

340. 至多包含 K 个不同字符

样例1:

输入: S="e c e b a"并且k=3

输出: 4

解释: T="e c e b "

样例2:

输入: S="W O R L D"并且k=4

输出: 4

解释: T="W O R L " 或 "O R L D"

```
public class Solution {
    public int lengthOfLongestSubstringWithDistinct(String s, int k) {
        if (s.length() == 0 || k == 0) {
            return 0;
        }
        int left = 0, right = 0, cnt = 0;
        int charSet[] = new int[256];
        int ans = 0;
        while (right < s.length()) {
            // 统计right指向的字符
            // 当字符在窗口内第一次出现时, 字符种类数+1, 该字符出现次数+1
            if (charSet[s.charAt(right)] == 0) {
                cnt++;
            }
            charSet[s.charAt(right)]++;
            right++;
            // 向右移动left, 保持窗口内只有k种不同的字符
            while (cnt > k) {
                charSet[s.charAt(left)]--;
                // 当该字符在本窗口不再出现时, 字符种类数-1
            }
        }
    }
}
```

340. 至多包含 K 个不同字符 (cont)

```
> if (charSet[s.charAt(left)] == 0) {
    cnt--;
}
left++;
}
// 更新答案, 此时 cnt == k
ans = Math.max(ans, right - left);
}
return ans;
}
```

4. 寻找两个正序数组的中位数

示例 1:

输入: nums1 = [1,3], nums2 = [2]

输出: 2.00000

解释: 合并数组 = [1,2,3], 中位数 2

示例 2:

输入: nums1 = [1,2], nums2 = [3,4]

输出: 2.50000

解释: 合并数组 = [1,2,3,4], 中位数 (2 + 3) / 2 = 2.5

```
class Solution {
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
        int len1 = nums1.length;
        int len2 = nums2.length;
        // 减少操作次数
        if (len1 > len2) {
            // 确保N1是短的部分
            return findMedianSortedArrays(nums2, nums1);
        }
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 1 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

4. 寻找两个正序数组的中位数 (cont)

```
> }
// 开始从nums1中取元素
int left = 0;
int right = len1;
while (left <= right) {
    // 从nums1和nums2中分别取m1和m2个元素组成k个元素
    int m1 = left + (right - left) / 2;
    int m2 = (len1 + len2) / 2 - m1;
    // k -> max(L1, L2), (k + 1) -> min(R1, R2)
    // nums1没有取元素?
    // 此时左边的元素一定从nums2中取, 因此要使得L1 =
    MIN_VALUE
    // 这样max(L1, L2)就会取到L2了
    // 这时所取得元素中num2的元素都要比nums1中的小
    // 因此在min(R1,R2)的时候, 选中的也是R2
    double m1Left = (m1 == 0) ? Integer.MIN_VALUE :
    nums1[m1 - 1];
    // nums1取了所有的元素?
    double m1Right = (m1 == len1) ? Integer.MAX_VALUE :
    nums1[m1];
    // nums2没有取元素?
    double m2Left = (m2 == 0) ? Integer.MIN_VALUE :
    nums2[m2 - 1];
    // nums2取了所有的元素?
    double m2Right = (m2 == len2) ? Integer.MAX_VALUE :
    nums2[m2];
    // 左边取多了
    if (m1Left > m2Right) {
        right = m1 - 1;
    } else if (m2Left > m1Right) {
        // 左边取少了
        left = m1 + 1;
    } else {
```

4. 寻找两个正序数组的中位数 (cont)

```
> // 此时是符合要求的排列情况
if ((len1 + len2) % 2 == 0){
    double medLeft = Math.max(m1Left, m2Left);
    double medRight = Math.min(m1Right, m2Right);
    return (medLeft + medRight) / 2.0;
} else {
    return Math.min(m1Right, m2Right);
}
}
return -1;
}
```

662. 二叉树最大宽度

给你一棵二叉树的根节点 `root`，返回树的 **最大宽度**。
树的 **最大宽度** 是所有层中最大的 **宽度**。
每一层的 **宽度** 被定义为该层 **最左和 最右的 非空节点**（即，两个 **端点**）之间的 **长度**。将这个 **二叉树** 视作与 **满二叉树** 结构相同，**两端点** 间会出现一些 **延伸到这一层的 null 节点**，这些 **null 节点** 也计入长度。
题目数据保证 **答案** 将会在 **32 位 带符号整数** 范围内。

示例 1：

输入：`root = [1,3,2,5,3,null,9]`

输出：4

解释：最大宽度出现在树的第 3 层，宽度为 4 (5,3,null,9)。

示例 2：

输入：`root = [1,3,2,5,null,1,null,9,6,null,7]`

输出：7

解释：最大宽度出现在树的第 4 层，宽度为 7 (6,null,1,null,null,null,7)。



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 2 of 23.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

662. 二叉树最大宽度 (cont)

> 示例 3 :

输入 : root = [1,3,2,5]

输出 : 2

解释 : 最大宽度出现在树的第 2 层, 宽度为 2 (3,2)。

此题求二叉树所有层的最大宽度, 比较直观的方法是求出每一层的宽度, 然后

求出最大值。求每一层的宽度时, 因为两端点间的 null 节点也需要计入宽

度, 因此可以对节点进行编号。一个编号为 index 的左子节点的编号记为

$2 \times \text{index}$, 右子节点的编号记为 $2 \times \text{index} + 1$, 计算每层宽度时, 用每层节点的最大编号减去最小编号再加 1 即为宽度。

遍历节点时, 可以用广度优先搜索来遍历每一层的节点, 并求出最大值。

