二叉搜索树

https://github.com/CoderMJLee

http://cnblogs.com/mjios

思考

- 在 n 个动态的整数中搜索某个整数？（查看其是否存在）
- 假设使用动态数组存放元素，从第 0 个位置开始遍历搜索，平均时间复杂度：O(n)

0	1	2	3	4	5	6	7	8	9
31	66	17	15	28	20	59	88	45	56

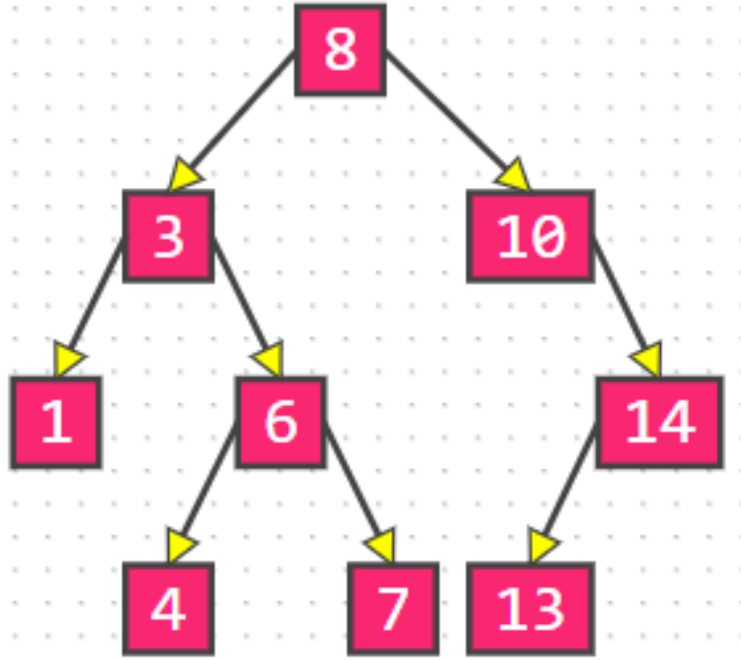
- 如果维护一个有序的动态数组，使用二分搜索，最坏时间复杂度：O(logn)
  - 但是添加、删除的平均时间复杂度是 O(n)

0	1	2	3	4	5	6	7	8	9
15	17	20	28	31	45	56	59	66	88

- 针对这个需求，有没有更好的方案？
  - 使用二叉搜索树，添加、删除、搜索的最坏时间复杂度均可优化至：O(logn)

二叉搜索树（Binary Search Tree）

- 二叉搜索树是二叉树的一种，是应用非常广泛的一种二叉树，英文简称为 BST
  - 又被称为：二叉查找树、二叉排序树
  - 任意一个节点的值都大于其左子树所有节点的值
  - 任意一个节点的值都小于其右子树所有节点的值
  - 它的左右子树也是一棵二叉搜索树

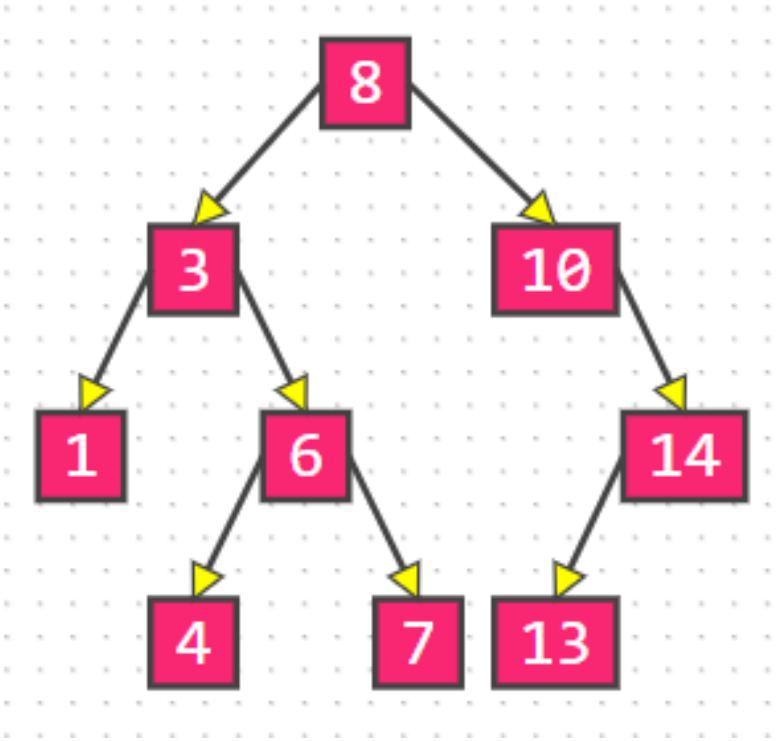


- 二叉搜索树可以大大提高搜索数据的效率

- 二叉搜索树存储的元素必须具备可比较性
  - 比如 `int`、`double` 等
  - 如果是自定义类型，需要指定比较方式
  - 不允许为 `null`

