Binary Search Tree功能說明

Def :

左子樹所有Node小於Root ; 右子樹所有Node大於Root

輸入的資料皆需與Root比，比Root小往左子樹放，比Root大往右子樹

左子樹與右子樹也都是B.S.T

1. 查詢search

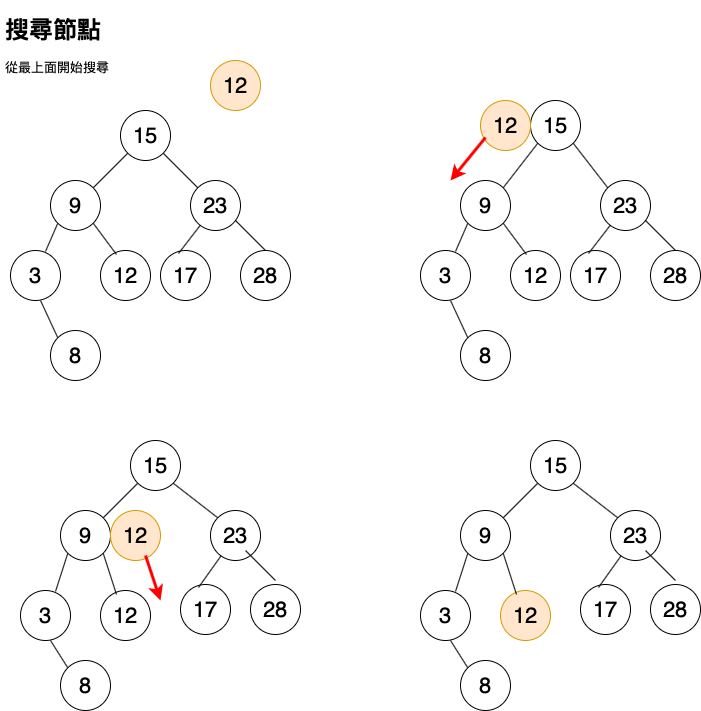
如果在B.S.T找到資料，return Data ; 否則return not found

“target = root” : 找到資料

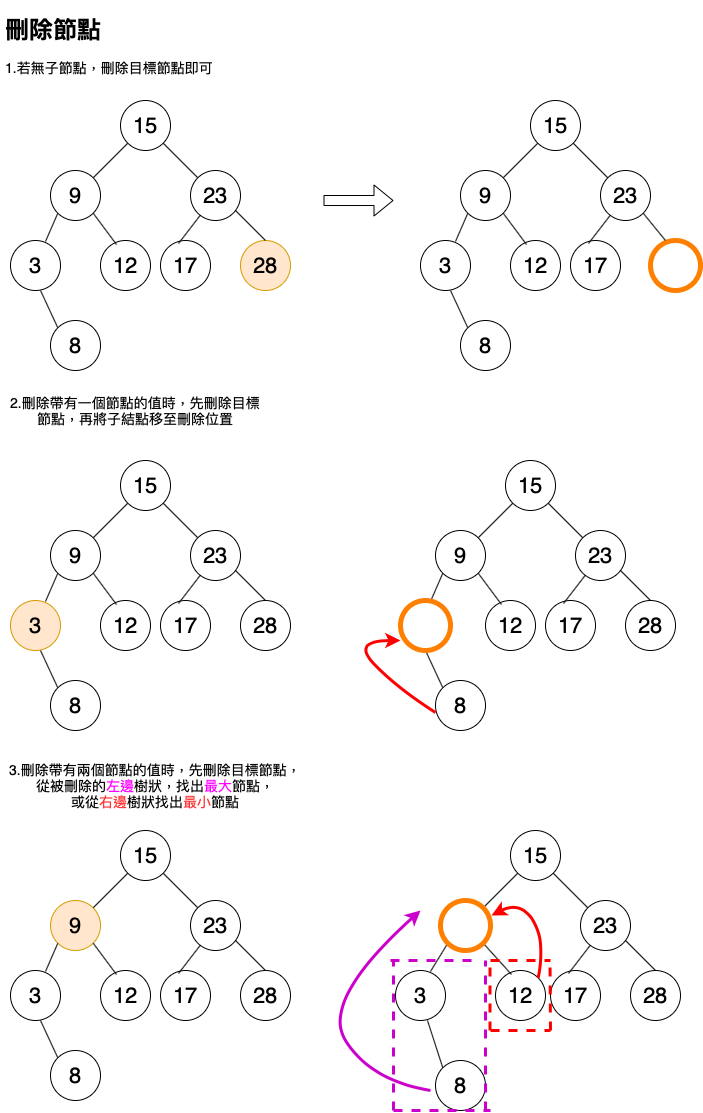
“target < root” : 往左子樹找

“target > root” : 往右子樹找

利用遞迴重新呼叫：self.search，例如在第2層級的數字，將被比較兩次。



1. 刪除delete
2. 若無子節點，刪除目標節點即可。
3. 刪除帶有一個子節點時degree = 1，先刪除目標節點，再將子結點移至刪除位置。
4. 刪除帶有兩個子節點時degree = 2，先刪除目標節點，再從被刪除的左子樹，找出最大值取代刪除值，或從右子樹找出最小值取代。



