**時間複雜度 —遞迴（上）**

這次要來討論的是 — Divide and Conquer(中文俗稱「分治法」），主要是把一個大大的問題，分成很多小問題，並一一破解他！

在解相關的問題時，其時間複雜度，通常都使用遞迴關係式來表達，所以這篇要帶大家來找divide and conquer的時間複雜度關係，也就是當我今天寫出一個divide and Conquer的演算法(可以想成Merge Sort)時，我要怎麼找到他的時間複雜度。

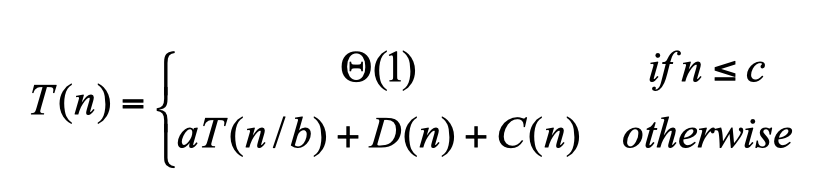
那就讓我們開始吧～～

**分析Divide & Conquer演算法時，我們通常使用以下兩種方法：**

1. Recursion-Tree Method搭配Substitution Method(這裡稱為「數學歸納法」)

2. Master Method：這個方法很直觀（有時間的話，建議先瞭解「方法一」再了解這個方法會比較全面）

首先，我們先來看一般Divide & Conquer的遞迴公式：



遞迴公式圖

*其中****T(n)****：問題所需要的時間（步驟）。*

***a T(n/b)****：****拆成 a個子問題****，每個子問題需要的時間。*

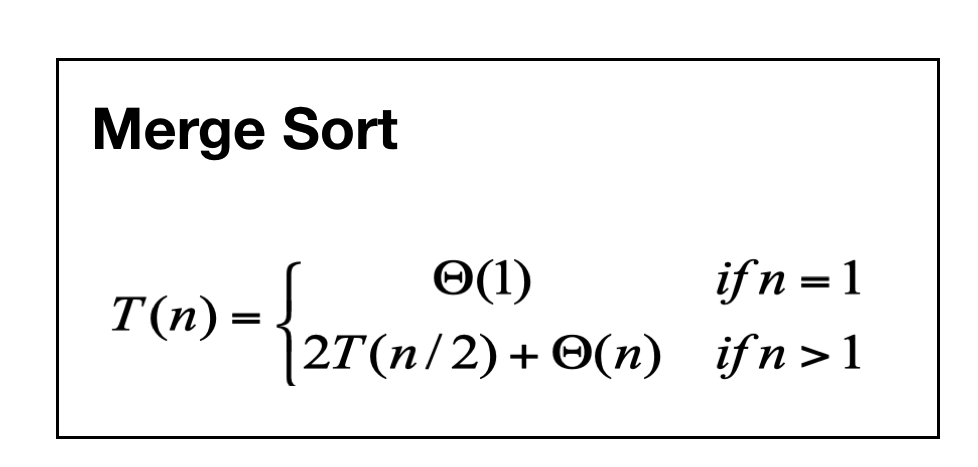
***＊(n / b)****代表的是，假設我們資料原本有n個，我****資料拆成 a個****後，每個所需要花費的時間。（如果不是很清楚沒關係，下面會舉例說明。）*

