# 题目

输入两棵二叉树A和B，判断B是不是A的子结构。(约定空树不是任意一个树的子结构)

B是A的子结构， 即 A中有出现和B相同的结构和节点值。

例如:

给定的树 A:

     3

    / \

   4   5

  / \

 1   2

给定的树 B：

   4

  /

 1

返回 true，因为 B 与 A 的一个子树拥有相同的结构和节点值。

示例 1：

输入：A = [1,2,3], B = [3,1]

输出：false

示例 2：

输入：A = [3,4,5,1,2], B = [4,1]

输出：true

限制：

0 <= 节点个数 <= 10000

# 分析

// hasSubStructure函数是判断如果A,B根节点相同，B是不是A的子结构

bool hasSubStructure(TreeNode\*A, TreeNode\*B)

{

if (!B) //递归结束条件1：A的一个节点B的对应位置没有,可以认为是子结构

return true;

if (!A || A->val != B->val) //递归结束条件2：B的一个节点A的对应位置没有 / A,B对应位置节点值不同，此时必然不可能是子结构

return false;

return hasSubStructure(A->left, B->left) && hasSubStructure(A->right, B->right); //返回值：继续在对应位置递归判断

}

bool isSubStructure1(TreeNode<T> \*A, TreeNode<T> \*B)

{

if (!A || !B) //特殊判断

return false;

// 根节点相同的话直接进入比较，根节点不相同看B是不是A的左/右子树的子结构

return hasSubStructure(A, B) || isSubStructure1(A->left, B) || isSubStructure1(A->right, B);

}