二叉搜索树的接口设计

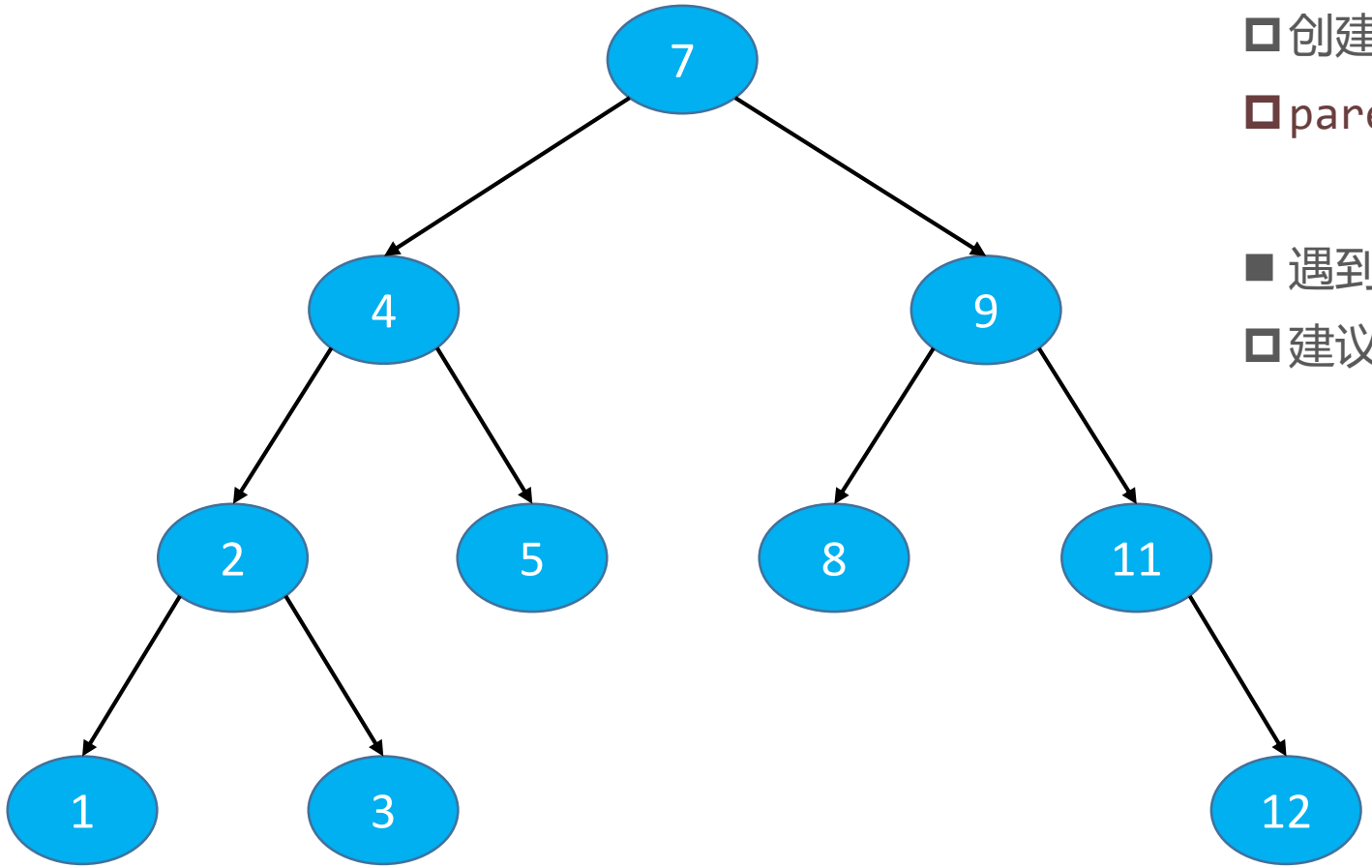
- `int size()  // 元素的数量`
- `boolean isEmpty()  // 是否为空`
- `void clear() // 清空所有元素`
- `void add(E element) // 添加元素`
- `void remove(E element)  // 删除元素`
- `boolean contains(E element) // 是否包含某元素`



