Syllabus

-Binary\_Search

Binary Search

前言:

甲和乙在玩一種猜數字遊戲，甲為出題方，在心中默想1~20以內的其中一個數，乙為猜數字方，他必須要盡可能猜出甲的數字。為了避免遊戲無法進行，每次猜完後甲都會給一個提示，提示乙猜的數字比他的題目大還是小。那麼，問題來了，請問乙最多猜幾次一定可以猜到?(不考慮最少的原因是因為有機率性)

當然，乙可以選擇一個最簡單的方法:隨便亂猜。從1猜到20一定可以猜中，而且不會太麻煩。不過如果今天範圍變成了1到100、1到1000呢?一個一個猜一定會死掉。因此，我們可以在每一步都猜正中間的數，利用甲給的提示逐步縮小範圍，很快就可以把答案包夾出來。聽不懂我的意思?沒關係，看看下面的例子

舉例而言，假設甲的題目是13，乙的範圍是1~20，於是他先猜正中間的數字10，得到了”比10大”的提示，接著將範圍修正為”11~20”，依舊是猜正中間的數字15，發現題目比15小，將範圍再縮小成11~14，下一步猜12，得到比12大的提示，範圍變成12~15，最後猜13，得到答案。這樣子總共才猜4次，比一個一個猜的13次少了非常多。而這，就是二分搜尋法的精髓。

如果你閱讀文字感到吃力，我們可以將以上的文字整理成下表，這邊我們用x代表題目:

|  |  |  |  |
| --- | --- | --- | --- |
| 原範圍 | 猜的數字 | 提示 | 修正後範圍 |
| 1~20 | 10 | x>10 | 11~20 |
| 11~20 | 15 | X<15 | 11~14 |
| 11~14 | 12 | x>12 | 13~14 |
| 13~14 | 13 | X=13 |  |

如果你對數學稍微有點感覺，你就會發現每次取中間值都是以無條件捨去的方式(如範圍11~20的中間值理應是(11+20)/2=15.5，這邊卻是15)，這是因為程式語言會默認使用無條件捨去，以上的文字只是將程式做的事情表達出來而已。

我們知道他會比較快，但是會快到多少呢?如果用瞄線法搜尋，最差你需要O(n)的時間複雜度；但是以二分搜尋法，最壞最壞也就是O(log2n)而已，這邊要注意程式語言中除非特別設定，否則對數的底數都是2，故接下來都不會特別把底數寫出來。

那要如何證明它的複雜度僅有O(logn)?因為每一次都是將整個搜尋的陣列剖半，假設你最壞需要切x次，那就代表你將”切兩半”這個動作進行了x次，因此可以得到2x=n，自然而然就可以發現x=logn了(再次提醒，底數是2不是10)。

了解了他的複雜度以及原理後，下一節我們可以來實作二分搜。

實作:

首先，你必須要了解這是一個遞迴的概念。那麼遞迴要怎麼寫呢?

根據前面提到的內容，遞迴首先需要一個終止條件，不然無限運行到系統分配的記憶體極限後會被強制終止，因此我們先設定遞迴條件”如果左界在右界的右邊(很顯然不合邏輯吧)，那麼就停止遞迴”，