【第二十三周】手撕红黑树(下)-删除调整

1、99. 恢复二叉搜索树

1、把二叉搜索树看成有序序列，在有序序列里面，找到被调换的两个节点

2、就等价于交换有序序列的值，当后面的值小于前面的值(任意位置)，就是特殊位置，中序遍历二叉树，找到两个后面值比前面值小的位置

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {void} Do not return anything, modify root in-place instead.
 */
//  pre记录前一个值，p指向第一个值，q指向最后一个值。p 和 q都是指向两个被交换的节点
let pre,p,q;
var inorder = function(root) {
    if(root == null) return;
    inorder(root.left);
    // 从小到大依次访问每一个节点
    // 当前值小于前一个值
    if(pre && root.val < pre.val){
        // p指向第一个值
        if(p == null) p = pre;
        // q指向最后一个值
        q = root;
    }
    // 把前一个节点更新为root
    pre = root;
    inorder(root.right);
    return;
};
var recoverTree = function(root) {
    pre = p = q = null;
    inorder(root);
    // 交换p 和 q的值
    let temp;
    temp = p.val;
    p.val = q.val;
    q.val = temp;
    return;
};
```

2、653. 两数之和 IV - 输入 BST

1、中序遍历二叉搜索树，排成一列从小到大的有序序列。

2、在有序序列里面设置头尾指针，如果小于k，头指针往后走，否则，尾指针往前走

3、如果p还是小于q,证明这个时候二叉排序树中存在两个值相加等于

```
 /**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @param {number} k
 * @return {boolean}
 */
var inorder = function(root, ret) {
    if(root == null) return;
    inorder(root.left,ret);
    // ret数组中依次按着从小到大的顺序，存储二叉树的节点值
    ret.push(root.val);
    inorder(root.right,ret);
    return;
};
var findTarget = function(root, k) {
    let ret = [];
    inorder(root,ret);
    // 设置头p尾q指针
    let p = 0, q = ret.length - 1;
    //
    while(p < q && ret[p] + ret[q] - k){
        // 如果小于k，头指针往后走，否则，尾指针往前走
        if(ret[p] + ret[q] < k) p += 1;
        else q -= 1;
    }
    // 如果p还是小于q,证明这个时候二叉排序树中存在两个值相加等于k
    return p < q;
};
```

3、204. 计数质数

使用枚举

1、质数是只能被1和自身整除的数2

2、从[0, n)枚举数，如果其能被[2, n - 1]中任意数整除，不是质数

```
const isPrime = (x) => {
    for (let i = 2; i * i <= x; ++i) {
        if (x % i == 0) {
            return false;
        }
    }
    return true;
}

var countPrimes = function(n) {
    let ans = 0;
    for (let i = 2; i < n; ++i) {
        ans += isPrime(i);
    }
    return ans;
};
```

4、504. 七进制数

1、记录负号，负数取正

2、逐位提取，先模后除

3、0要特判，否则WA

4、负号补上，答案翻转

```
/**
 * @param {number} num
 * @return {string}
 */
var convertToBase7 = function(num) {
  let next = Math.abs(num)
  let res = ''
  while(next >= 7){
    res =  next % 7 + res
    next = next / 7 | 0
  }
  res = next + res
  return num < 0 ? '-' + res : res
};
```

5、384. 打乱数组

1、打乱数组度评判标准，打通数组的效果是 n个数产生的结果必须有n!种可能。

2、遍历nums数组，每次取[i, n-1]闭区间的一个随机数nums[rand]，交换nums[i]和nums[rand]即可。

3、拿第一个例子距离，执行for循环如下：

(1)第一轮，i = 0，rand 取值范围是 [0, 3]，有 4 个可能的取值 ['Solution', 'shuffle', 'reset', 'shuffle']

(2)第二轮，i = 1，rand 取值范围是 [1, 3]，有 3 个可能的取值 ['shuffle', 'reset','shuffle']

(3)第三轮，i = 2，rand 取值范围是 [2, 3]，有 2 个可能的取值 ['reset','shuffle']

(4)第四轮，i = 3，rand 取值范围是 [3, 3]，有 1 个可能的取值 ['shuffle']

共有结果数为 4*3*2*1 = 24 = 4!

```
/**
 * @param {number[]} nums
 */
var Solution = function(nums) {
    this.nums = nums;
};

/**
 * 获取原数组
 * Resets the array to its original configuration and return it.
 * @return {number[]}
 */
Solution.prototype.reset = function() {
    return this.nums;
};

/**
 * 打乱数组
 * Returns a random shuffling of the array.
 * @return {number[]}
 */
Solution.prototype.shuffle = function() {
    const nums = this.nums.slice(0);
    let n = nums.length;

    // 产生的结果有 n! 种可能
    for (let i = 0; i < n; i++) {
        // 从 i 到 n-1 随机选一个
        const rand = randOne(i, n - 1);
        // 交换nums数组i和rand下标的两个元素
        [ nums[i], nums[rand] ] = [ nums[rand], nums[i] ];
    }

    return nums;
};

// 获取闭区间 [n, m] 内的一个随机整数
function randOne(n, m) {
    return Math.floor(Math.random() * (m - n + 1)) + n;
};
```

开课吧