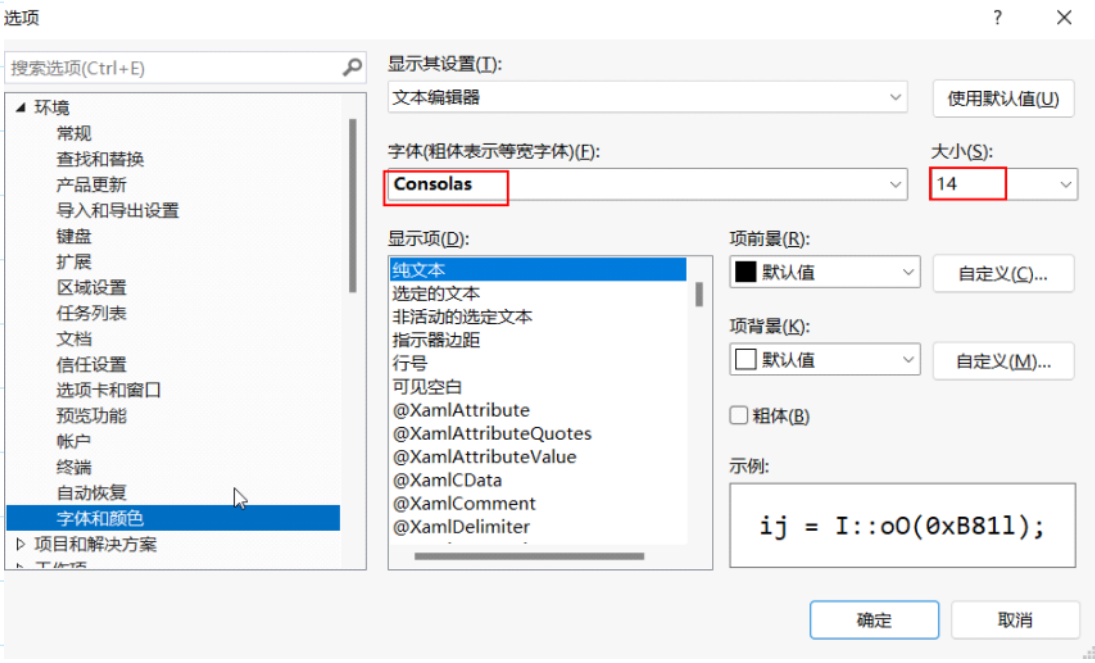
VS的字体设置

2024年5月7日 15:49

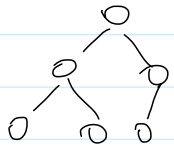
工具 → 选项



二叉搜索树

2024年5月7日 9:22

#1. 定义



⇒ 每一个结点的度 (degree) ≤ 2

#2. 二叉树的特殊形态.

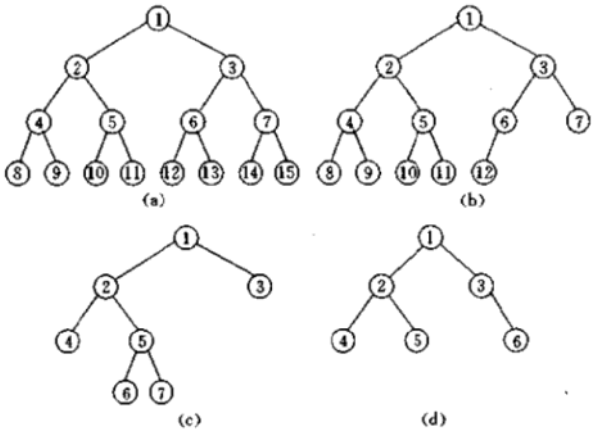
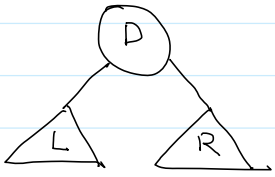


图 6.4 特殊形态的二叉树

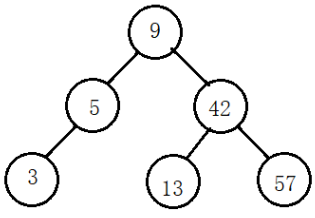
(a) 满二叉树; (b) 完全二叉树; (c)和(d)非完全二叉树。

满二叉树 ⊆ 完全二叉树.