```
class Solution {
    public int widthOfBinaryTree(TreeNode root) {
        int res = 1;
        List<Pair<TreeNode, Integer>> arr = new ArrayList<Pair<TreeNode, Integer>>();
        arr.add(new Pair<TreeNode, Integer>(root, 1));
        while (!arr.isEmpty()) {
            List<Pair<TreeNode, Integer>> tmp = new ArrayList<Pair<TreeNode, Integer>>();
            for (Pair<TreeNode, Integer> pair : arr) {
                TreeNode node = pair.getKey();
                int index = pair.getValue();
                if (node.left != null) {
                    tmp.add(new Pair<TreeNode, Integer>(node.left, index * 2));
                }
                if (node.right != null) {
                    tmp.add(new Pair<TreeNode, Integer>(node.right, index * 2 + 1));
                }
            }
            arr = tmp;
        }
    }
}
```

662. 二叉树最大宽度 (cont)

```
>
    res = Math.max(res, arr.get(arr.size() - 1).getValue() -
arr.get(0).getValue() + 1);
    arr = tmp;
}
return res;
}
}
```

236. 二叉树的最近公共祖先

给定一个二叉树, 找到该树中两个指定节点的最近公共祖先。

百度百科中最近公共祖先的定义: “对于有根树 T 的两个节点 p、q, 最近公共祖先表示为一个节点 x, 满足 x 是 p、q 的祖先且 x 的深度尽可能大 (一个节点也可以是它自己的祖先)。”

示例 1 :

输入 : root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1

输出 : 3

解释 : 节点 5 和节点 1 的最近公共祖先是节点 3。

示例 2 :

输入 : root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4

输出 : 5

解释 : 节点 5 和节点 4 的最近公共祖先是节点 5。因为根据定义最近公共祖先节点可以为节点本身。

示例 3 :

输入 : root = [1,2], p = 1, q = 2

输出 : 1

```
class Solution {
    public TreeNode lowestCommonAncestor(
TreeNode root, TreeNode p, TreeNode q) {
        // base case
        if (root == null) return null;
        if (root == p || root == q) return
root;
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 3 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

236. 二叉树的最近公共祖先 (cont)

```
>   TreeNode left = lowestCommonAncestor(root.left, p, q);
   TreeNode right = lowestCommonAncestor(root.right, p, q);
   // 情况 1
   if (left != null && right != null) {
       return root;
   }
   // 情况 2
   if (left == null && right == null) {
       return null;
   }
   // 情况 3
   return left == null ? right : left;
}
```

25. K 个一组翻转链表

给你链表的头节点 `head`，每 `k` 个节点一组进行翻转，请你返回修改后的链表。

`k` 是一个正整数，它的值小于或等于链表的长度。如果节点总数不是 `k` 的整数倍，那么请将最后剩余的节点保持原有顺序。

你不能只是单纯的改变节点内部的值，而是需要实际进行节点交换。

示例 1：

输入：head = [1,2,3,4,5], k = 2

输出：[2, 1, 4, 3, 5]

示例 2：

输入：head = [1,2,3,4,5], k = 3

输出：[3, 2, 1, 4, 5]

```
class Solution {
    public ListNode reverseKGroup(ListNode head, int k) {
```

25. K 个一组翻转链表 (cont)

```
>   if (head == null) return null;
   // 区间 [a, b) 包含 k 个待反转元素
   ListNode a, b;
   a = b = head;
   for (int i = 0; i < k; i++) {
       // 不足 k 个，不需要反转，base case
       if (b == null) return head;
       b = b.next;
   }
   // 反转前 k 个元素
   ListNode newHead = reverse(a, b);
   // 递归反转后续链表并连接起来
   a.next = reverseKGroup(b, k);
   return newHead;
}
/ 反转区间 [a, b) 的元素，注意是左闭右开 /
ListNode reverse(ListNode a, ListNode b) {
    ListNode pre, cur, nxt;
    pre = null;
    cur = a;
    nxt = a;
    // while 终止的条件改一下就行了
    while (cur != b) {
        nxt = cur.next;
        cur.next = pre;
        pre = cur;
        cur = nxt;
    }
```

 By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 4 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

25. K 个一组翻转链表 (cont)

```
> // 返回反转后的头结点
return pre;
}
```

95. 不同的二叉搜索树 II

给你一个整数 n ，请你生成并返回所有由 n 个节点组成且节点值从 1 到 n 互不相同的不同二叉搜索树。可以按任意顺序返回答案。

示例 1:

输入: $n = 3$

输出: $[[1, null, 2, null, 3], [1, null, 3, 2], [2, 1, 3], [3, 1, null, 2], [3, 2, null, 1]]$

示例 2:

输入: $n = 1$

输出: $[[1]]$

```
class Solution {
    public List<TreeNode> generateTrees(int n) {
        if (n == 0) {
            return new LinkedList<TreeNode>();
        }
        return generateTrees(1, n);
    }

    public List<TreeNode> generateTrees(int start, int end) {
        List<TreeNode> allTrees = new LinkedList<TreeNode>();
        if (start > end) {
            allTrees.add(null);
            return allTrees;
        }
        // 枚举可行根节点
        for (int i = start; i <= end; i++) {
```

95. 不同的二叉搜索树 II (cont)

```
> // 获得所有可行的左子树集合
List<TreeNode> leftTrees = generateTrees(start, i - 1);
// 获得所有可行的右子树集合
List<TreeNode> rightTrees = generateTrees(i + 1, end);
// 从左子树集合中选出一棵左子树，从右子树集合中选出一棵
右子树，拼接到根节点上
for (TreeNode left : leftTrees) {
    for (TreeNode right : rightTrees) {
        TreeNode currTree = new TreeNode(i);
        currTree.left = left;
        currTree.right = right;
        allTrees.add(currTree);
    }
}
return allTrees;
}
```

103. 二叉树的锯齿形层次

103. 二叉树的锯齿形层次遍历

给你二叉树的根节点 $root$ ，返回其节点值的锯齿形层次遍历。（即先从左往右，再从右往左进行下一层遍历，以此类推，层与层之间交替进行）。

示例 1:

输入: $root = [3, 9, 20, null, null, 15, 7]$

输出: $[[3], [20, 9], [15, 7]]$

示例 2:

输入: $root = [1]$

输出: $[[1]]$



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 5 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

103. 二叉树的锯齿形层次 (cont)

> 示例 3 :

输入 : root = []

输出 : []

```
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        // 定义需要返回的变量
        List<List<Integer>> res = new ArrayList<>();
        // 创建双端队列
        Queue<TreeNode> queue = new ArrayDeque<>();
        // 创建层数索引
        int idx = 0;
        // 如果节点不为空
        if (root != null) {
            queue.add(root);
        }
        // 如果队列不为空
        while (!queue.isEmpty()) {
            // 计算这一层的根节点数
            int n = queue.size();
            idx++;
            // 新建List,用于存储每一层的节点数
            List<Integer> level = new ArrayList<>();
            for (int i = 0; i < n; i++) {
                TreeNode node = queue.poll();
                // 记录当前根节点
                level.add(node.val);
                // 将根节点删除,并依次加入所删除根节点的左节点与右节点。
                if (node.left != null) {
```

103. 二叉树的锯齿形层次 (cont)

```
>
            queue.add(node.left);
        }
        if (node.right != null) {
            queue.add(node.right);
        }
    }
    // 判断奇偶层
    if (idx%2 == 0){
        //反转level
        Collections.reverse(level);
    }
    res.add(level);
}
// 返回结果
return res;
}
}
```

700. 二叉搜索树中的搜索

给定二叉搜索树 (BST) 的根节点 `root` 和一个整数值 `val`。
你需要在 BST 中找到节点值等于 `val` 的节点。返回以该节点为根的子树。如果节点不存在,则返回 `null`。

示例 1:

输入 : root = [4,2,7,1,3], val = 2

输出 : [2,1,3]

示例 2:

输入 : root = [4,2,7,1,3], val = 5

输出 : []

```
class Solution {
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 6 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

700. 二叉搜索树中的搜索 (cont)

```
> public TreeNode searchBST(TreeNode root, int target) {
    if (root == null) {
        return null;
    }
    // 去左子树搜索
    if (root.val > target) {
        return searchBST(root.left, target);
    }
    // 去右子树搜索
    if (root.val < target) {
        return searchBST(root.right, target);
    }
    return root;
}
}
```

1038. 从二叉搜索树到更大和树

给定一个二叉搜索树 `root` (BST), 请将它的每个节点的值替换成树中大于或者等于该节点值的所有节点值之和。

提醒一下, 二叉搜索树 满足下列约束条件:

节点的左子树仅包含键 小于 节点键的节点。

节点的右子树仅包含键 大于 节点键的节点。

左右子树也必须是 二叉搜索树。

输入: [4, 1, 6, 0, 2, 5, 7, null, null, null, 3, null, null, null, 8]

输出: [30, 36, 21, 36, 35, 26, 15, null, null, null, 33, null, null, null, 8]

示例 2:

输入: root = [0, null, 1]

输出: [1, null, 1]

// 右根左遍历

1038. 从二叉搜索树到更大和树 (cont)

```
> class Solution {
    int tmp = 0;
    public TreeNode bstToGst(TreeNode root) {
        traverse(root);
        return root;
    }
    public void traverse(TreeNode root) {
        if (root == null) return;
        traverse(root.right);
        tmp += root.val;
        root.val = tmp;
        traverse(root.left);
    }
}
```

20. 有效的括号

```
Stack<Character> stack = new Stack<>();
for (char c : s.toCharArray()) {
    switch (c) {
        case '(': stack.push(c); break;
        case '[': stack.push(c); break;
        case '{': stack.push(c); break;
        default: if
            (stack.isEmpty() || c != stack.pop()) return
            false;
    }
}
return stack.isEmpty();
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 7 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

297. 二叉树的序列化与反序列化

输入: root = [1,2,3,null,null,4,5]

输出: [1, 2, 3, null, null, 4, 5]

示例 2:

输入: root = []

输出: []

示例 3:

输入: root = [1]

输出: [1]

示例 4:

输入: root = [1,2]

输出: [1,2]

```
public class Codec {
    Linked Lis t<S tri ng> list = new Linked -
    Lis t<>();
    public String serial ize (Tr eeNode root)
    {
        return traver se( root);
    }
    public String traver se( Tre eNode root) {
        if (root == null) return " nul l";
        String left = traver se( roo t.l -
eft);
        String right = traver se( roo -
t.r ight);
        return root.val + " ," + left +
        " ," + right;
    }
    public TreeNod e deseri ali ze( String
data) {
        Str ing[] strs = data.s pli -
t(", ");
        for (String str : strs) {
            lis t.a dd( str);
        }
        return traver se();
    }
}
```

297. 二叉树的序列化与反序列化 (cont)

```
> }
public TreeNod e traverse() {
    String str = list.removeFirst();
    if ("null".equals(str)) return null;
    TreeNod e root = new TreeNod e(Integer.valueOf(str));
    root.left = traverse();
    root.right = traverse();
    return root;
}
}
```

114. 二叉树展开为链表

输入: root = [1,2,5,3,4,null,6]

输出: [1, null, 1, 2, null, 1, 3, null, 1, 4, null, 1, 5, null, 1, 6]

示例 2:

输入: root = []

输出: []

示例 3:

输入: root = [0]

输出: [0]

```
class Solution {
    // 定义: 将以 root 为根的树拉平为链表
    public void flatte n(T reeNode root) {
        // base case
        if (root == null) return;
        // 先递归拉平左右子树
        fla tte n(r oot.left);
        fla tte n(r oot.r ight);
        *后序遍历位置*
    }
}
```



By **woshiamiaojiang**

Not published yet.

Last updated 20th September, 2023.