- 需要注意的是
  - 对于我们现在使用的二叉树来说，它的元素没有索引的概念
  - 为什么？

添加节点

- 比如添加12、1



- 添加步骤
  - 找到父节点 parent
  - 创建新节点 node
  - parent.left = node 或者 parent.right = node

- 遇到值相等的元素该如何处理?
  - 建议覆盖旧的值

元素的比较方案设计

1. 允许外界传入一个 Comparator 自定义比较方案

2. 如果没有传入 Comparator，强制认定元素实现了 Comparable 接口

推荐一些神奇的网站

- http://520it.com/binarytrees/
- http://btv.melezinek.cz/binary-search-tree.html
- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
- https://yangez.github.io/btree-js
- https://www.codelike.in

打印二叉树

- 工具：https://github.com/CoderMJLee/BinaryTrees

- 使用步骤
  - 实现 BinaryTreeInfo 接口
  - 调用打印API
    - ✓ BinaryTrees.println(bst);

```
/********** BinaryTreeInfo **********/
@Override
public Object root() {
    // 根节点是谁？
    return root;
}

@Override
public Object left(Object node) {
    // 如何查找左节点？
    return ((Node<E>)node).left;
}

@Override
public Object right(Object node) {
    // 如何查找右节点？
    return ((Node<E>)node).right;
}

@Override
public Object string(Object node) {
    // 如何打印单个节点？
    return ((Node<E>)node).element;
}
/********** BinaryTreeInfo **********/
```

根据元素内容获取节点

```
private Node<E> node(E element) {
    elementNotNullCheck(element);
    Node<E> node = root;
    while (node != null) {
        int cmp = compare(element, node.element);
        if (cmp == 0) return node;
        if (cmp > 0) {
            node = node.right;
        } else {
            node = node.left;
        }
    }
    return node;
}
```