***D(n)****：****分成(Divide)****子問題所需的時間。*

***C(n)****：****合併(Combine)****子問題所需的時間。*

**使用Merge Sort來舉例：**

Merge Sort的遞迴關係式如下：



Merge Sort地回關係圖

*其中****T(n)****：問題所需要的時間（步驟）。*

***a T(n/b)****：****拆成 2個子問題****，每個子問題需要的時間是****T( n / 2 )****。*

***D(n)****：****分成(Divide)****子問題所需的時間。從程式碼來看的話，我們拆成 n/2的時間，可以想成****只要「一個除法」即可完成 n / 2****，所以****D (n) = Θ (1)****。*

***C(n)****：****合併(Combine)****子問題所需的時間。從程式碼的角度來看，我們****只需要一個 for迴圈****，就可以把兩個子問題合併，所以****C (n) = Θ (n)。***

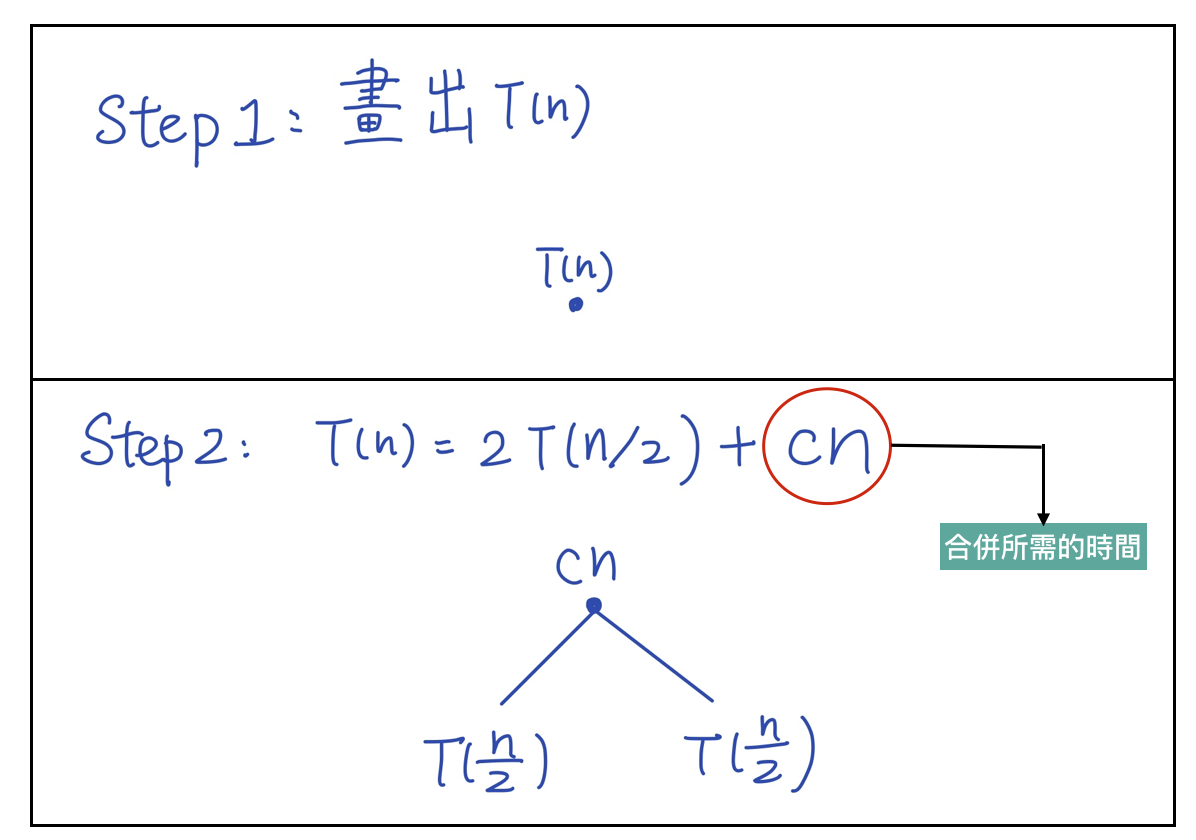
**找到遞迴式之後，要怎麼分析其時間複雜度呢？**

這邊先介紹方法一：

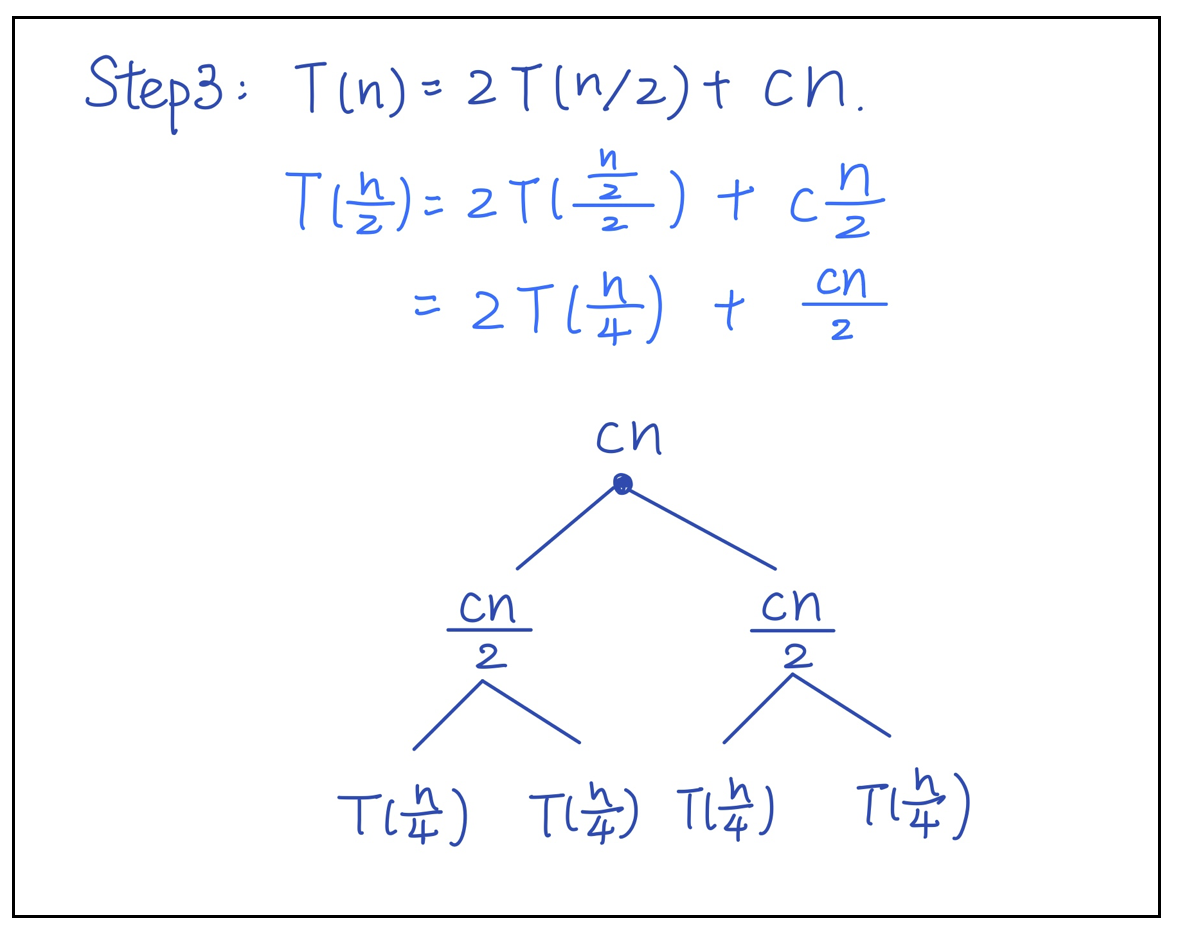
Recursion-Tree Method 搭配 Substitution Method

1. 利用Recursion-Tree，我們可以找到一個猜測的答案。
2. 使用Substitution Method（數學歸納法）作驗證（證明我猜測的答案是正確的）