Page 8 of 23.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

114. 二叉树展开为链表 (cont)

```
> // 1、左右子树已经被拉平成一条链表
TreeNode left = root.left;
TreeNode right = root.right;
// 2、将左子树作为右子树
root.left = null;
root.right = left;
// 3、左子树的末端连接上当前的右子树
TreeNode p = root;
while (p.right != null) {
    p = p.right;
}
p.right = right;
}
```

101. 对称二叉树

示例 1:

输入: root = [1,2,2,3,4,4,3]

输出: true

示例 2:

输入: root = [1,2,2,null,3,null,3]

输出: false

```
class Solution {
    public boolean isSymmetric(TreeNode root) {
        return isSymmetric(root.left, root.right);
    }
    public boolean isSymmetric(TreeNode left, TreeNode right) {
        if (left == null && right == null)
            return true;
        if (left == null || right == null || left.val != right.val)
            return false;
    }
}
```

101. 对称二叉树 (cont)

```
> return isSymmetric(left.left, right.right) && isSymmetric(left.right, right.left);
}
```

144. 二叉树的前序遍历

```
class Solution {
    List<Integer> res = new LinkedList<>();
    public List<Integer> preorderTraversal(TreeNode root) {
        traverse(root);
        return res;
    }
    public void traverse(TreeNode root) {
        if (root == null) {
            return;
        }
        res.add(root.val);
        traverse(root.left);
        traverse(root.right);
    }
}
```

111. 二叉树的最小深度

```
class Solution {
    public int minDepth(TreeNode root) {
        if (root == null) return 0;
        LinkedList<TreeNode> list = new LinkedList<>();
        list.add(root);
        int depth = 0;
        while (!list.isEmpty()) {
            int size = list.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = list.poll();
                if (node.left == null && node.right == null)
                    return depth + 1;
                if (node.left != null)
                    list.add(node.left);
                if (node.right != null)
                    list.add(node.right);
            }
            depth++;
        }
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 9 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

111. 二叉树的最小深度 (cont)

```
> int size = list.size();
depth++;
for (int i = 0; i < size; i++) {
    TreeNode cur = list.removeFirst(); // eaily wrong
    if (cur.left == null && cur.right == null) {
        return depth;
    }
    if (cur.left != null) list.add(cur.left);
    if (cur.right != null) list.add(cur.right);
}
}
return depth;
}
```

105. 从前序与中序遍历序列构造二叉树

```
class Solution {
    HashMap<Integer, Integer> inOrderValMap = new HashMap<>();
    public TreeNode buildTree (int[]
preOrder, int[] inOrder) {
        for (int i = 0; i < preOrder.length; i++) inOrderValMap.put(inOrder[i], i);
        return buildTree (preOrder,
inOrder, 0, preOrder.length - 1, 0, inOrder.length - 1);
    }
    // pre: 3 9 20 15 7
    // T L
    // in : 9 3 15 20 7
    // L T
    public TreeNode buildTree (int[]
preOrder, int[] inOrder, int preLeft, int preRight,
int inLeft, int inRight) {
        if (preLeft > preRight) return null;
```

105. 从前序与中序遍历序列构造二叉树 (cont)

```
> int rootVal = preOrder[preLeft];
TreeNode root = new TreeNode(rootVal);
int rootIdx = inOrderValMap.get(rootVal);
int inLeftSize = rootIdx - inLeft; // 1
root.left = buildTree(preOrder, inOrder, preLeft + 1, preLeft +
inLeftSize, inLeft, inLeft + inLeftSize - 1);
root.right = buildTree(preOrder, inOrder, preLeft + inLeftSize +
1, preRight, rootIdx + 1, inRight);
return root;
}
```

124. 二叉树中的最大路径和

```
class Solution {
    int res = Integer.MIN_VALUE;
    public int maxPathSum(TreeNode root) {
        getMaxPathSum(root);
        return res;
    }
    public int getMaxPathSum(TreeNode
root) {
        if (root == null) {
            return 0;
        }
        int left = Math.max(0, getMaxPathSum(
root.left));
        int right = Math.max(0, getMaxPathSum(
root.right));
        res = Math.max(left + right +
root.val, res);
        return Math.max(left, right) +
root.val;
    }
}
```



By **woshiamiaojiang**

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 10 of 23.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

124. 二叉树中的最大路径和 (cont)

> 543. 二叉树的直径

```
class Solution {
    int res = 0;
    public int diameterOfBinaryTree(TreeNode root) {
        traverse(root);
        return res;
    }
    public int traverse(TreeNode root) {
        if (root == null) {
            return 0;
        }
        int left = traverse(root.left);
        int right = traverse(root.right);
        res = Math.max(left + right, res);
        return Math.max(left, right) + 1;
    }
}
```

2. 两数相加 - 力扣 (Leetcode)

链表格式

```
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode first = l1, second = l2;
        int add = 0;
        ListNode fakeHead = new ListNode(-1);
        ListNode res = fakeHead;
        while (first != null || second != null) {
            int sum = (first == null ? 0 : first.val) + (second == null ? 0 : second.val) + add;
            add = sum / 10;
        }
    }
}
```

2. 两数相加 - 力扣 (Leetcode) (cont)

```
>
    res.next = new ListNode(sum % 10);
    res = res.next;
    if (first != null) first = first.next;
    if (second != null) second = second.next;
}
if (add != 0) {
    res.next = new ListNode(add % 10);
}
return fakeHead.next;
}
```

138. 复制带随机指针的链表

示例 1:

输入: head = [[7, null], [1, 3, 0], [11, 4], [10, 2], [1, 0]]

输出: [[7, null], [1, 3, 0], [11, 4], [10, 2], [1, 0]]

示例 2:

输入: head = [[1, 1], [2, 1]]

输出: [[1, 1], [2, 1]]

示例 3:

输入: head = [[3, null], [3, 0], [3, null]]

输出: [[3, null], [3, 0], [3, null]]

```
class Solution {
    Map<Node, Node> map = new HashMap<Node, Node>();
    public Node copyRandomList(Node oldNode) {
        if (oldNode == null) {
            return null;
        }
        if (map.containsKey(oldNode))
        {
        }
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 11 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