删除节点 – 叶子节点

- 直接删除

☐ `node == node.parent.left`

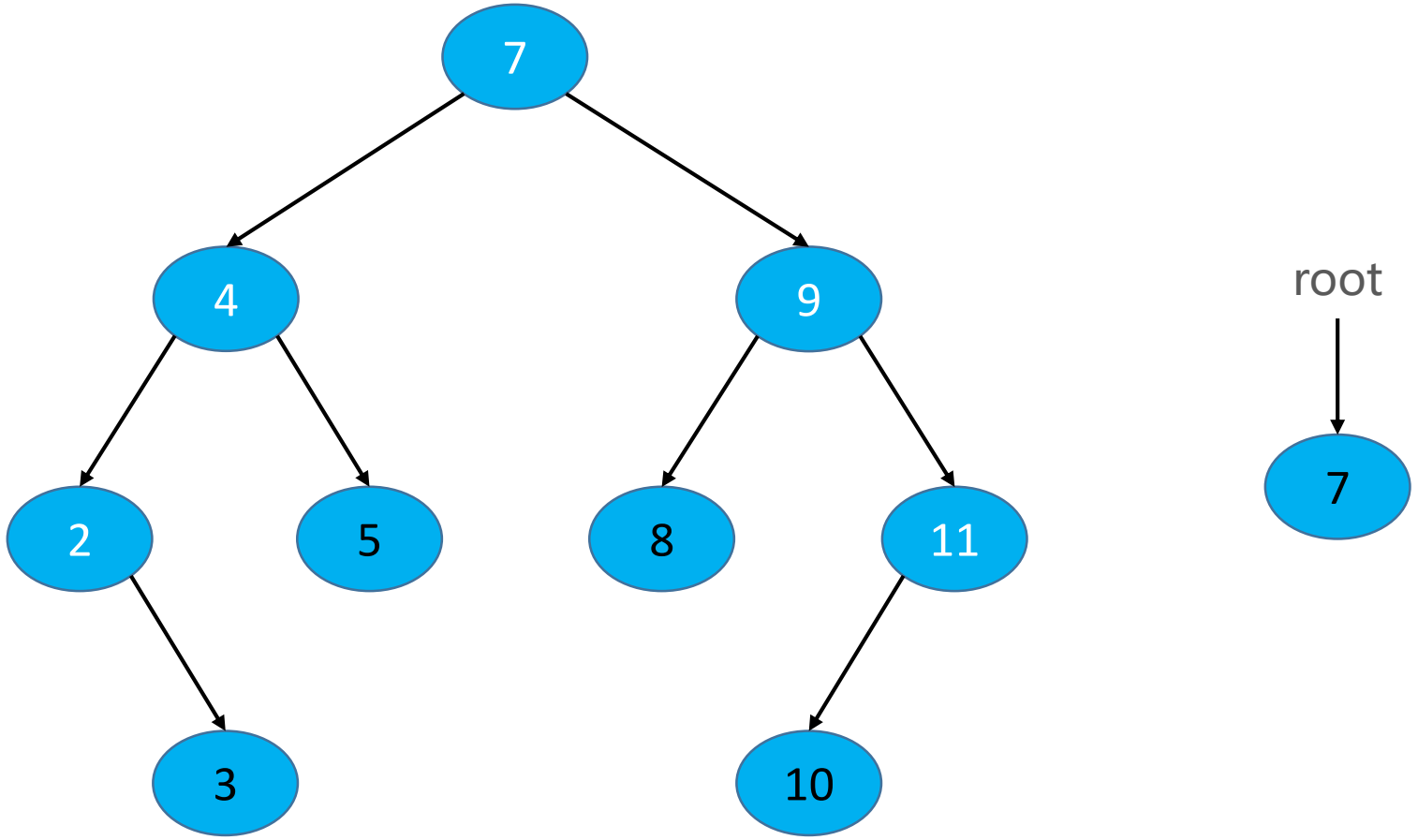
✓ `node.parent.left = null`

☐ `node == node.parent.right`

✓ `node.parent.right = null`

☐ `node.parent == null`

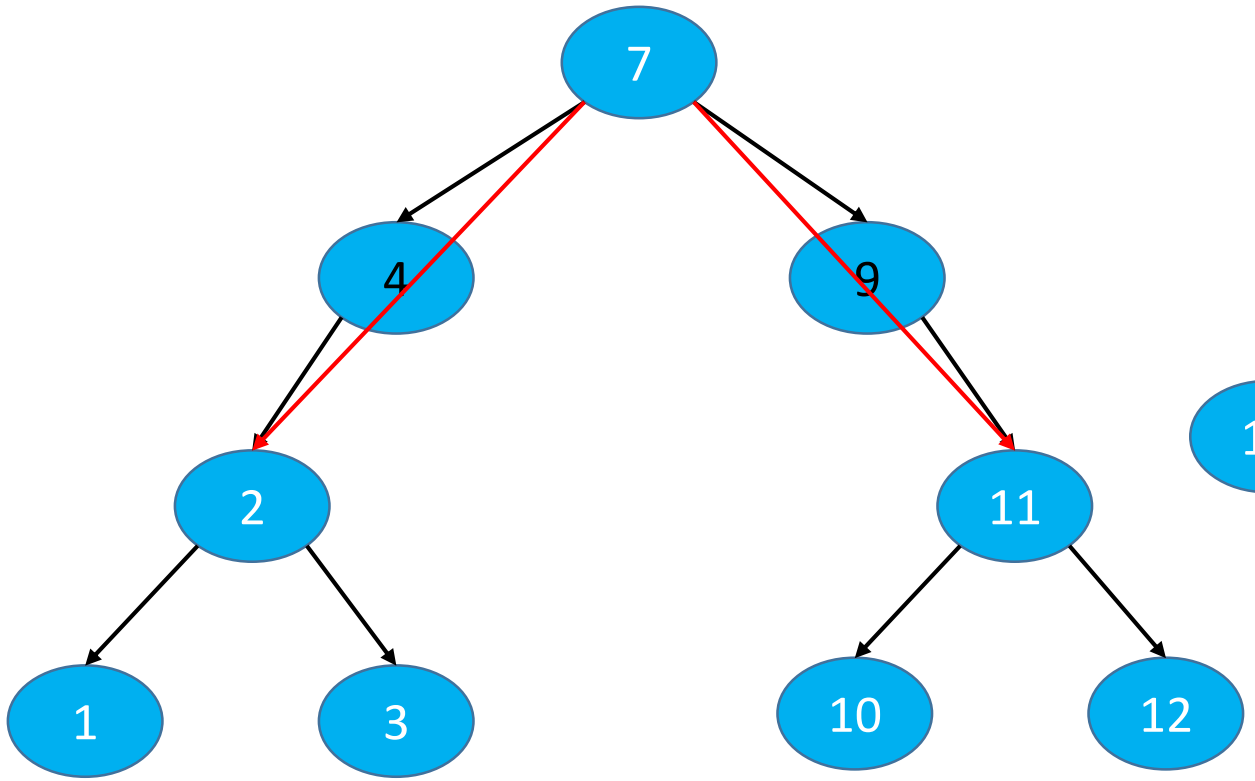
✓ `root = null`

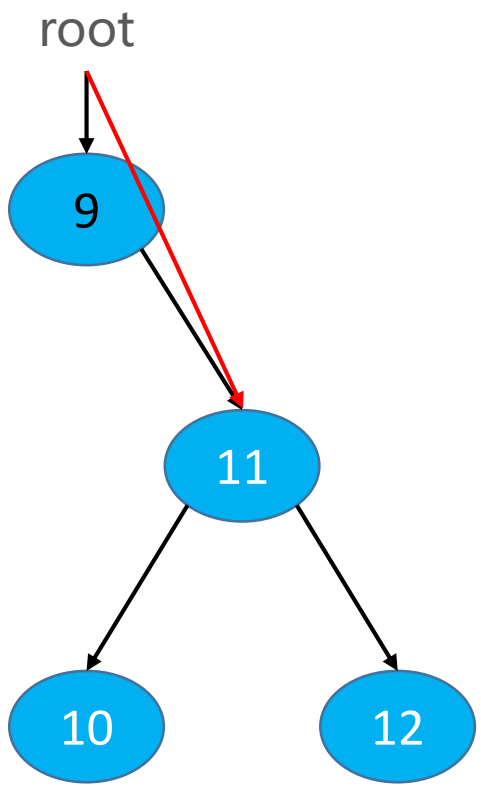


删除节点 – 度为1的节点

- 用子节点替代原节点的位置
