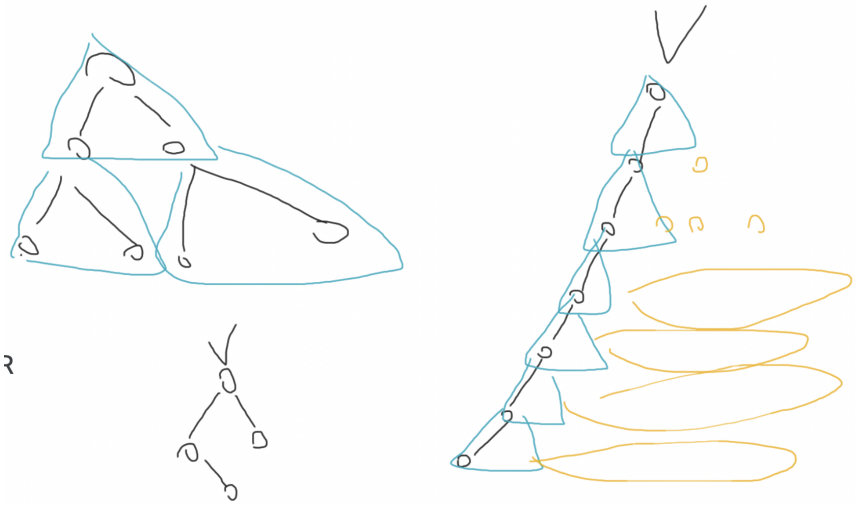
Phill-DS-0320

https://youtu.be/a854lhHq_a0

AVL tree - Adelson V L

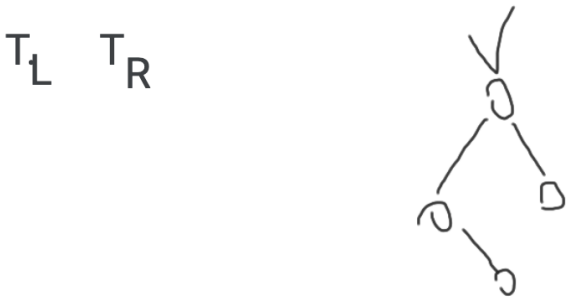
高度平衡的二元搜尋樹 → efficiency (不能有空隙, 計算量會增加)