138. 复制带随机指针的链表 (cont)

```
> return map.get(oldNode);
} else {
    Node newNode = new Node(oldNode.val);

    newNode.next = copyRandomList(oldNode.next);
    newNode.random = copyRandomList(oldNode.random);
    map.put(oldNode, newNode);
    return newNode;
}
}
```

146. LRU缓存机制

```
public class LRUCache {
    Map<Integer, Integer> map = new LinkedHashMap<Integer, Integer>();
    int cap;

    public LRUCache(int capacity) {
        cap = capacity;
    }

    public int get(int key) {
        if (!map.containsKey(key))
            return -1;

        int val = map.remove(key);
        map.put(key, val);
        return val;
    }

    public void set(int key, int value) {
        if (map.containsKey(key)) {
            map.remove(key);
            map.put(key, value);
        }
    }
}
```

146. LRU缓存机制 (cont)

```
> return;
}

map.put(key, value);

if (map.size() > cap)
    map.remove(map.entrySet().iterator().next().getKey());
}
```

93. 复原IP地址

示例 1:

输入: s = "255 255 111 35"

输出: ["25 5.2 55.1 1.1 35 ", "25 5.2 55.1 11.35 "]

示例 2:

输入: s = "000 0"

输出: ["0.0.0.0"]

示例 3:

输入: s = "101 023 "

输出: ["1.0.1 0.2 3", "1.0.1 02.3", "10.1.0.23 ", "10.10.2.3 ", "101.0.2.3"]

```
class Solution {
    // Temporary list to store parts of the IP address
    List<String> temp = new ArrayList<>();
    // Result list to store all valid IP addresses
    List<String> res = new ArrayList<>();

    // Public method to start the process of restoring IP addresses
    public List<String> restoreIpAddresses(String s) {
        // Start backtracking from the first character
        backtrack(s, 0);
        // Return the result list
        return res;
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 12 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

93. 复原IP地址 (cont)

```

> }
// Private method to perform backtracking
private void backtrack(String s, int curIdx) {
    // If we have 4 parts in the IP address and we haven't reached
    the end of the string, return
    if (temp.size() == 4 && curIdx != s.length()) return;
    // If we have 4 parts in the IP address and we have reached the
    end of the string, add to result
    if (temp.size() == 4 && curIdx == s.length()){
        // Join the parts of the IP address with a dot and add to the
        result list
        res.add(String.join(".", temp)); //这个join函数用时2ms, 手动-
        StringBuilder的话就是1ms
        return;
    }
    // Loop through the next 3 characters of the string
    for(int i = curIdx; i < s.length() && i < curIdx + 3; i++){
        // Get the substring from the current index to the current index
        + 1
        String sub = s.substring(curIdx, i + 1);
        // If the integer value of the substring is less than or equal to
        255
        if (Integer.parseInt(sub) <= 255){
            // If the substring has more than one character and the first
            character is '0', return
            if(sub.length() > 1 && s.charAt(curIdx) == '0'){
                return;
            }
            // Add the substring to the temporary list
            temp.add(sub);
            // Continue backtracking from the next character
            backtrack(s, i + 1);
            // Remove the last element from the temporary list
            temp.remove(temp.size() - 1);
        } else {

```

93. 复原IP地址 (cont)

```

> // If the integer value of the substring is greater than 255,
return
    return;
    }
    }
    }
}

```

513. 找树左下角的值

示例 1:

输入: root = [2,1,3]

输出: 1

示例 2:

输入: [1,2,3 ,4, null, 5 ,6, null, null, 7]

输出: 7

```

class Solution {
    public int findBottomLeftValue(TreeNode root) {
        Queue<TreeNode> q = new
        ArrayDeque<>();
        q.offer(root);
        int ans = 0;
        while (!q.isEmpty()) {
            ans = q.peek().val;
            for (int i = q.size(); i >
            0; --i) {
                TreeNode node =
                q.poll();
                if (node.left !=
                null) {
                    q.offer(
                    node.left);
                }
                if (node.right !=
                null) {
                    q.offer(
                    node.right);
                }
            }
        }
        return ans;
    }
}

```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 13 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

513. 找树左下角的值 (cont)

```
>
    }
    }
    return ans;
}
}
```

129. 求根节点到叶节点数字之和

给你一个二叉树的根节点 `root`，树中每个节点都存放有一个 0 到 9 之间的数字。

每条从根节点到叶节点的路径都代表一个数字：

例如，从根节点到叶节点的路径 1 -> 2 -> 3 表示数字 123。

。

计算从根节点到叶节点生成的所有数字之和。

叶节点是指没有子节点的节点。

示例 1：

输入：`root = [1,2,3]`

输出：25

解释：

从根到叶子节点路径 1->2 代表数字 12

从根到叶子节点路径 1->3 代表数字 13

因此，数字总和 = 12 + 13 = 25

示例 2：

输入：`root = [4,9,0,5,1]`

输出：1026

解释：

从根到叶子节点路径 4-> 9->5 代表数字 495

从根到叶子节点路径 4-> 9->1 代表数字 491

从根到叶子节点路径 4->0 代表数字 40

因此，数字总和 = 495 + 491 + 40 = 1026

```
class Solution {
```

129. 求根节点到叶节点数字之和 (cont)