  - child 是 node.left 或者 child 是 node.right

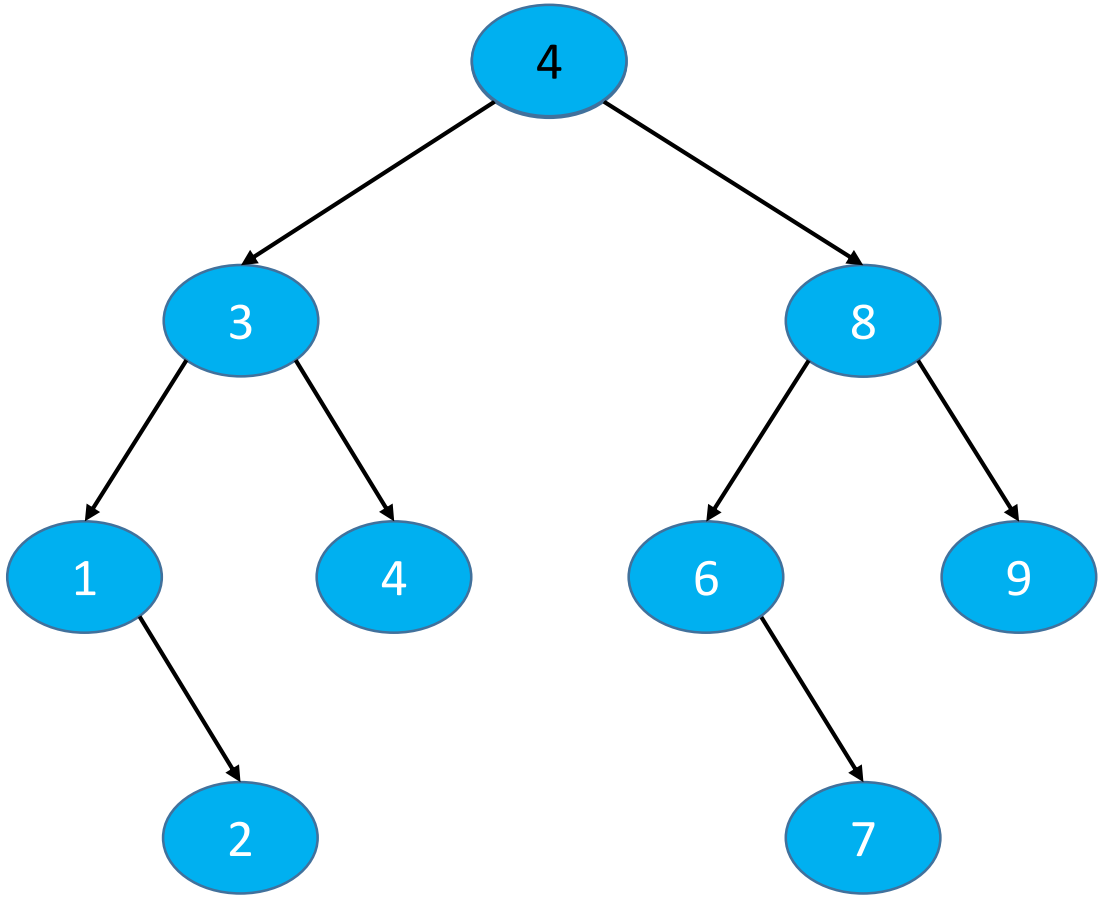
  - 用 child 替代 node 的位置
    - ✓ 如果 node 是左子节点
      - child.parent = node.parent
      - node.parent.left = child
    - ✓ 如果 node 是右子节点
      - child.parent = node.parent
      - node.parent.right = child
    - ✓ 如果 node 是根节点
      - root = child
      - child.parent = null



