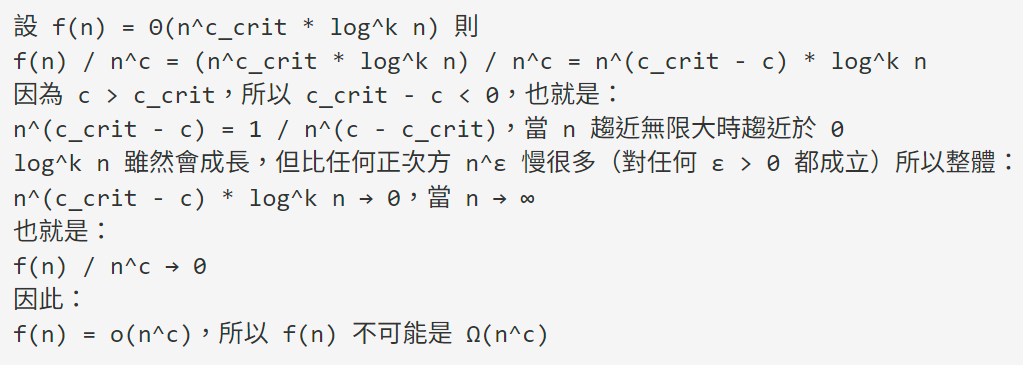
**🧩 題目一：改寫演算法找第 3 大元素**

**原始目標**：找最大值  
**改寫目標**：找第 3 大值  
**Hint**：

* 把原本只回傳最大值的 solve() 改成回傳「前三大」的
* 最後只取其中第三大的那個即可

**📐 題目二**

****

**🧮 題目三：用 Master Theorem 分析 $k = 5$ 的 Merge Sort 時間複雜度**

**基本資訊**：

* 把問題分成 5 個子問題 ⇒ a = 5
* 每個子問題規模是 n/5 ⇒ b = 5
* 合併成本是 O(n) ⇒ f(n) = n

T(n)=5T(5n​)+Θ(n) a=5 b=5

ccrit​=log5​5=1 n^c\_crit = n^1 = n f(n) = Θ(n)

f(n) = Θ(n^c\_crit \* log^k n)，這裡 k = 0

T(n) = Θ(n^c\_crit \* log^{k+1} n) = Θ(n log n)

**🔁 題目四：修改 Merge Sort，合併時統計出現次數**

**輸入範例**：[3, 2, 3, 5, 2, 3, 4, 5, 6, 22]  
**輸出範例**：[{2, 2}, {3, 3}, {4, 1}, {5, 2}, {6, 1}, {22, 1}]



**Lab 02. 分治法**

分治法是一種將問題拆解為兩個或多個相同或相關類型的子問題，直到達到簡單狀態（即你能直接知道答案，通常情況下，你可以使用一組 if-else 條件來獲得答案）。

當我們設計使用分治法的演算法時，我們需要多次解決相同類型的問題（原始問題和子問題）。因此，當我們實現使用分治法的函數時，該函數通常會遞歸地調用自己來解決子問題。其偽代碼可能如下所示：

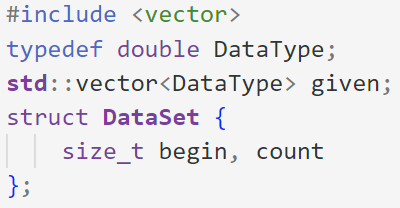


現在，我們可以嘗試使用分治法來解決問題。假設我們要設計一個演算法來查找無序集合中的最大元素。我們可以按以下步驟解決問題：

1. 拆解問題 - 將給定的集合分成 $k$ 個不重疊的區塊。
2. 解決子問題 - 找到每個區塊中的最大元素。
3. 聚合子問題的結果 - 從第二步的集合中找出最大的元素。