```
> public int sumNumbers(TreeNode root) {
    return cal(root, 0);
}

public int cal(TreeNode root, int cur) {
    if (root == null) return 0;
    cur = cur * 10 + root.val;
    if (root.left == null && root.right == null) return cur;
    return cal(root.left, cur) + cal(root.right, cur);
}
}
```

958. 二叉树的完全性检验

给你一棵二叉树的根节点 `root`，请你判断这棵树是否是一棵完全二叉树。

在一棵完全二叉树中，除了最后一层外，所有层都被完全填满，并且最后一层中的所有节点都尽可能靠左。最后一层（第 h 层）中可以包含 1 到 2^h 个节点。

示例 1：

输入：`root = [1,2,3,4,5,6]`

输出：`true`

解释：最后一层前的每一层都是满的（即，节点值为 {1} 和 {2,3} 的两层），且最后一层中的所有节点（{4,5,6}）尽可能靠左。

示例 2：

输入：`root = [1,2,3,4,5,null,7]`

输出：`false`

解释：值为 7 的节点不满足条件「节点尽可能靠左」。

// Java层序遍历，判断size和index是否相等即可

```
class Solution {
    public boolean isCompleteTree(TreeNode root) {
        if (root == null) return false;
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 14 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

958. 二叉树的完全性检验 (cont)

```
> LinkedList<TreeNode> queue = new LinkedList<>();
LinkedList<Integer> indexQueue = new LinkedList<>();
queue.offer(root);
indexQueue.offer(0);
int maxIndex = 0;
int size = -1;
while (!queue.isEmpty()) {
    TreeNode node = queue.poll();
    int index = indexQueue.poll();
    maxIndex = Math.max(index, maxIndex);
    size++;
    if (node.left != null) {
        queue.offer(node.left);
        indexQueue.offer(index * 2 + 1);
    }
    if (node.right != null) {
        queue.offer(node.right);
        indexQueue.offer(index * 2 + 2);
    }
}
return size == maxIndex;
}
```

199. 二叉树的右视图

```
输入: [1,2,3,null,5,null,4]
输出: [1,3,4]
示例 2:
输入: [1,null,3]
输出: [1,3]
示例 3:
输入: []
输出: []
// BFS
class Solution {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> res = new ArrayList<>();
        if (root == null) {
            return res;
        }
        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);
        while (!queue.isEmpty()) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = queue.poll();
                if (node.left != null) {
                    queue.offer(node.left);
                }
                if (node.right != null) {
                    queue.offer(node.right);
                }
                if (i == size - 1) {
                    // 将当前层的最后一个节点放入结果列表
                }
            }
        }
    }
}
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 15 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

199. 二叉树的右视图 (cont)

```
>         res.add(node.val);
        }
    }
}
return res;
}
```

// DFS

我们按照「根结点 -> 右子树 -> 左子树」的顺序访问，就可以保证每层都是最先访问最右边的节点的。

(与先序遍历「根结点 -> 左子树 -> 右子树」正好相反，先序遍历每层最先访问的是最左边的节点)

```
class Solution {
    List<Integer> res = new ArrayList<>();
    public List<Integer> rightSideView(TreeNode root) {
        dfs(root, 0); // 从根节点开始访问，根节点深度是0
        return res;
    }
    private void dfs(TreeNode root, int depth) {
        if (root == null) {
            return;
        }
        // 先访问当前节点，再递归地访问右子树和左子树。
        if (depth == res.size()) { // 如果当前节点所在深度还没有出现在res里，说明在该深度下当前节点是第一个被访问的节点，因此将当前节点加入res中。
            res.add(root.val);
        }
        depth++;
    }
}
```

199. 二叉树的右视图 (cont)

```
>     dfs(root.right, depth);
        dfs(root.left, depth);
    }
}
```

206. 反转链表

示例 1:

输入: head = [1,2,3,4,5]

输出: [5,4,3,2,1]

示例 2:

输入: head = [1,2]

输出: [2,1]

示例 3:

输入: head = []

输出: []

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null, next;
        ListNode curr = head;
        while (curr != null) {
            next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 16 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

222. 完全二叉树的节点个数

给你一棵完全二叉树的根节点 `root`，求出该树的节点个数。
完全二叉树的定义如下：在完全二叉树中，除了最底层节点可能没填满外，其余每层节点数都达到最大值，并且最下面一层的节点都集中在该层最左边的若干位置。若最底层为第 h 层，则该层包含 $1 \sim 2^h$ 个节点。

示例 1：

输入：`root = [1,2,3,4,5,6]`

输出：6

示例 2：

输入：`root = []`

输出：0

示例 3：

输入：`root = [1]`

输出：1

```
class Solution {
    public int countNodes(TreeNode root) {
        if (root == null) return 0;
        TreeNode left = root, right = root;
        int left_cnt = 1, right_cnt = 1;
        while (left.left != null) {
            left = left.left;
            left_cnt++;
        }
        while (right.right != null) {
            right = right.right;
            right_cnt++;
        }
        if (left_cnt == right_cnt) {
            return (int) Math.pow(2, left_cnt) - 1;
        }
    }
}
```

222. 完全二叉树的节点个数 (cont)

```
> }
    return 1 + countNodes(root.left) + countNodes(root.right);
}
}
```

235. 二叉搜索树的最近公共祖先

给定一个二叉搜索树，找到该树中两个指定节点的最近公共祖先。
百度百科中最近公共祖先的定义为：“对于有根树 T 的两个结点 p 、 q ，最近公共祖先表示为一个结点 x ，满足 x 是 p 、 q 的祖先且 x 的深度尽可能大（一个节点也可以是它自己的祖先）。”

例如，给定如下二叉搜索树：`root = [6,2,8,0,4,7,9,null,null,3,5]`

示例 1：

输入：`root = [6,2,8,0,4,7,9,null,null,3,5]`,

`p = 2, q = 8`

输出：6

解释：节点 2 和节点 8 的最近公共祖先是 6。

示例 2：

输入：`root = [6,2,8,0,4,7,9,null,null,3,5]`,

`p = 2, q = 4`

输出：2

解释：节点 2 和节点 4 的最近公共祖先是 2，因为根据定义最近公共祖先节点可以为节点本身。

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if (p.val > q.val) {
            return traverse(root, q, p);
        }
        return traverse(root, p, q);
    }
    public TreeNode traverse(TreeNode root, TreeNode p, TreeNode q) {
        if (root == null) return null;
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 17 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

235. 二叉搜索树的最近公共祖先 (cont)