1. 插入insert (與搜尋類似)
2. root不存在，則直接新增
3. root存在，接著root與val比較，產生三種可能

root.val > val，val往左比較

(2-1)若無左節點，直接插入

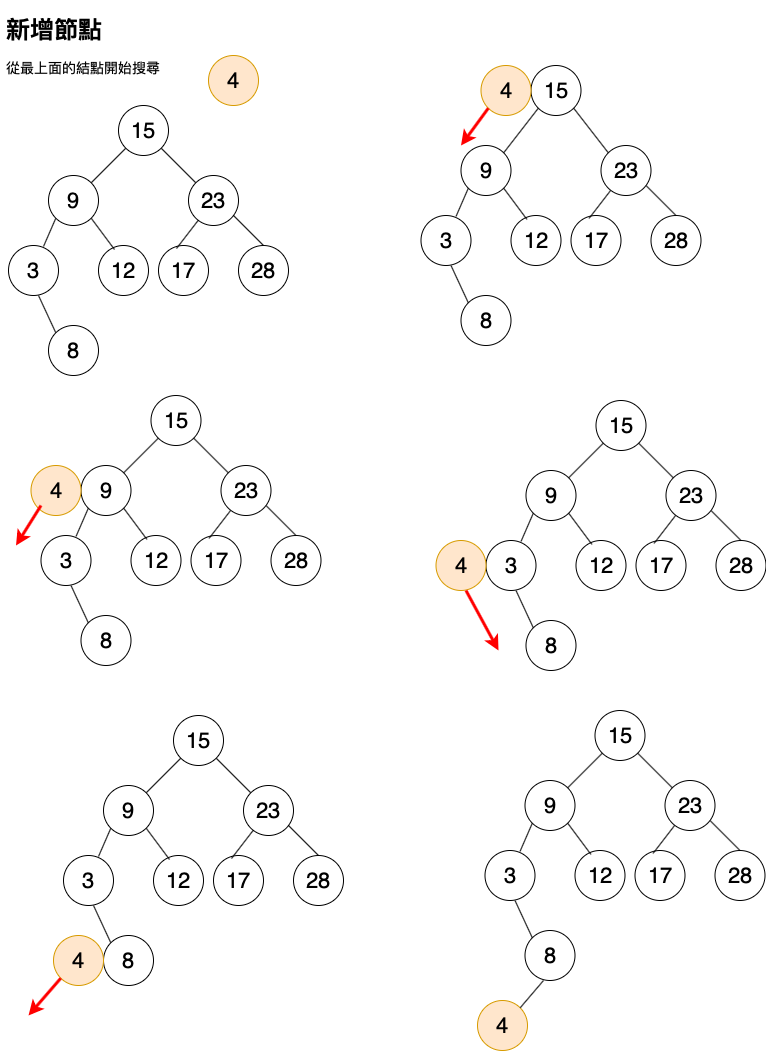
(2-2) 若有左節點，利用遞迴重新呼叫，”self.insert(root.left,val)”

(3.)root.val < val，val往右比較

(3-1) 若無右節點，直接插入

(3-2) 若有右節點，利用遞迴重新呼叫，”self.insert(root.right,val)”

(4.)root.val = val，欲插入的val已存在，則插入到root.val的左節點



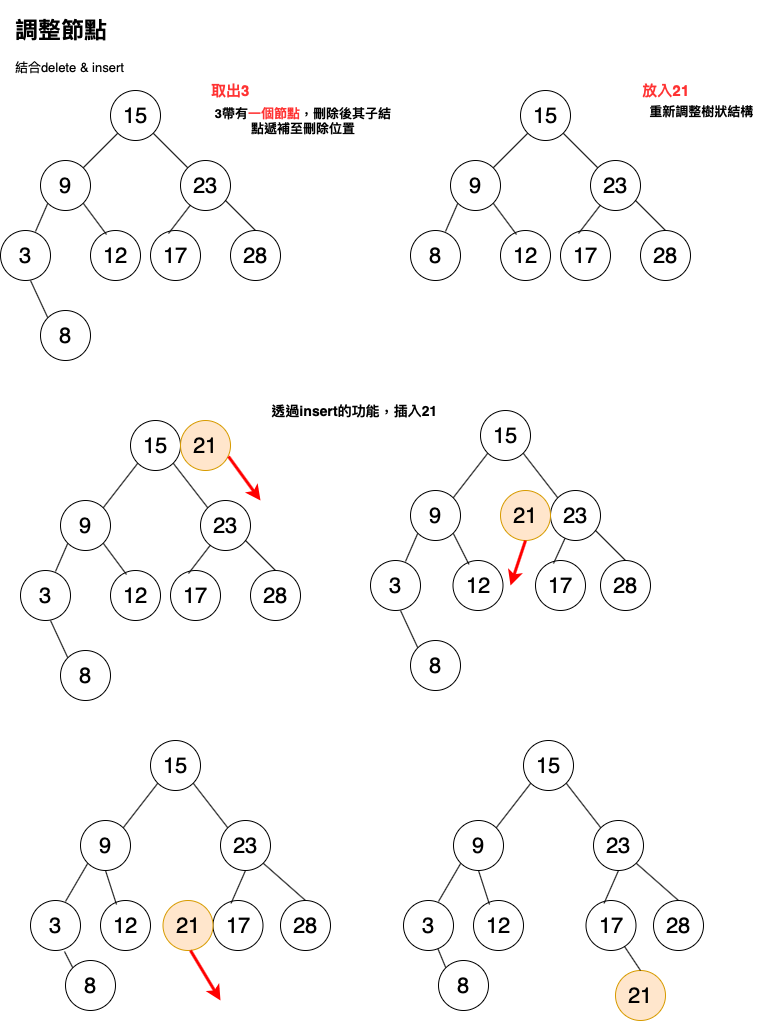
1. 修改modify

先利用delete功能，再利用insert功能

(1.) delete，需判斷無節點、有一個節點、有二個節點的狀況

(2.)重新整理樹狀後

(3.) insert，需與root比較，決定向右走(val > root.val)、向左走(val <= root.val)



參考資料：1.圖書館線上書：圖解資料結構 使用python

2.圖書館：演算法圖鑑 石田保輝著

3. http://alrightchiu.github.io/SecondRound/binary-search-tree-introjian-jie.html