#3. BST (Binary Search Tree)

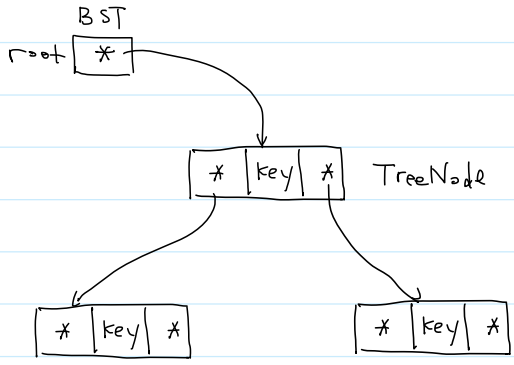


L < D < R

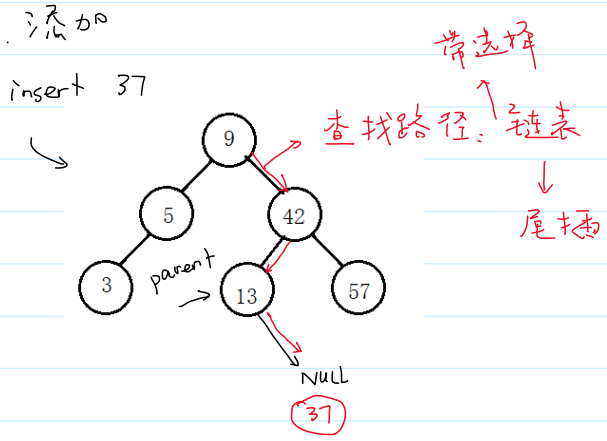


#4. 实现BST

```c
typedef int K;

typedef struct tree_node {
    K key;
    struct tree_node* left;
    struct tree_node* right;
} TreeNode;

typedef struct {
    TreeNode* root;
} BST;

// API
BST* bst_create();
void bst_destroy(BST* tree);

void bst_insert(BST* tree, K key);
TreeNode* bst_search(BST* tree, K key);
void bst_delete(BST* tree, K key);

void bst_preorder(BST* tree);
void bst_inorder(BST* tree);
void bst_postorder(BST* tree);
void bst_levelorder(BST* tree);
```



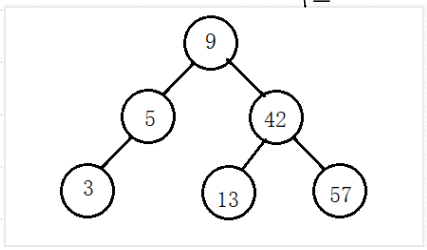
a. 添加



b. 查找

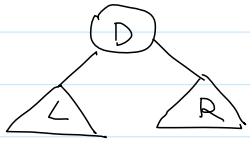
d. 遍历

二维



一维

(1) 深度优先遍历



(√)

L D R 中序遍历

L R D 后序遍历

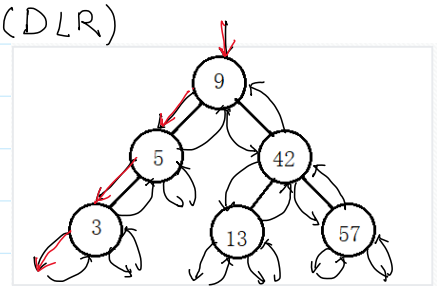
D L R 先序遍历

~~D R L~~

~~R L D~~

~~R D L~~

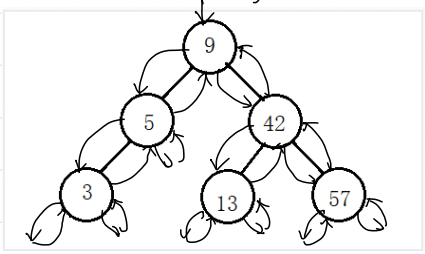
(DLR)



9 5 3 42 13 57

O(n)

(L D R)

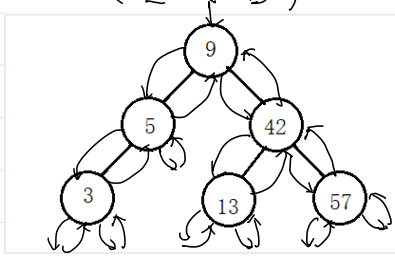


3 5 9 13 42 57

O(n)