```
> if (root == p || root == q) return root;
if (root.val < p.val) {
    return traverse(root.right, p, q);
} else if (root.val > q.val) {
    return traverse(root.left, p, q);
}
return root;
}
```

341. 扁平化嵌套列表迭代器

示例 1:

输入: nestedList = [[1,1], 2, [1,1]]

输出: [1, 1, 2, 1, 1]

解释: 通过重复调用 next 直到 hasNext 返回 false, next 返回的元素的顺序应该是: [1, 1, 2, 1, 1]。

示例 2:

输入: nestedList = [1, [4, [6]]]

输出: [1, 4, 6]

解释: 通过重复调用 next 直到 hasNext 返回 false, next 返回的元素的顺序应该是: [1, 4, 6]。

```
public class NestedIterator implements Iterator<Integer> {
    LinkedList<NestedInteger> list;
    public NestedIterator(List<NestedInteger> nestedList) {
        list = new LinkedList<>(nestedList);
    }
    @Override
    public Integer next() {
        return list.removeFirst().getInteger();
    }
}
```

341. 扁平化嵌套列表迭代器 (cont)

```
> }
@Override
public boolean hasNext() {
    while (!list.isEmpty() && !list.getFirst().isInteger()) {
        List<NestedInteger> cur_list = list.removeFirst().getList();
        for (int i = cur_list.size() - 1; i >= 0; i--) {
            list.addFirst(cur_list.get(i));
        }
    }
    return !list.isEmpty();
}
```

96. 不同的二叉搜索树

给你一个整数 n, 求恰由 n 个节点组成且节点值从 1 到 n 互不相同的二叉搜索树有多少种? 返回满足题意的二叉搜索树的种数。

示例 1:

输入: n = 3

输出: 5

示例 2:

输入: n = 1

输出: 1

```
class Solution {
    public int numTrees(int n) {
        return sum(1, n);
    }
    private int sum(int left, int right) {
        if (right <= left) return 1;
        int sum = 0;
    }
}
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 18 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

96. 不同的二叉搜索树 (cont)

```
> for(int i=left;j<=right;i++) sum += sum(left,i-1) * sum(i+1,right);
return sum;
}
```

98. 验证二叉搜索树

给你一个二叉树的根节点 `root`，判断其是否是一个有效的二叉搜索树。

有效 二叉搜索树定义如下：

节点的左子树只包含 小于 当前节点的数。

节点的右子树只包含 大于 当前节点的数。

所有左子树和 右子树 自身必须也是 二叉搜索树。

示例 1：

输入：`root = [2,1,3]`

输出：`true`

示例 2：

输入：`root = [5,1,4,null,3,6]`

输出：`false`

解释：根节点的值是 5，但是右子节点的值是 4。

```
class Solution {
    Tre eNode pre = null;
    public boolean isVali dBS T(T reeNode
root) {
        if (root == null) {
            return true;
        }
        if (! isVali dBS T(r oot.left)) {
            return false;
        }
        if (pre != null && root.val <=
pre.val) {
            return false;
        }
        pre = root;
        return isVali dBS T(r oot.right);
    }
}
```

98. 验证二叉搜索树 (cont)

```
> }
pre = root;
return isValidBST(root.right);
}
```

652. 寻找重复的子树

输入：`root = [1,2,3,4,null,2,4,null,null,4]`

输出：`[[2,4],[4]]`

示例 2：

输入：`root = [2,1,1]`

输出：`[[1]]`

示例 3：

输入：`root = [2,2,2,3,null,1,3,null]`

输出：`[[2,3],[3]]`

```
class Solution {
    Has hMa p<S tring, Intege r> map = new
HashMa p<>();
    Lis t<T ree Nod e> res = new Linke d Lis t<>
();
    public List<T ree Nod e> findDu pli cat -
eSu btr ees (Tr eeNode root) {
        tra ver se( root);
        return res;
    }
    public String tra ver se( Tre eNode root) {
        if (root == null) return " nul l";
        String left = tra ver se( roo t.l -
eft);
        String right = tra ver se( roo -
t.r ight);
        String str = root.val + " ," +
left + " ," + right;
        if (map.c ont ain sKe y(str) &&
map.ge t(str) == 1) {
            res.ad d(r oot);
        }
    }
}
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 19 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopod.com>

652. 寻找重复的子树 (cont)

```
> }
    map.put(str, map.getOrDefault(str, 0) + 1);
    return str;
}
}
```

654. 最大二叉树

给定一个不重复的整数数组 `nums`。最大二叉树 可以用下面的算法从 `nums` 递归地构建:

创建一个根节点, 其值为 `nums` 中的最大值。

递归地在最大值 左边 的子数组前缀上 构建左子树。

递归地在最大值 右边 的子数组后缀上 构建右子树。

返回 `nums` 构建的 最大二叉树 。

输入: `nums = [3,2,1,6,0,5]`

输出: `[6, 3, 5, null, 2, 0, null, null, 1]`

解释: 递归调用如下所示:

- `[3,2,1,6,0,5]` 中的最大值是 `6`, 左边部分是 `[3,2,1]`, 右边部分是 `[0,5]`。

- `[3,2,1]` 中的最大值是 `3`, 左边部分是 `[]`, 右边部分是 `[2,1]`。

- 空数组, 无子节点。

- `[2,1]` 中的最大值是 `2`, 左边部分是 `[]`

, 右边部分是 `[1]`。

- 空数组, 无子节点。

- 只有一个元素, 所以子节点是

一个值为 `1` 的节点。

- `[0,5]` 中的最大值是 `5`, 左边部分是 `[0]`, 右边部分是 `[]`。

- 只有一个元素, 所以子节点是一个值为 `0` 的节点。

- 空数组, 无子节点。

