

124. 二叉树中的最大路径和

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

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



322. 零钱兑换

```

/*
    开辟一个动态数组，基底dp[0]=0。
    dp[i] = num。
    i代表目标值，num代表硬币个数。
    num初值为amt+1。代表不可能。
    每个dp[i]都与dp[i-coin]的结果相关联。
*/
class Solution {
    public int coinChange(int[] coins, int amount) {

        int[] dp = new int[amount + 1];
        Arrays.fill(dp, amount + 1);
        dp[0] = 0;
        for (int i = 1; i < dp.length; i++) {
            for (int coin : coins) {
                if (i - coin >= 0) {
                    dp[i] = Math.min(dp[i - coin] + 1, dp[i]);
                }
            }
        }
        return dp[dp.length - 1] == amount + 1 ? -1 : dp[dp.length - 1];
    }
}

```

46. 全排列

```

class Solution {
    boolean[] used;
    List<List<Integer>> res;
    public List<List<Integer>> permute(int[] nums) {
        res = new LinkedList<>();
        used = new boolean[nums.length];
        backtrack(nums, new LinkedList<>());
        return res;
    }
    public void backtrack(int[] nums, List<Integer> track) {
// 临时track大小满的时候
        if (track.size() == nums.length) {
            res.add(new LinkedList(track));
            return;
        }
        for (int i = 0; i < nums.length; i++) {
            if (used[i]) continue;

```



46. 全排列 (cont)

```
> // 做选择, 添加, 下探结果
    used[i] = true;
    track.add(nums[i]);
    backtrack(nums, track);
    used[i] = false;
    track.removeLast();
  }
}
```

105. 从前序与中序遍历序列构造二叉树

```
class Solution {
    Has hMa p<Integer, Integer> inOrderValMapIdx = new HashMap<>();
    public TreeNode buildTree (int[] preorder, int[] inorder) {
        for (int i = 0; i < inorder.length; i++) {
            inOrderValMapIdx.put(inorder[i], i);
        }
        return buildTree (preorder, inorder, 0, preorder.length - 1, 0, inorder.length - 1);
    }
    // pre: 3 9 20 15 7
    // T L R
    // in : 9 3 15 20 7
    // L T R
    public TreeNode buildTree (int[] preorder, int[] inorder, int preStart, int preEnd, int inStart,
int inEnd) {
        if (preStart > preEnd) {
            return null;
        }
        int rootVal = preorder[preStart];
        int inOrderRootIdx = inOrderValMapIdx.get(rootVal);
        TreeNode root = new TreeNode(rootVal);
        root.left = buildTree (preorder, inorder, preStart + 1, preStart + inOrderRootIdx -
inStart, inStart, inOrderRootIdx - 1);
        root.right = buildTree (preorder, inorder, preStart + inOrderRootIdx - inStart + 1,
preEnd, inOrderRootIdx + 1, inEnd);
        return root;
    }
}
```



By woshiamiaojiang

Not published yet.

Last updated 17th October, 2023.

Page 3 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>