**二叉查找树（BST）具备什么特性呢？**

**1.左子树上所有结点的值均小于或等于它的根结点的值。**

**2.右子树上所有结点的值均大于或等于它的根结点的值。**

**3.左、右子树也分别为二叉排序树。**

**好处：同二叉搜索树思想，查找次数最大为二叉树的的高度；**

**插入利用类似的方法，通过一层一层比较大小，找到新阶段插入位置。**

**缺陷：普通二叉树，某左子树或右二叉树深度过深，高度不平衡，查找或插入时查询次数过高；**

**平衡二叉树：**

**树的左右两边的层级数相差不会大于1;**

**没有值相等重复的节点;**

**红黑树是一种自平衡的平衡二叉树**

**1.节点是红色或黑色。**

**2.根节点是黑色。**

**3.每个叶子节点都是黑色的空节点（NIL节点）。**

**4 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)**

**5.从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点。**

**变色：**

**为了重新符合红黑树的规则，尝试把红色节点变为黑色，或者把黑色节点变为红色。**

**左旋转：**

**逆时针旋转红黑树的两个节点，使得父节点被自己的右孩子取代，而自己成为自己的左孩子。说起来很怪异**

**右旋转：**

**顺时针旋转红黑树的两个节点，使得父节点被自己的左孩子取代，而自己成为自己的右孩子。**

**1.** **将一个数组生成二叉排序树**

排序，选数组中间的一个元素作为根节点，左边的元素构造左子树，右边的节点构造有子树。

**2.** **红黑树的定义和解释？B树的基本性质？**

红黑树：

性质1. 节点是红色或黑色。  
性质2. 根节点是黑色。  
性质3. 每个叶子结点都带有两个空的黑色结点（被称为黑哨兵），如果一个结点n的只有一个左孩子，那么n的右孩子是一个黑哨兵；如果结点n只有一个右孩子，那么n的左孩子是一个黑哨兵。  
性质4 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)  
性质5. 从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点。

B树：

1.所有非叶子结点至多拥有两个儿子（Left和Right）；

2.所有结点存储一个关键字；

3.非叶子结点的左指针指向小于其关键字的子树，右指针指向大于其关键字的子树；

**3.什么是平衡二叉树?**

左右子树都是平衡二叉树，而且左右子树的深度差值的约对值不大于1。

**4、BFS DFS 二叉树遍历**

https://github.com/kqxu1992/common-codes/blob/main/dfs\_bfs.cpp

**5、获取二叉树的最大深度**

void maxTreeDepth(node\* head, int depth, int max)

{

if (!head) return;

if (max < depth) max = depth;

if (head->left) maxTreeDepth(head->left, depth+1, max);

if (head->right) maxTreeDepth(head->right, depth+1, max);

}

int maxTreeDepth2(node\* head)

{

if (!head) return 0;

std::stack<node\*> st;

int depth = 0, max = 0;

while (head || !st.empty()) {

if (head) {

st.push(head);

++depth;

max = max < depth ? depth : max;

head = head->left;

}

else {

head = st.top();

st.pop();

--depth;

head = head->right;

}

}

return max;

}