**1. 畫出一棵Recursion-Tree，找到猜測的答案。**



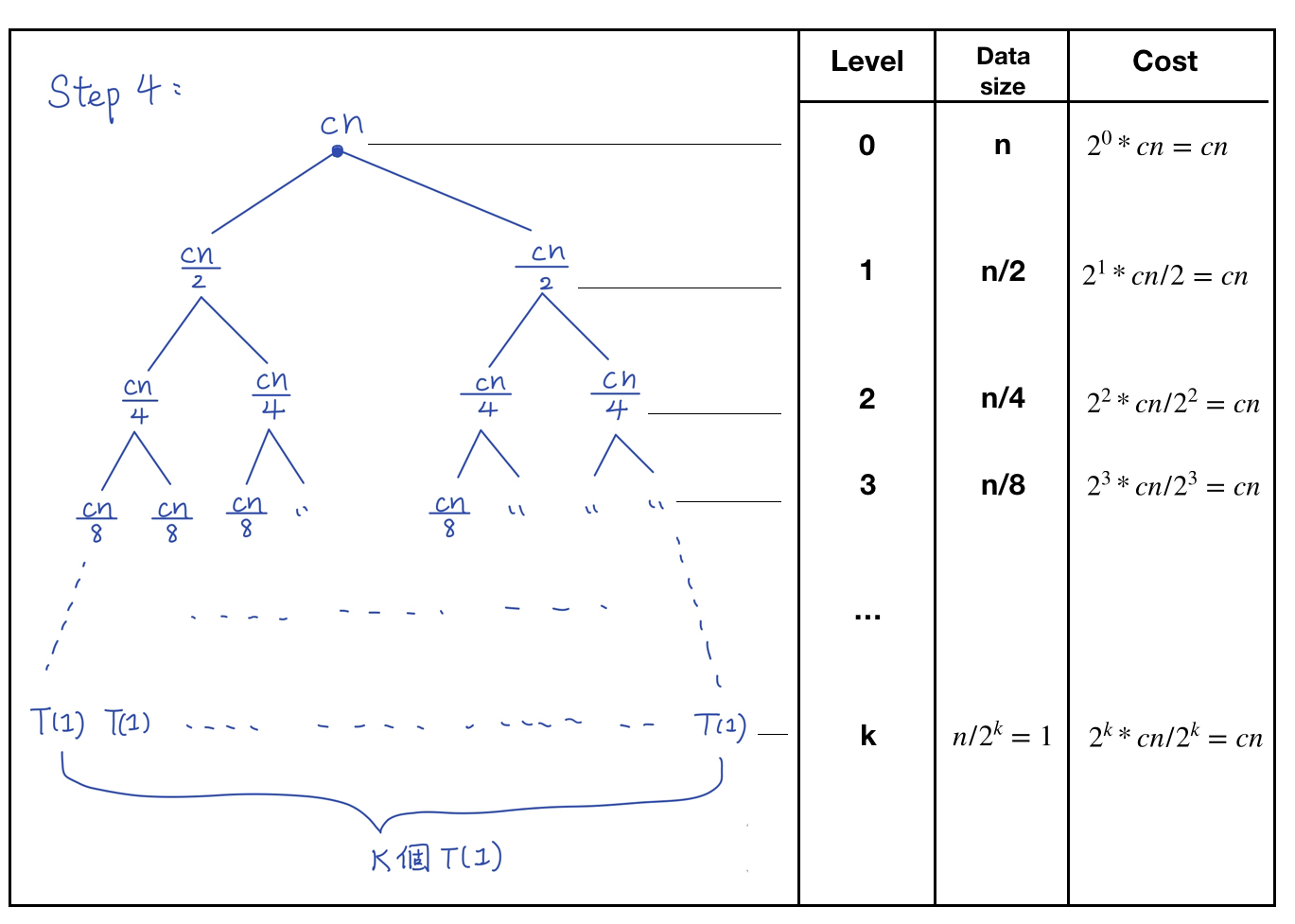
圖一



圖二

細心的讀者在這邊可能會發現原本**遞迴公式的 Θ(n)怎麼被 cn 給替換掉了**。

因為Θ(n)在畫Tree的時候，不方便表示，所以我們用一個cn來做替換。( cn = Θ(n)，cn最接近的時間複雜度即是Θ(n) )



圖三

*Cost代表的是，每一層合併所需要的時間。*

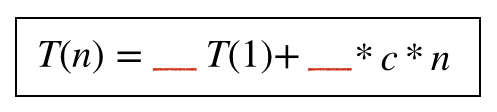
Step4所代表的是整個遞迴的狀態

當我們Merge Sort 不斷地拆成2個子問題，子問題又再拆成2個子問題，總有一天會讓問題最後只剩下一個子問題（可以想成是一個長度為10的陣列，不斷砍半再砍半，最後一定會砍到只剩一個數字的時候）

