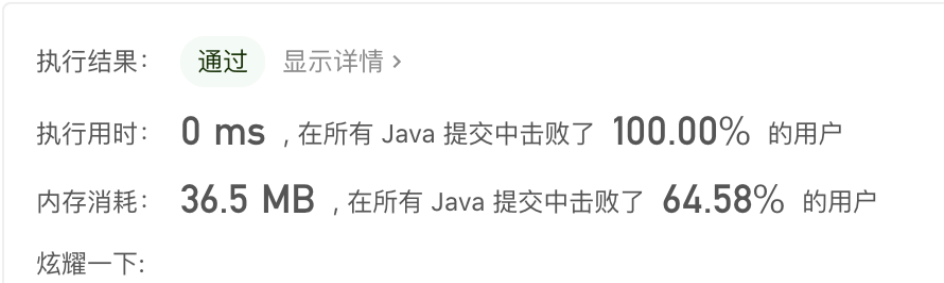
101-200

leecode 101-150

101 Symmetric Tree




```java
class Solution {
    public boolean isSymmetric(TreeNode root) {
        if(root == null)        return true;

        return isSymmetric(root.left, root.right) ;
    }

    public boolean isSymmetric(TreeNode node1, TreeNode node2){

        if(node1 == null || node2 == null){
            if(node1 != null || node2 != null)
                return false;
            return true;
        }

        if(node1.val != node2.val) return false;

        return isSymmetric(node1.left, node2.right)
            && isSymmetric(node1.right, node2.left);
    }
}
```

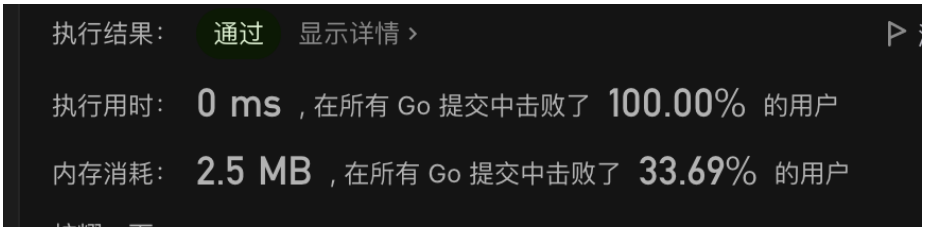
```cpp
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
```

```
        if(root == NULL)
            return true;

        return isSymmetric(root->left, root->right);
    }

    bool isSymmetric(TreeNode * left, TreeNode * right){
        if(left == NULL && right == NULL){
            return true;
        }

        if((left == NULL || right == NULL) || (left->val != right->val)){
            return false;
        }

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

```
class Solution {
public:
    bool checkNode(TreeNode * left, TreeNode * right){
        if((left == NULL && right == NULL))
            return true;

        if(left == NULL || right == NULL)
            return false;

        return left->val == right->val;
    }

    bool isSymmetric(TreeNode * root) {
        if(root == NULL){
            return true;
        }
        deque<TreeNode *> * dq = new deque<TreeNode *>();
        dq->push_back(root);

        bool flag = true;

        while(dq->size() != 0){
            if(flag){
                flag = false;
                TreeNode * cur =  dq->back();
                dq->pop_back();
                if(checkNode(cur->left, cur->right) == false)
```

```
                    return false;
                dq->push_front(cur->left);
                dq->push_back(cur->right);

            }else{
                TreeNode * left = dq->front();
                TreeNode * right = dq->back();
                dq->pop_back();
                dq->pop_front();

                if(checkNode(left, right) == false)
                    return false;

                if(left == NULL && right == NULL)
                     continue;

                dq->push_front(left->left);
                dq->push_back(right->right);


                dq->push_front(left->right);
                dq->push_back(right->left);
                }
            }


        return true;
    }
};
```

103 Binary Tree Zigzag Level Order Traversal




```
func zigzagLevelOrder(root *TreeNode) [][]int {
  res := make([][] int, 0)

  queue := make([]*TreeNode, 0)
  if root == nil{
```

```
    return res
  }
  queue = append(queue, root)
  odd   := true
  for ;len(queue) != 0;{
    size := len(queue)
    temp := make([]int, 0)
    for i := 0; i < size; i++{
      cur := queue[0]
      queue = queue[1:]

      temp = append(temp, cur.Val)
      if cur.Left != nil{
        queue = append(queue, cur.Left)
      }

      if cur.Right != nil{
        queue = append(queue, cur.Right)
      }
    }

    if !odd{
      for i, j := 0, len(temp) - 1; i < j;{
        temp[i], temp[j] = temp[j], temp[i]
                i++
                j--
      }
    }
    odd = !odd
    res = append(res, temp)
  }

  return res
}
```

104 Maximum Depth of Binary Tree

执行结果： 通过 显示详情 >

执行用时： 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 38.6 MB , 在所有 Java 提交中击败了 36.64% 的用户

