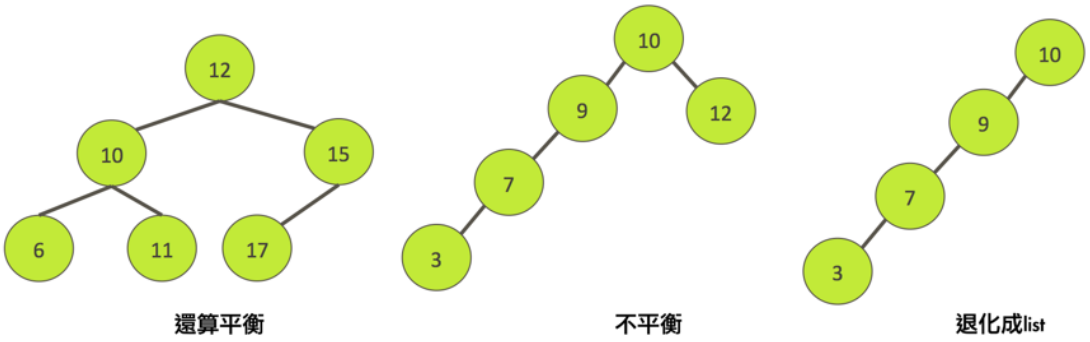
紅黑樹原理

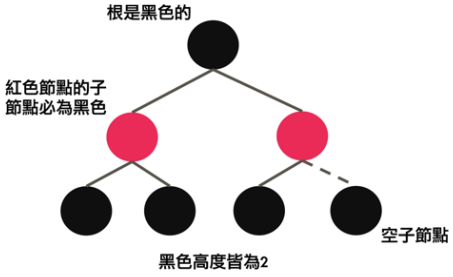
樹的平衡：所謂樹的平衡指的是，樹中每個節點的左邊後代的數目應該與其右邊後代的數目大致相等(不用完全一樣多)。
對於用隨機數構成的二元樹，一般來說是大致平衡的，但是對於有順序的資料，就可能導致二元樹極度的不平衡了．在最極端的情況下，甚至會退化成 list，此時的時間複雜度會退化到 O(n)，而不再是平衡樹的 O(logn)了。



紅黑樹(R-B Tree)：紅黑樹是增加了某些特性的二元搜尋樹，它可以保持樹的大致平衡．主要的思路為：在插入或刪除節點的時候，檢查是否破壞了樹的某些特徵，若破壞了，則進行糾正，進而保持樹的平衡。