所以在圖上的k，表示Merge Sort砍半砍到第 k 次的時候，會砍到只剩一個數字，**根據定義，剩的那一個數字時間複雜度為 Θ(1)**

那我們現在想要知道要在什麼情況之下，T(1)才能用公式推導出來（如下圖

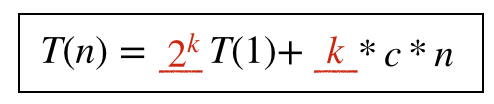
＊如果我可以推導到 T(1)的情況時，代表前面的情況都已經包含在內了



由圖三得到，在2\*k次時，可以得到T(1)，

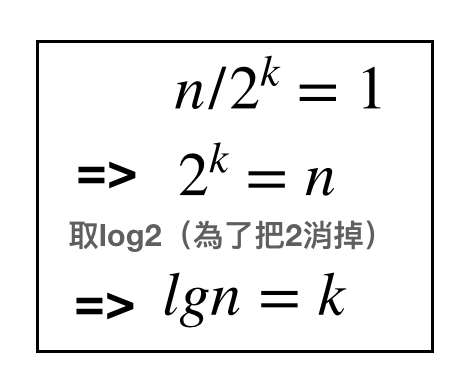
而後面的**c\*n**則是要看總共做了多少次的合併？

*Ans: 從圖三看到每一層的 Cost皆為 cn，然後總共有 k 層，得到 k \* c \* n*

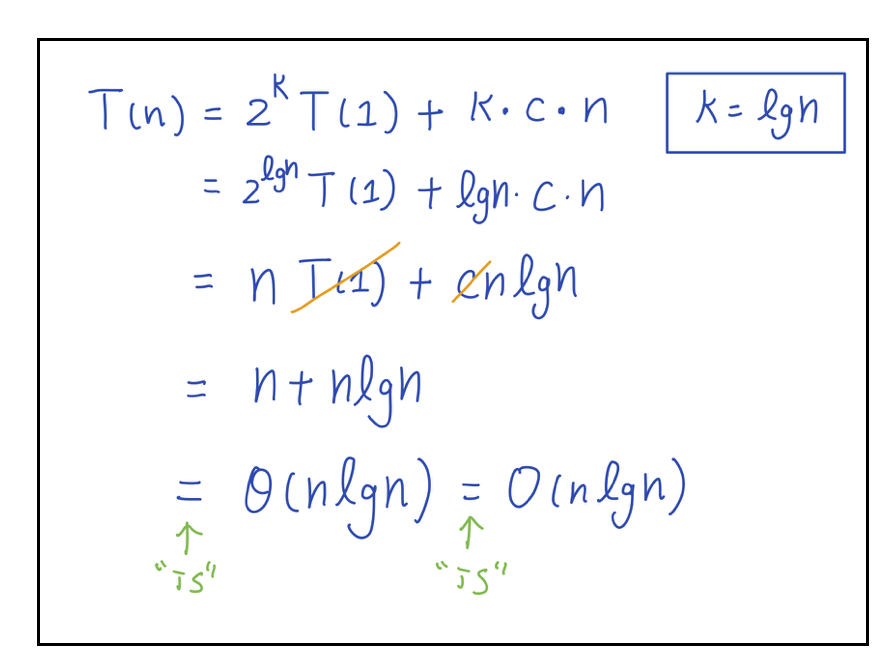


**這時另一個問題又出來了，那k是多少呢？**

*從圖三得到：****n/(2^k) = 1****(原本的資料 n不斷砍半再砍半，經過 k次的砍半後，最終只剩一個)*



最後整合所有的資訊：



終於得到我們猜測的答案了，再來要來驗證他的正確性。（這邊用了蠻大的篇幅去介紹什麼畫Tree，因為要講的東西很多，所以把它拆成小部分小部分去介紹，希望大家有耐心地把他看完）

**2. 使用Substitution Method（數學歸納法）作驗證（證明我猜測的答案是正確的）**

*＊＊ 數學歸納法驗證步驟 ＊＊*

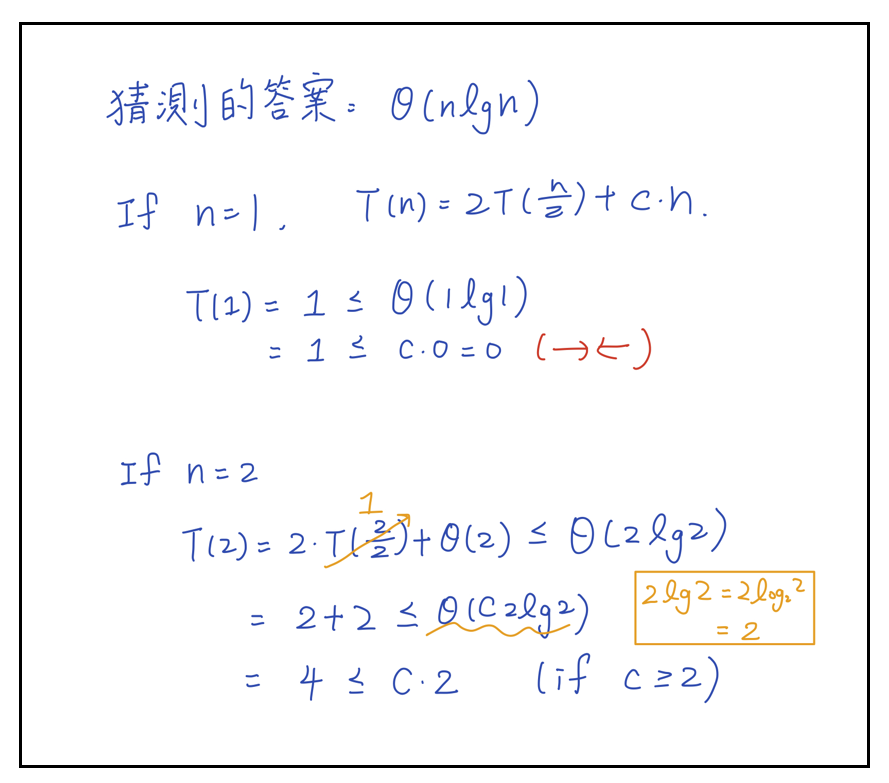
*(1) Induction Base：****找到從某一個開頭是正確的 (True)。***

