該如何使用二分搜尋法?

(1) 判斷題目是否能使用二分搜尋法

最明顯的特徵，若題目中出現「最小的最大值」或「最大的最小值」，八九不離十為二分搜尋法。剩下的，就只能靠自己判斷了。

(2) 找出二分搜尋法的模型

當使用二分搜尋法的時候，問題就會轉換成判斷某個值是否可行。這時得設計一套方法來測試。同時也得找出搜尋區間。

(3) 初始化搜尋區間

確認二分搜的模型後，再來就是確認，有沒有可能發生找不到答案的情況。如果有，初始搜尋區間就得做一些調整。詳見 lower_bound.cpp 內的詳解。

建立二分搜尋法的步驟，大致上都脫離不了上述三點，以上。

題目名稱中，開頭是 Online Judge 的名稱，後面是題號。像是 POJ 2976，就是指 PKU Online Judge 的編號 2976 題目。以下給出常用的四個 Online Judge 網址：

POJ : www.poj.org

LeetCode: https://leetcode.com/

UVa Online Judge: https://uva.onlinejudge.org/

Aizu Online Judge: http://judge.u-aizu.ac.jp/onlinejudge/

若你是這類題目的新手，建議你從 LeetCode 開始。因為你程式有錯，它會告訴你錯在哪，對於剛起步的新手而言，除錯會比較容易，挫折感會比較小。而且它的題目都有標籤，若想練習二分搜尋法的題目，只需要搜尋”Binary Search”的 tag，就可以找到二分搜尋法的題目了。

而對 C/C++還不熟的同學，建議等你們對程式語言有一定的掌握以後，再來挑戰這類的題目，會比較好。不過之後的禮拜五晚上，我們只會講演算法和資料結構的應用，不會接觸程式的實作。對程式語言的掌握度，理論上應該是不會影響到對演算法的理解度。就算你不會程式語言，依然歡迎你來聽聽。

我會把實作中，會用到的實用的函數、類別、或是樣板程式碼，放到網路上，讓想學的人自行學習。

下禮拜會講到貪心法，與其說它是演算法，倒不如說是一種概念。因為它並沒有制式的步驟。在石頭那題裡，我們把兩兩距離小於 d 的石頭移除，就是貪心法的展現。欲知後事如何，且聽下回分解。