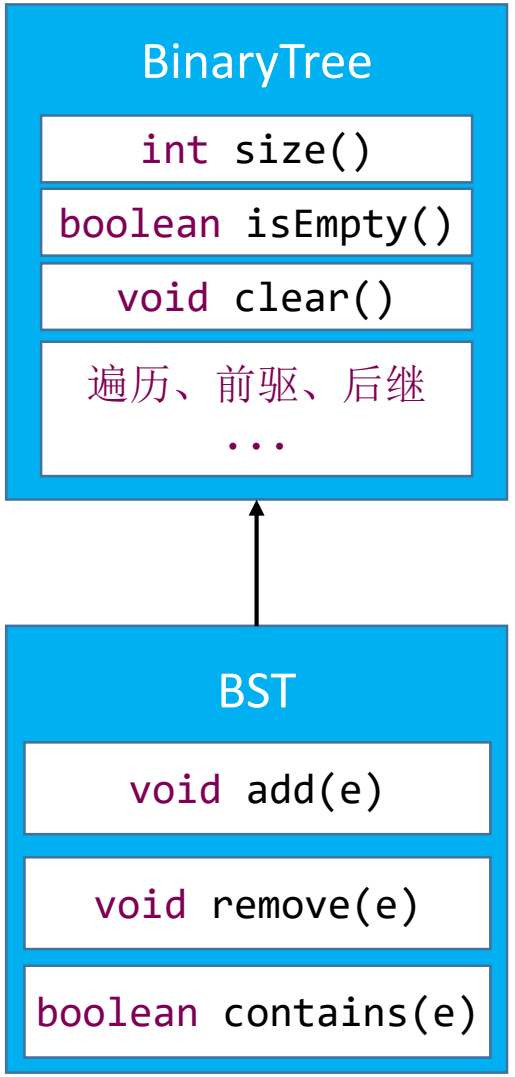


删除节点 – 度为2的节点

- 举例：先删除 5、再删除 4
- 先用前驱或者后继节点的值覆盖原节点的值
- 然后删除相应的前驱或者后继节点
- 如果一个节点的度为 2，那么
  - 它的前驱、后继节点的度只可能是 1 和 0



简单的继承结构


BinaryTree
int size()
boolean isEmpty()
void clear()
遍历、前驱、后继
...
BST
void add(e)
void remove(e)
boolean contains(e)

作业

- 删除二叉搜索树中的节点：https://leetcode-cn.com/problems/delete-node-in-a-bst/
- 二叉搜索树中的搜索：https://leetcode-cn.com/problems/search-in-a-binary-search-tree/
- 二叉搜索树中的插入操作：https://leetcode-cn.com/problems/insert-into-a-binary-search-tree/
- 验证二叉搜索树：https://leetcode-cn.com/problems/validate-binary-search-tree/comments/
- 二叉搜索树的最小绝对差：https://leetcode-cn.com/problems/minimum-absolute-difference-in-bst/comments/
- 二叉搜索树结点最小距离：https://leetcode-cn.com/problems/minimum-distance-between-bst-nodes/comments/

作业

- 将有序数组转换为二叉搜索树：https://leetcode-cn.com/problems/convert-sorted-array-to-binary-search-tree/
- 二叉搜索树的范围和：https://leetcode-cn.com/problems/range-sum-of-bst/
- 二叉搜索树的最近公共祖先：https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-search-tree/
- 二叉搜索树中第K小的元素：https://leetcode-cn.com/problems/kth-smallest-element-in-a-bst/
- 二叉搜索树迭代器：https://leetcode-cn.com/problems/binary-search-tree-iterator/
- 恢复二叉搜索树：https://leetcode-cn.com/problems/recover-binary-search-tree/