```
class Solution {
    public TreeNode constructMaximumBinaryTree(int[] nums) {
        return maxTree(nums, 0, nums.length - 1);
    }
}
```

654. 最大二叉树 (cont)

```
> public TreeNode maxTree(int[] nums, int l, int r){
    if(l > r){
        return null;
    }
    //bond为当前数组中最大值的索引
    int idx = findMax(nums, l, r);
    TreeNode root = new TreeNode(nums[bond]);
    root.left = maxTree(nums, l, idx - 1);
    root.right = maxTree(nums, idx + 1, r);
    return root;
}
//找最大值的索引
public int findMax(int[] nums, int l, int r){
    int max = Integer.MIN_VALUE, maxIndex = l;
    for(int i = l; i <= r; i++){
        if(max < nums[i]){
            max = nums[i];
            maxIndex = i;
        }
    }
    return maxIndex;
}
}
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 20 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

116. 填充每个节点的下一个右指针

```
class Solution {
    public Node connect(Node root) {
        if (root == null) return null;
        traverse(root.left, root.right);
        return root;
    }
    public void traverse(Node left, Node right) {
        if (left == null || right == null) return;
        left.next = right;
        traverse(left.left, left.right);
        traverse(right.left, right.right);
        traverse(left.right, right.left);
    }
}
```

226. 翻转二叉树

```
class Solution {
    public TreeNode invertTree(TreeNode root) {
        if (root == null) return null;
        TreeNode left = invertTree(root.left);
        TreeNode right = invertTree(root.right);
        root.right = left;
        root.left = right;
        return root;
    }
}
```

515. 在每个树行中找最大值

示例1:

输入: root = [1,3,2,5,3,null,9]

输出: [1,3,9]

示例2:

输入: root = [1,2,3]

输出: [1,3]

```
// dfs(root, 结果, 层级)
// 如果root 是null返回
//
class Solution {
    // 二叉树层序遍历 寻找最大值
    public List<Integer> largestValues(TreeNode root) {
        List<Integer> res = new LinkedList<>();
        if (null == root) {
            return res;
        }
        LinkedList<TreeNode> queue = new LinkedList<>();
        queue.add(root);
        while (!queue.isEmpty()) {
            int size = queue.size();
            int max = queue.getFirst().val;
            for (int i = 0; i < size; i++) {
                TreeNode node = queue.removeFirst();
                if (node.val > max) {
                    max = node.val;
                }
                if (node.left != null) {
                    queue.addLast(node.left);
                }
            }
        }
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 20th September, 2023.

Page 21 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish Yours!

<https://apollopod.com>

515. 在每个树行中找最大值 (cont)

```
>     }
    if (node.right != null) {
        queue.addLast(node.right);
    }
}
res.add(max);
}
return res;
}
```

230. 二叉搜索树中第K小的元素

```
class Solution {
    int k = 0;
    int res = 0;
    public int kthSmallest(TreeNode root,
int k) {
        this.k = k;
        traverse(root);
        return res;
    }
    public void traverse(TreeNode root) {
        if (root == null) {
            return;
        }
        traverse(root.left);
        k--;
        if (k == 0) {
            res = root.val;
            return;
        }
    }
}
```

230. 二叉搜索树中第K小的元素 (cont)

```
>     }
    traverse(root.right);
}
}
```

78. 子集

示例 1:

输入: nums = [1,2,3]

输出: [[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]

示例 2:

输入: nums = [0]

输出: [[], [0]]

```
class Solution {
    //定义二维数组res用于存储结果
    List<List<Integer>> res = new Linked -
List<>();
    public List<List<Integer>> subset -
s(int[] nums) {
        //定义路径数组
        List<Integer> track = new
LinkedList<>();
        backtrack(nums, 0, track);
        return res;
    }
    public void backtrack(int[] nums, int
start, List<Integer> track) {
        //添加路径数组到结果数组中
        res.add(new LinkedList<>(tr -
ack));
        //for循环遍历数组nums
        for (int i = start; i < nums.l -
ength; i++) {
            //做选择, 将选择添加到路径
数组中
            track.add(nums[i]);
            //回溯, 继续向后遍历
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 22 of 23.

Sponsored by [ApolloPad.com](https://apollopod.com)

Everyone has a novel in them. Finish

Yours!

<https://apollopod.com>

78. 子集 (cont)

```
> backtrack(nums, i + 1, track);
//撤销选择，将选择从路径中删除
track.remove(track.size() - 1);
}
}
}
```

125. 验证回文串

示例 1:

输入: s = "A man, a plan, a canal: Panama "

输出: true

解释: " ama nap lan aca nal pan ama " 是回文串。

示例 2:

输入: s = "race a car"

输出: false

解释: " rac eac ar" 不是回文串。

示例 3:

输入: s = " "

输出: true

解释: 在移除 非字母 数字字 符之后, s 是一个空字符串 " "。

由于空字符串 正着反 着读都 一样, 所以是回文串。

```
class Solution {
    public boolean isPalindrome (String s) {
        int n = s.length();
        int left = 0, right = n - 1;
        while (left < right) {
            while (left < right &&
!Character.isLetterOrDigit(s.charAt(left))) {
                left++;
            }
            while (left < right &&
!Character.isLetterOrDigit(s.charAt(right))) {
                right--;
            }
            if (Character.toLowerCase(s.charAt(left)) != Character.to-
LowerCase(s.charAt(right))) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

125. 验证回文串 (cont)

```
> while (left < right && !Character.isLetterOrDigit(s.charAt(right))) {
    right--;
}
if (left < right) {
    if (Character.toLowerCase(s.charAt(left)) != Character.to-
LowerCase(s.charAt(right))) {
        return false;
    }
    left++;
    right--;
}
return true;
}
```



By woshiamiaojiang

cheatography.com/woshiamiaojiang/

Not published yet.

Last updated 20th September, 2023.

Page 23 of 23.

Sponsored by [ApolloPad.com](https://apollopad.com)

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>