```java
    public int maxDepth(TreeNode root) {
        return root == null ? 0 : (1 + Math.max(maxDepth(root.left), maxDepth(root.right)));
    }
```

105 Construct Binary Tree From Preorder and Inorder Traversal

105. Construct Binary Tree from Preorder and Inorder Traversal

难度 中等 557

Given preorder and inorder traversal of a tree, construct the binary tree.

Note:
You may assume that duplicates do not exist in the tree.

For example, given

```
preorder = [3,9,20,15,7]
inorder = [9,3,15,20,7]
```

Return the following binary tree:

```
    3
   / \
  9  20
    /  \
   15   7
```

执行结果： 通过 显示详情 >

执行用时： 10 ms , 在所有 Java 提交中击败了 13.11% 的用户

内存消耗： 38.8 MB , 在所有 Java 提交中击败了 43.35% 的用户

炫耀一下：

```java
  public TreeNode buildTree(int[] preorder, int[] inorder) {
        if(preorder.length == 0)    return null;
        assert(preorder.length == inorder.length);

        int i = 0;
        for(; i < inorder.length; i++)
            if(inorder[i] == preorder[0])
                break;

        int[] preLeft  = Arrays.copyOfRange(preorder, 1, i + 1);
        int[] preRight = Arrays.copyOfRange(preorder, i + 1, preorder.length);
        int[] inLeft   = Arrays.copyOfRange(inorder, 0, i);
        int[] inRight  = Arrays.copyOfRange(inorder, i + 1, inorder.length);

        TreeNode root = new TreeNode(preorder[0]);

        root.left  = buildTree(preLeft, inLeft);
        root.right = buildTree(preRight, inRight);

        return root;
    }
```

```java
/*
  这个递归写的蛮漂亮的
*/
class Solution {
  public TreeNode buildTree(int[] preorder, int[] inorder) {
    if(preorder.length==0 || inorder.length==0) {
      return null;
    }
    //根据前序数组的第一个元素，就可以确定根节点
    TreeNode root = new TreeNode(preorder[0]);

    for(int i=0;i<preorder.length;++i) {
```

```
      //用preorder[0]去中序数组中查找对应的元素
      if(preorder[0]==inorder[i]) {
        //将前序数组分成左右两半，再将中序数组分成左右两半
        //之后递归的处理前序数组的左边部分和中序数组的左边部分
        //递归处理前序数组右边部分和中序数组右边部分
        int[] pre_left = Arrays.copyOfRange(preorder,1,i+1);
        int[] pre_right = Arrays.copyOfRange(preorder,i+1,preorder.length);
        int[] in_left = Arrays.copyOfRange(inorder,0,i);
        int[] in_right = Arrays.copyOfRange(inorder,i+1,inorder.length);
        root.left = buildTree(pre_left,in_left);
        root.right = buildTree(pre_right,in_right);
        break;
      }
    }
    return root;
  }
}

作者：wang_ni_ma
链接：https://leetcode-cn.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/solution/dong-hua-yan-shi-105-cong-qian-xu-yu-zhong-xu-bian/
```

106 Construct Binary Tree From Inorder And Postorder

106. Construct Binary Tree from Inorder and Postorder Traversal

难度 中等 235

Given inorder and postorder traversal of a tree, construct the binary tree.

Note:
You may assume that duplicates do not exist in the tree.

For example, given

```
inorder = [9,3,15,20,7]
postorder = [9,15,7,20,3]
```

Return the following binary tree:

```
    3
   / \
  9  20
    /  \
   15   7
```

通过次数 41,464 | 提交次数 60,160

106. Construct Binary Tree from Inorder and Postorder Traversal

难度 中等 235

Given inorder and postorder traversal of a tree, construct the binary tree.

Note:
You may assume that duplicates do not exist in the tree.

For example, given

```
inorder = [9,3,15,20,7]
postorder = [9,15,7,20,3]
```

Return the following binary tree:

```
    3
   / \
  9  20
    /  \
   15   7
```

执行结果： 通过 显示详情 >

执行用时： 11 ms ，在所有 Java 提交中击败了 12.28% 的用户

内存消耗： 38.8 MB ，在所有 Java 提交中击败了 76.06% 的用户

炫耀一下：

```java

class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder) {
        if(postorder.length == 0)    return null;
        //assert(preorder.length == inorder.length);

        int i = 0;
        for(; i < inorder.length; i++)
            if(inorder[i] == postorder[postorder.length - 1])
                break;

        int[] postLeft   = Arrays.copyOfRange(postorder, 0, i);
        int[] postRight  = Arrays.copyOfRange(postorder, i, postorder.length - 1);
        int[] inLeft     = Arrays.copyOfRange(inorder, 0, i);
        int[] inRight    = Arrays.copyOfRange(inorder, i + 1, inorder.length);