*(2) Induction Hypothesis：****假設 < n 之中間部分也是正確的 (True)。***

*(3) Induction Step：****利用(1), (2)是正確(True)，來證明 n也是正確(True)***

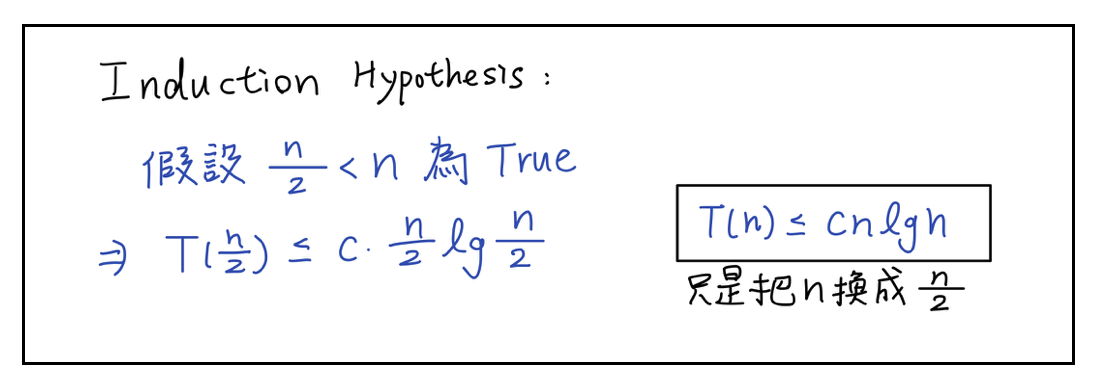
*上述證明稱為****強數學歸納法 ( Strong Induction )****。*