二叉排序树

(L R D)



3 5 13 57 42 9

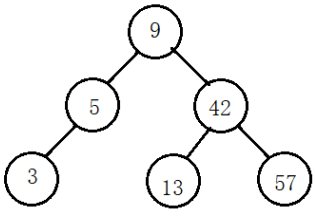
O(n)

(2) 广度优先遍历 (层次遍历)

9

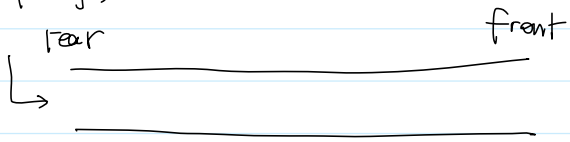
5 42

3 13 57



9 5 42 3 13 57

(队列)



① 将根结点入队列

② 判断队列是否为空

空：结束

非空：出队列.

遍历该结点.

判断该结点是否有左孩子

(9) (5) (42) (3) (13) (57)

不变式：当上一层所有结点出队列时，

队列中放的是下一层所有的结点

遍历该结点.

判断该结点是否有左孩子

有：将左孩子入队列

判断该结点是否有右孩子

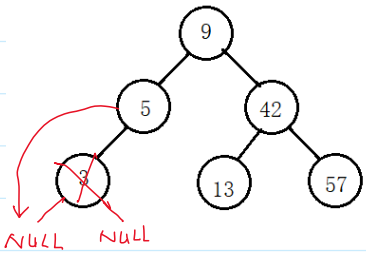
有：将右孩子入队列

BST的删除

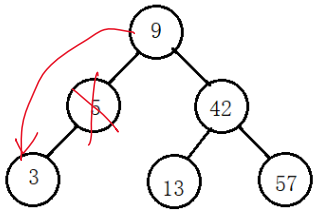
2024年5月7日 15:11

C. 删除

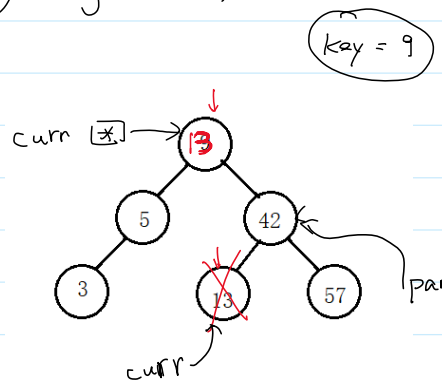
① degree = 0



② degree = 1



③ degree = 2



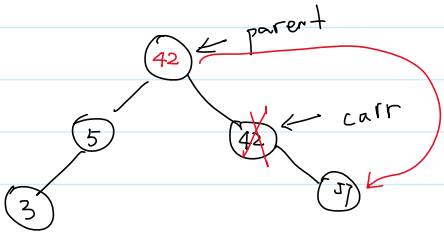
① 左子树的最大结点

② 右子树的最小结点

⇓

一定没有左孩子 → 规化成①、②

特例！ 右子树最小结点，就是右子树根结点




```
curr->key = min->key;
parent = p;
curr = min;
```

```
int cmp = curr->key - parent->key;
```

BST的性能分析

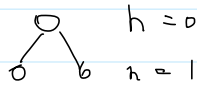
2024年5月7日 16:25

h：树的高度

查找 search: O(h)

插入 insert O(h)

删除 delete O(h)



问题：一棵二叉树，有n个结点，高度的取值范围是多少？

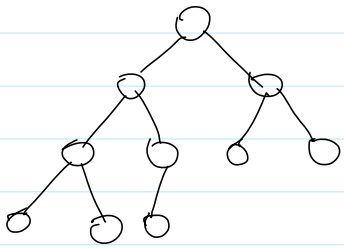
$h \in [\lfloor \log_2 n \rfloor, n-1]$

⇓

一棵二叉树，有n个结点，高度的最低是多少？

⇓ 完全二叉树

一棵有n个结点的完全二叉树，高度h是多少？



0	2^0
1	2^1
2	2^2
⋮	⋮
h	2^h

⇓

一棵高度为h的完全二叉树，其结点数目范围是多少

$$2^h = 1 + 2^0 + 2^1 + \cdots + 2^{h-1} \le n < 2^0 + 2^1 + 2^2 + \cdots + 2^h + 1 = 2^{h+1}$$

