium.com/%E7%94%9F%E6%88%90%E6%A8%B9-spanning-tree-fa19df652efb?source=post_page-----397ca19cdc4c----4----------------------------)

**[生成樹 — Spanning Tree](https://mycollegenotebook.medium.com/%E7%94%9F%E6%88%90%E6%A8%B9-spanning-tree-fa19df652efb?source=post_page-----397ca19cdc4c----4----------------------------)**

[如果是第一次接觸的話，應該會對Spanning Tree這詞，感到很陌生。所以這次要帶各位來了解什麼叫做生成樹。 在看這篇之前，建議請先了解一下圖的結構是什麼，再來看會沒那麼複雜，因為會用到一些專有名詞。 什麼是生成樹？ 先用例子來解釋，現在我們有一個圖(Graph)稱為G，如下圖： 而生成樹就是 1. 新建一個圖，包含圖 G的所有「點(Vertex)」（這邊就是A, B, C, D） 2. 將所有「點」連接再一起，即為生成樹。（要特別注意，要用最少的邊(Edge)連接再一起才會是生成樹） 另一種想法是，可以連接每一個點而且用的是最少的邊。（這個概念可以想成郵差送信，每一戶(Vertex)都要去，但又想要花最少的時間(edges)） 如果看不懂沒關係，看下面的圖就能理解是什麼意思。](https://mycollegenotebook.medium.com/%E7%94%9F%E6%88%90%E6%A8%B9-spanning-tree-fa19df652efb?source=post_page-----397ca19cdc4c----4----------------------------)

[Spanning Tree](https://medium.com/tag/spanning-tree?source=post_page-----397ca19cdc4c---------------spanning_tree-----------------)

[4 min read](https://mycollegenotebook.medium.com/%E7%94%9F%E6%88%90%E6%A8%B9-spanning-tree-fa19df652efb?source=post_page-----397ca19cdc4c----4----------------------------)