(1) Induction Base：找到從某一個開頭是正確的 (True)。

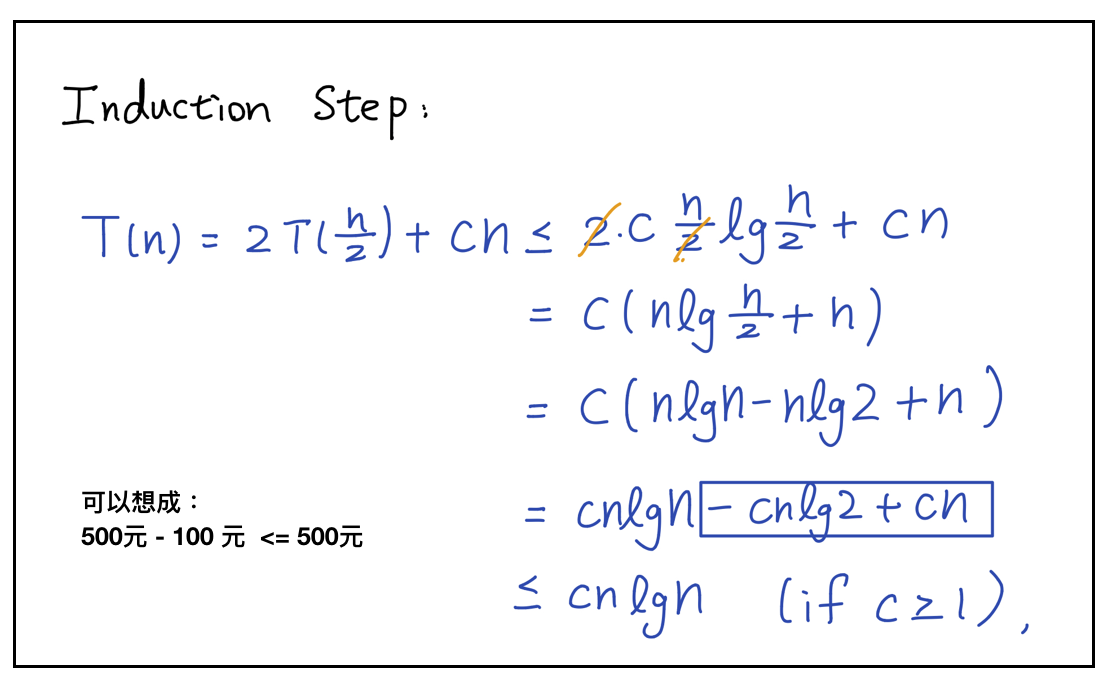


找到 n = 2 是正確的 (True)。

(2) Induction Hypothesis：假設 < n 之中間部分也是正確的 (True)。



(3) Induction Step：利用(1), (2)是正確(True)，來證明 n 也是正確(True)



由數學歸納法證明我們的猜測是對的。（完成作答！）

沒想到寫方法一就花了我這麼多時間，所以方法二就等下次再來寫吧～

如有錯誤還請各位多多指教，也希望這篇文章能幫助到在遞迴關係式感到迷惘的朋友，那我們就下次再見吧。

[More from Sharon Peng](https://mycollegenotebook.medium.com/?source=post_page-----f6d51a462394--------------------------------)

Follow

一起精進程式能力吧！！

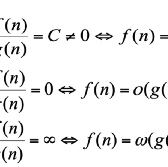
[Apr 29, 2021](https://mycollegenotebook.medium.com/%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6-%E5%87%BD%E6%95%B8%E9%96%93%E7%9A%84%E6%AF%94%E8%BC%83-3e9ce0788f0f?source=post_page-----f6d51a462394----0----------------------------)

**[時間複雜度 — 函數間的比較](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?source=post_page-----f6d51a462394----0----------------------------)**

[希望上次所提到的內容，各位讀者都有暸解到其中的精髓。 1. 幫助函數之間比較 — 數學公式與技巧 2. 簡單的例題（不同函數之間的時間複雜度的比較） 分析函數間的關係，目的在於要學會知道自己寫出來的程式，是不是真的可以使用（不會讓使用者等太久），或是當有兩個人用程式解決相同問題，老闆要怎麼知道誰的比較好，所以要靠今天的章節，來證明「我的演算法比較好！」。 而函數 …](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?source=post_page-----f6d51a462394----0----------------------------)

[Time Complexity](https://medium.com/tag/time-complexity?source=post_page-----f6d51a462394---------------time_complexity-----------------)

[5 min read](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?source=post_page-----f6d51a462394----0----------------------------)

[[](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?source=post_page-----f6d51a462394----0----------------------------)](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?source=post_page-----f6d51a462394----0----------------------------)

Share your ideas with millions of readers.

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

[Apr 29, 2021](https://mycollegenotebook.medium.com/%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6-%E6%BC%B8%E9%80%B2%E5%87%BD%E6%95%B8-397ca19cdc4c?source=post_page-----f6d51a462394----1----------------------------)

**[時間複雜度 — 漸進函數](https://mycollegenotebook.medium.com/%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6-%E6%BC%B8%E9%80%B2%E5%87%BD%E6%95%B8-397ca19cdc4c?source=post_page-----f6d51a462394----1----------------------------)**

[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。 Ome …](https://mycollegenotebook.medium.com/%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6-%E6%BC%B8%E9%80%B2%E5%87%BD%E6%95%B8-397ca19cdc4c?source=post_page-----f6d51a462394----1----------------------------)

[Time Complexity](https://medium.com/tag/time-complexity?source=post_page-----f6d51a462394---------------time_complexity-----------------)

[9 min read](https://mycollegenotebook.medium.com/%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6-%E6%BC%B8%E9%80%B2%E5%87%BD%E6%95%B8-397ca19cdc4c?source=post_page-----f6d51a462394----1----------------------------)