$$2^h \le n < 2^{h+1}$$

$$\log_2 n - 1 < h \le \log_2 n \quad (\log_2 n - 1, \log_2 n]$$

$h = \lfloor \log_2 n \rfloor$ (向下取整) 下限

缺陷：不能保证 O(lgn) 时间复杂度的插入，删除和查找

平衡二叉搜索树

2024年5月7日 16:43

AVL、平衡：对任意一个结点，左子树和右子树的高度之差不超过1.

定义很严格 ⇒ 高度比较小

每次添加或删除，需要调整的操作多.

红黑树、平衡：整棵树的高度 $O(\lg n)$

定义比较宽松 ⇒ 高度比较高

每次添加或删除，需要调整的操作少.

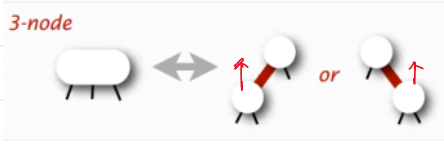
红黑树

2024年5月7日 16:50

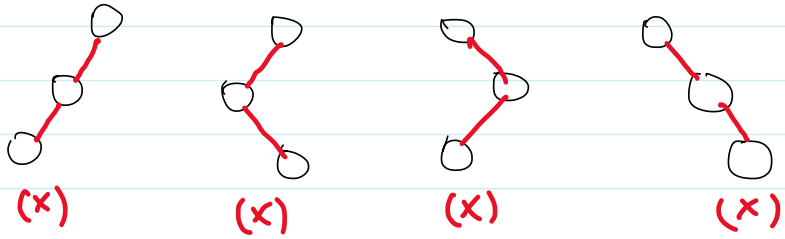
#1. 模型

2-3-4 树 模型 ——— 红黑树 实现

Q1：如何表示 3-node 和 4-node



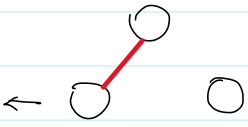
⇒ 控制整棵树的高



⇒ 不能有两条连续的红色边

Q2：“边”是一个逻辑结构，是不存在，如何表示“边”的颜色？

把边的颜色放入孩子结点



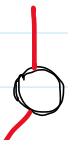
```
typedef struct treenode_s {
    int val;
    bool color; // 表示该节点和父节点之间那条边的颜色
    struct treenode_s* left;
    struct treenode_s* right;
}TreeNode;
```

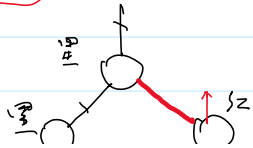
我们来看一看经典教科书(算法导论)对红黑树的定义：

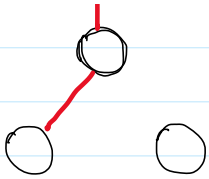
一棵红黑树是满足下面红黑性质的二叉搜索树：

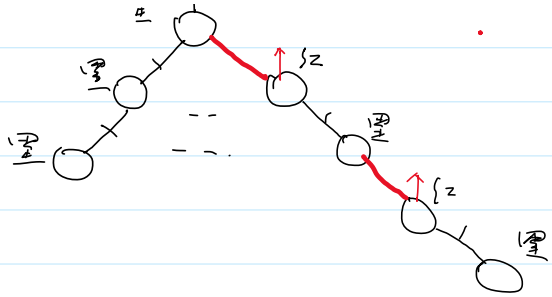
1. 每个结点或者是红色的，或者是黑色的
2. 根结点是黑色的
3. 叶子结点 (Nil) 是黑色的 (注：在算法导论中，叶子结点指得是 NULL 结点)
4. 如果一个结点是红色的，则它的两个子结点都是黑色的 (4-node 只有一种编码方式)
5. 对每个结点，从该结点到其所有后代叶子结点的路径上，包含相同数目的黑色结点。(黑高平衡，2-3-4树是一个完美平衡的树)

2-3-4 树的高度









2-3-4树的高度 ≤ 红黑树高度 ≤ 2 * (2-3-4树的高度)