資訊工程學系　　黃右萱(0416323)

物件導向程式設計暨第六次作業報告

問題簡述：

二分搜尋樹是用於有序資料搜尋非常有用的資料結構，然而一個不平衡的二分搜尋樹會大幅度地降低搜索效率，所以針對這個問題，設計出了一種自平衡二分搜尋樹，讓二分搜尋樹在幾個基礎運算後能夠維持其平衡性，進而增進搜索的效率，其中一種自平衡樹為 AVL 樹，本次目標要實作 AVL 樹中的搜索、迭代器移位、左旋以及右旋運算。

二分搜尋樹單次搜索的時間約正比於樹的高度，令樹高為 h，結點數量為 n，它進行單次搜索的時間複雜度是 O(h)，而自平衡二分搜尋樹單次搜索效率則能控制在 O(log n)，因為它內部的平衡機制確保 $\mathrm{h} \approx \log n$ 的關係恆然成立。

實作方法：

一、 搜索(Find)：

搜索由樹的根部開始，由於元素間符合嚴格弱序關係，每次搜索朝向期待地大小方向搜索，若目標結點比目前所在結點還大，則往樹的左下子樹尋找，若比目前所在結點還小，則朝向右下子樹尋找，直到無法繼續尋找或已尋得目標結點為止。

二、 迭代器移位：

迭代器移位目的在尋找結點大小關係中，下一個比目前結點小或大的結點，考慮要尋找下一個較大結點(++運算)，若右下子樹非空，向右下子樹尋找一次，接著向左下子樹尋找直到抵達樹的葉部；若右下子樹不存在，向親節點尋找，第一個比原結點大的親節點即為所尋目標；若此二者皆無法成功找到滿足條件的節點，則代表下一個較大結點不存在，於是迭代器內回傳空指標，若實作(--)運算實作方法相同，但左右方向顛倒即可。

三、 左右旋：

左右旋必須在不打亂樹的序關係為前提下，改變樹的結構，其中牽涉到約莫四個結點關係的重分配和高度修正，因此我另外宣告了幾個暫存指標，把每個需要重分配的節點位置記錄下來，直接針對這些位置進行關係的重分配，按照左右旋標準的寫法實作就不會出問題了。

實作感想：

我認為這次的作業非常有意思，或許是所有作業裡頭最有意思的一份作業，出題者竟然為了一次作業將整顆 AVL 樹刻了出來，同時這次作業能夠讓多數同學了解到自平衡樹幾個非常重要的自平衡手段。