$T \in AVL$

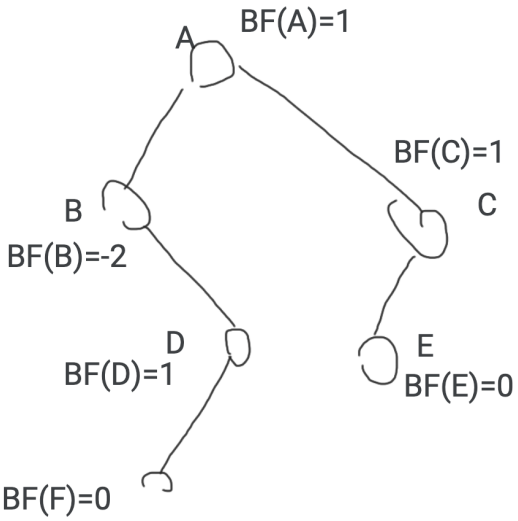
$T_L, T_R \in AVL$

$|h_L - h_R| <= 1$ → balanced



balance factor 平衡因子 BF → 針對某一個節點的高度計算結果 → BF(n) , n→ node

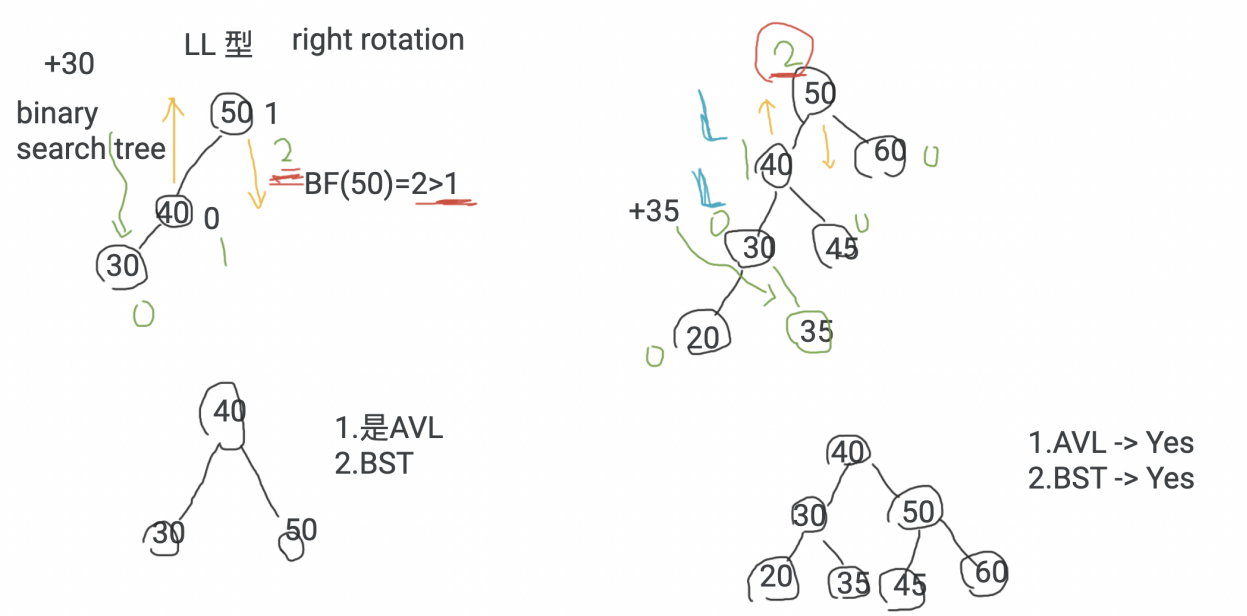
$|BF(n)| \leq 1$ → -1, 0, 1



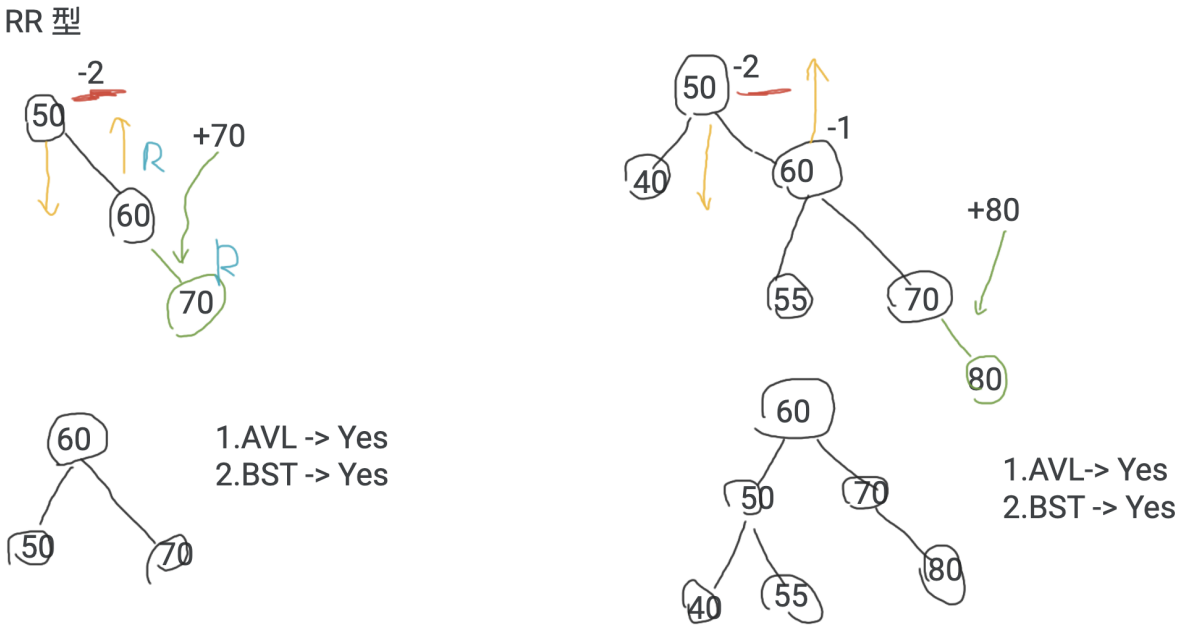
AVL tree 加入 node

- 針對四種不同的 types → 作 re-balanced

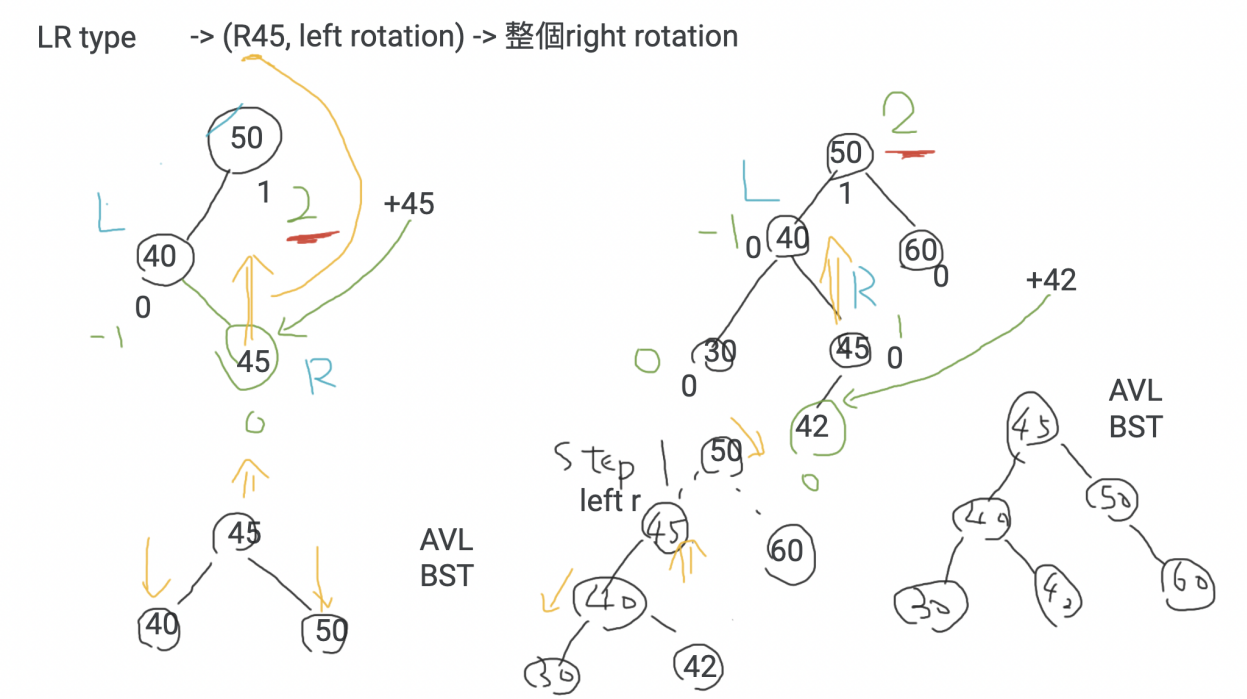
LL type



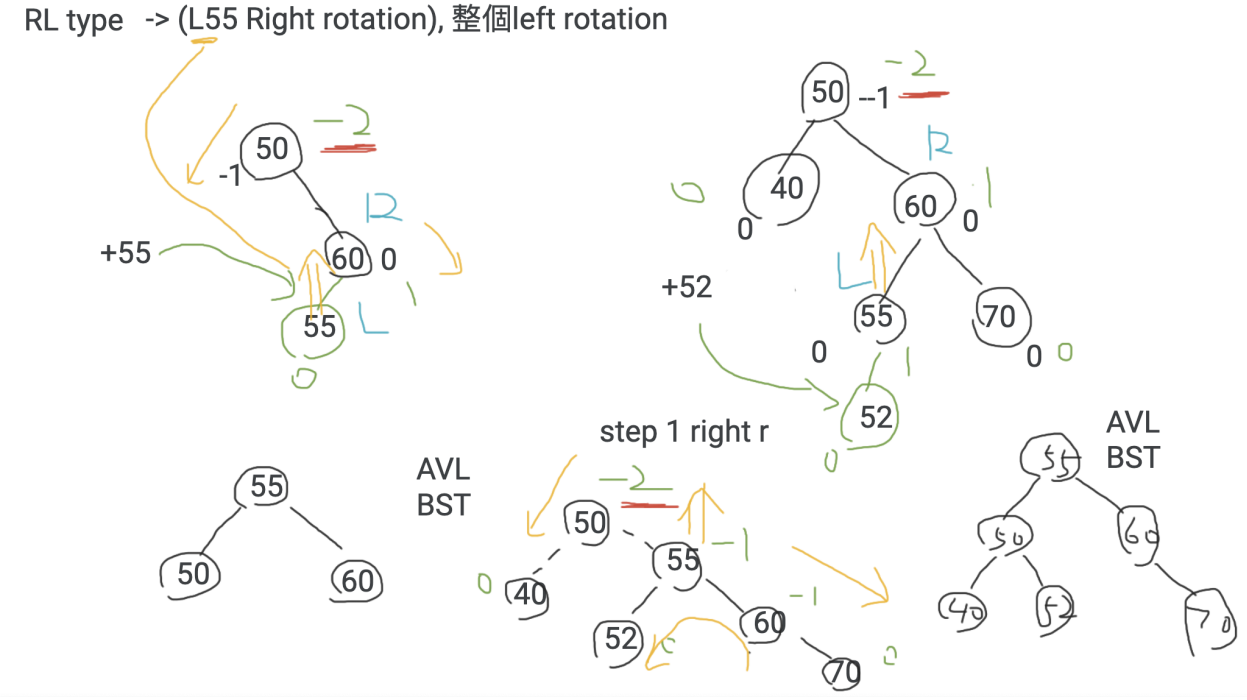
RR type → left rotation



LR type

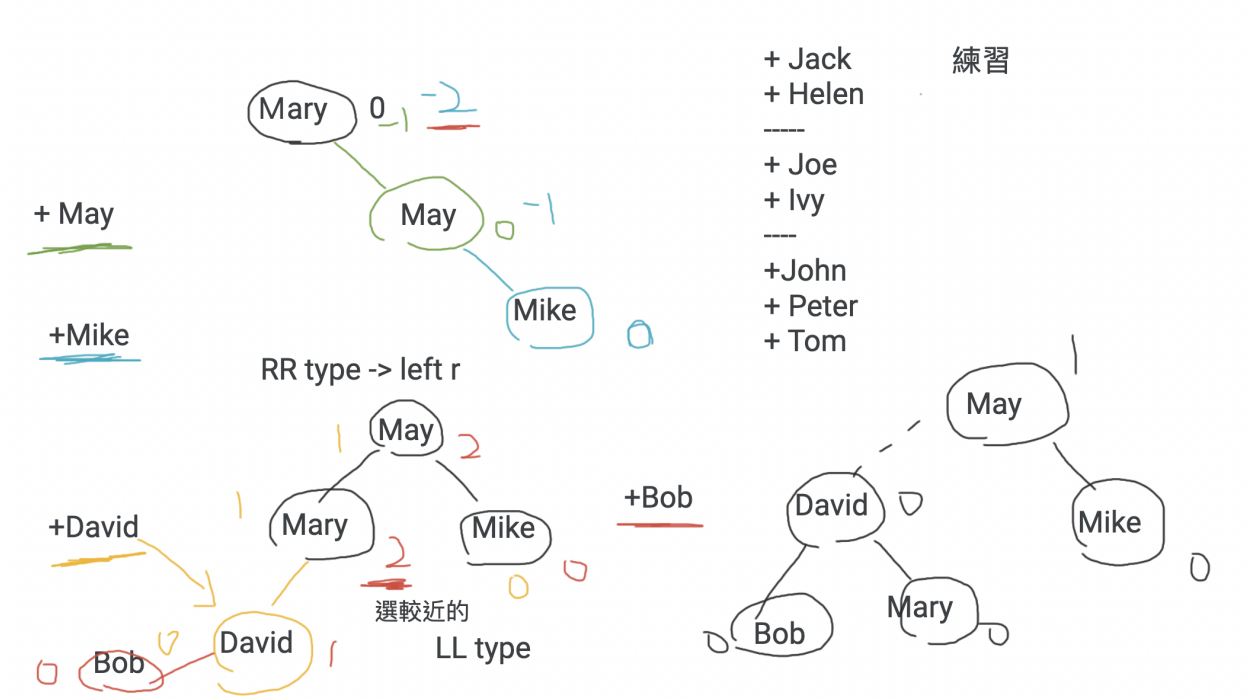


RL type



例子

字串 AVL → BST 原則大小 alphabetlic order 大小



tldraw

A free and instant collaborative diagramming tool.

； https://www.tldraw.com/r/V5WtU61rVHub3ZHeUXEg8?v=11,4492,1920,1004&p=page

實作

```
#include <iostream>
using namespace std;

class Node{
  public:
    int key;
    Node* left;
    Node* right;
    int height;
```

```
    Node(int k):key(k), left(nullptr), right(nullptr), height(1)
};

class AVLTree{
  public:
    Node* root;

    AVLTree():root(nullptr){}

    int height(Node* n){
      if(n==nullptr) return 0;
      return n->height;
    }

    int getBalance(Node* n){
      if(n==nullptr) return 0;
      return height(n->left) - height(n->right);
    }

    void updateHeight(Node* n){
      if(n!=nullptr){
        n->height = max(height(n->left), height(n->right))+1;
      }
    }

    Node* rightRotation(Node* y){
      Node* x=y->left;
      Node* T2 = x->right;

      x->right = y;
      y->left = T2;

      updateHeight(y);
      updateHeight(x);

      return x;
```

```
    }
};

int main()
{
    cout << "Hello, World!";
    return 0;
}
```