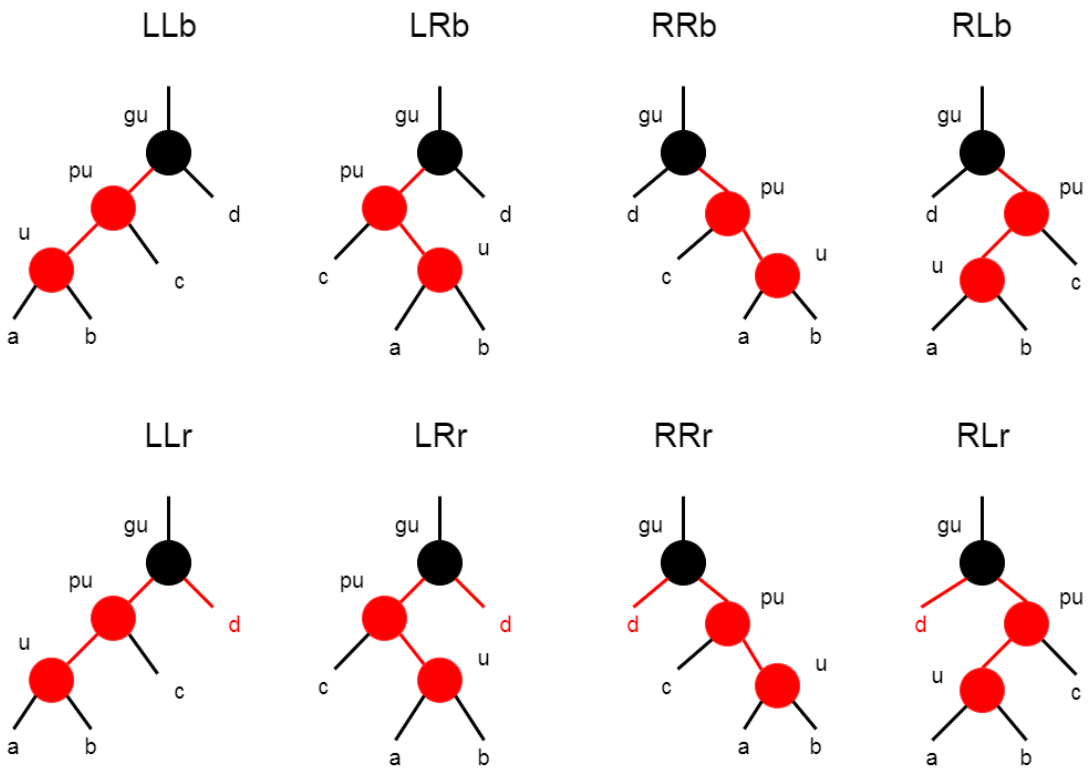
紅黑樹的規則(紅黑規則)：

1. 每個節點不是紅就是黑的
2. 根總是黑色的
3. 若節點是紅色的，則其子節點必為黑色，反之不必為真(亦即若節點是黑色，則其子節點可為紅也可為黑)，這條規則其實也是在說明就垂直方向來看，紅色節點不可以相連
4. 每個空子節點都是黑色的：所謂的空子節點指的是，對非葉節點而言，本可能有，但實際沒有的那個子節點．譬如一個節點只有右子節點，那麼其空缺的左子節點就是空子節點
5. 從根節點到葉節點或空子節點的每條路徑(簡單路徑)，必須包含相同數目的黑色節點(這些黑色節點的數目也稱為黑色高度)



紅黑樹實作

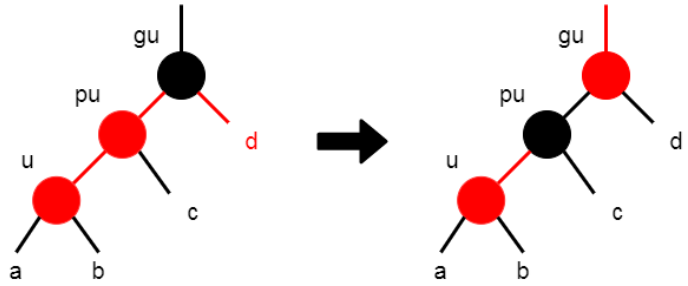
在建立紅黑樹的實作上需考慮 8 種情況: LLb、LRb、RLb、RRb、LLr、LRr、RLr、RRr。

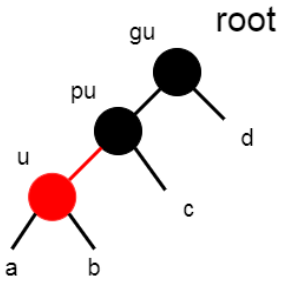


1. 以四個節點為單位作考慮: 祖父節點(gu)、父節點(pu)、子節點(u)、叔節點(d)。分成兩種情況: r、b，r 代表叔節點為紅色；b 代表叔節點為黑色。
2. 前兩個大寫代表發生連續兩個紅色節點連在一起的方向。
3. 如果無子節點則將該虛擬的子節點視為黑色。
4. 每次增加節點時都增加紅色節點。
5. 每增加一個節點就要重新檢查一次是否發生那 8 種情形。

r:

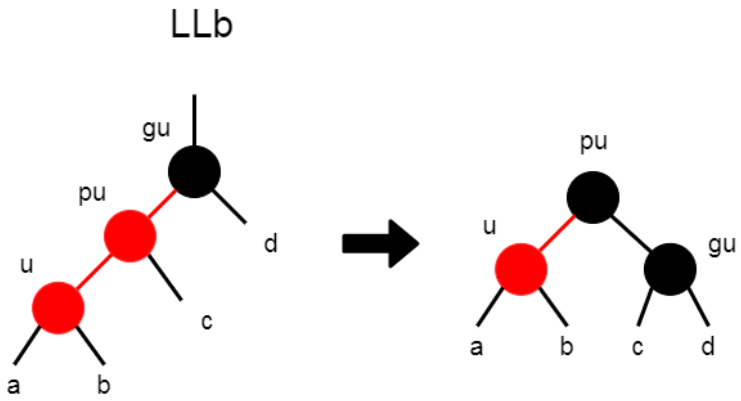
祖父節點的顏色與父節點的顏色互換，叔節點由紅色變黑色。此時祖父節點變為紅色，如果祖父節點為根節點，則違反定義，因此在將其變成黑色；如果祖父節點不為根節點，則往上檢查，將祖父節點是為新的子節點，檢查是否發生連續兩個紅色相連的情況。





b:

如果是 LL(前兩個大寫)則先做 LL(前兩個大寫) rotation，然後祖父節點的顏色與父節點的顏色互換。



參考資料

1. https://yotsuba1022.gitbooks.io/data-structure-note/content/1.4.3-red-black-tree.html
2. https://www.youtube.com/watch?v=4WjwmHeKa1Q&t=4s
3. https://www.youtube.com/watch?v=fP1taNiz7ZI