我們可以按照上述偽代碼來實現這個算法。

為了提高速度，將參數作為切片傳遞，也就是傳遞合法範圍，而不是複製整個子集。返回類型取決於我們要查找的內容，即給定數據的類型。



現在，假設我們選擇 $k = 2$。我們可以按以下方式實現這個演算法：



**題目：重寫上述算法以查找第三大元素。**

提示：你可以嘗試找到前三大的元素，而不僅僅是找第三大的。

**Master Theorem**

如果我們有一個確定性的程序且不含遞歸調用，它通常會在多項式時間內運行。通常情況下，答案應該是 $n^k$，其中 $n$ 是輸入的大小，$k$ 是最大嵌套深度。但當我們有遞歸調用時，這種情況不適用。

現在，假設我們有一個函數實現了分治算法，像上面的偽代碼一樣。我們可以將時間複雜度寫成T(n)=g(n)+a×T(b/n​)+h(n)

其中：

* g(n) 表示將問題拆分為子問題所需的時間複雜度。
* a 是將問題分成的子問題的數量。
* 每個子問題的大小為 n/b。
* h(n) 表示合併子問題結果所需的時間複雜度。

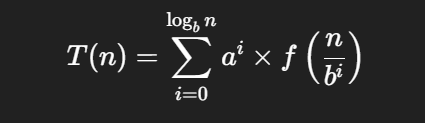
定義 f(n)=g(n)+h(n)，我們得到：

T(n)=a×T(b/n​)+f(n)

展開這個方程，會得到：

T(n)=a2×T(n/b2)+f(n)+a×f(n/b​)

繼續展開後，我們得到：



最終展開的結果是：

T(n)=f(n)+a×f(n/b​)+⋯+alogb​n×f(1)

根據上述遞迴式，我們發現整體的時間複雜度由兩個部分決定：「遞迴成本（a 的 log\_b 次方）」與「其他成本 f(n)」。  
其中： a^(log\_b n) = n^(log\_b a)

我們將這個次方部分稱為 **臨界指數（critical exponent）**，記作： c\_crit = log\_b a

臨界指數在分析時間複雜度時非常有幫助：

* 當 f(n) = O(n^c)，且 c < c\_crit 時，T(n) = Θ(n^c\_crit) T(n) = Θ(n^(log\_b a))
* 當 f(n) = Ω(n^c)，且 c > c\_crit 時，T(n) = Θ(f(n)) = Ω(n^c) T(n) = Θ(f(n))
* 其他情況是：f(n) = Θ(n^c\_crit \* (log n)^k)，其中 k ≥ 0，此時 T(n) = Θ(n^c\_crit \* (log n)^(k+1)) T(n) = Θ(n^(log\_b a) \* (log n)^(k+1))

**題目：當屬於上述第三種情況時，請嘗試證明：不可能存在 c > c\_crit 使得 f(n) = Ω(n^c)**

**分治法的優點**

由於子問題彼此之間互不相干，我們可以**同時處理**這些子問題。  
這樣實際花費的時間會比原始的 T(n) = a\*T(n/b) + f(n) 少得多。

假設我們擁有無限的計算能力，就可以一次同時解出 a 個子問題，  
此時時間近似為：

T(n) = T(n/b) + f(n)

若我們一次最多只能同時跑 k 個程序，則實際時間會接近：

T(n) = ceil(a/k) \* T(n/b) + f(n)

MapReduce 是一個典型的分治式計算模型，用於 Hadoop 平台上處理大量資料。

**分治法的典型範例：合併排序（Merge Sort，以升序排序）**

* **基本情況**：如果只有一個或沒有元素，則視為已排序
* **分割**：將未排序陣列分成 k 個部分，讓每個部分大小盡量平均
* **合併**：比較每個子陣列開頭的元素，取最小值加入到已排序陣列中

**題目：請使用 Master Theorem 分析當 k = 5 時，Merge Sort 的時間複雜度**

**題目：若輸入資料中存在重複的元素，請修改演算法使其能夠合併相同數值並統計出現次數**範例輸入：[3, 2, 3, 5, 2, 3, 4, 5, 6, 22]  
範例輸出：[{2, 2}, {3, 3}, {4, 1}, {5, 2}, {6, 1}, {22, 1}]