        TreeNode root = new TreeNode(postorder[postorder.length - 1]);

        root.left  = buildTree(inLeft, postLeft);
        root.right = buildTree(inRight, postRight);

        return root;
    }
}
```

```java
/*
  和上一个题，异曲同工之秒
*/
public TreeNode buildTree(int[] inorder, int[] postorder) {
    if(inorder.length == 0 || postorder.length == 0)
        return null;

    TreeNode root = new TreeNode(postorder[postorder.length-1]);

    for(int i = 0; i < postorder.length; i++)
    {
        if(postorder[postorder.length-1] == inorder[i])
```

```
        {
            int[] in_left = Arrays.copyOfRange(inorder,0,i);
            int[] in_right = Arrays.copyOfRange(inorder, i+1,inorder.length);
            int[] po_left = Arrays.copyOfRange(postorder, 0,i);
            int[] po_right = Arrays.copyOfRange(postorder,i,postorder.length-1);

            root.left = buildTree(in_left, po_left);
            root.right = buildTree(in_right, po_right);
            break;
        }
    }
    return root;
}
```

108 Convert Sorted Array To BST

108. Convert Sorted Array to Binary Search Tree

难度 简单 510

Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

Example:

```
Given the sorted array: [-10,-3,0,5,9],

One possible answer is: [0,-3,9,-10,null,5], which
represents the following height balanced BST:

      0
     / \
   -3   9
   /   /
 -10  5
```

执行结果: 通过 显示详情 >

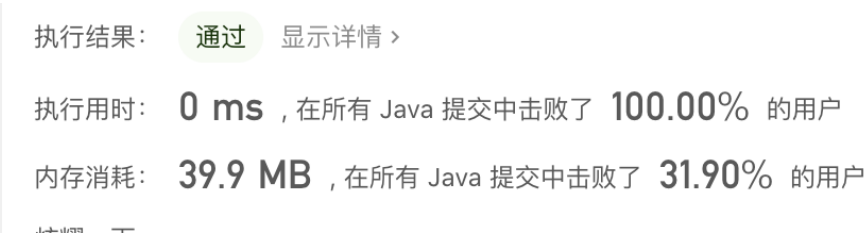
执行用时: 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 40.1 MB , 在所有 Java 提交中击败了 8.70% 的用户

炫耀一下:

```java
public TreeNode sortedArrayToBST(int[] nums) {
    if(nums.length == 0)        return null;
    if(nums.length == 1)          return new TreeNode(nums[0]);

    int mid = nums.length / 2;
    TreeNode root = new TreeNode(nums[mid]);

    int[] left_part = Arrays.copyOfRange(nums, 0, mid);
    int[] right_part = Arrays.copyOfRange(nums, mid+1, nums.length);
    root.left = sortedArrayToBST(left_part);
    root.right = sortedArrayToBST(right_part);

    return root;
}
```

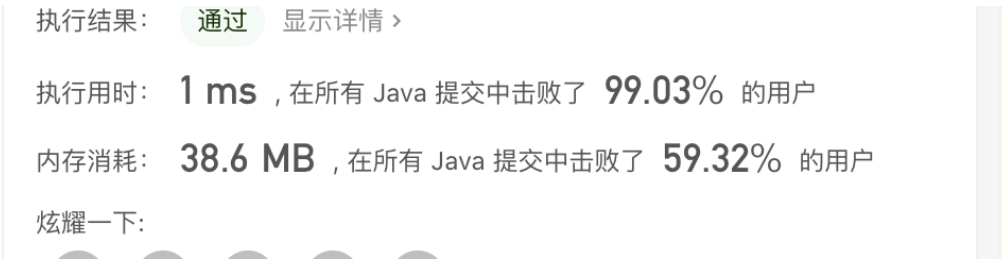
109 Convert Sorted List to Binary Search Tree




```java
class Solution {
    public TreeNode sortedListToBST(ListNode head) {
        return helper(head, null);
    }

    public TreeNode helper(ListNode head, ListNode tail){
        if(head == tail)                    return null;
        if(head.next == tail)               return new TreeNode(head.val);

        ListNode slow = head;
        ListNode fast = head;

        while(fast != tail && fast.next != tail){
            fast = fast.next.next;
            slow = slow.next;
        }
```

```
        TreeNode root = new TreeNode(slow.val);

        root.left  = helper(head, slow);
        root.right = helper(slow.next, tail);

        return root;
    }
}

```

110 Balanced Binary Tree




```
/*
  注意这里拿的一定是最大深度
*/
class Solution {
    public boolean isBalanced(TreeNode root) {
        if(root == null)    return true;

        return Math.abs(maxDepth(root.left) - maxDepth(root.right)) <= 1 &&
            isBalanced(root.left) && isBalanced(root.right);
    }

    public int maxDepth(TreeNode root){
        return root == null ? 0 : (1 + Math.max(maxDepth(root.left),
maxDepth(root.right)));
    }
}
```

111 Minimum Depth of Binary Tree

111. Minimum Depth of Binary Tree

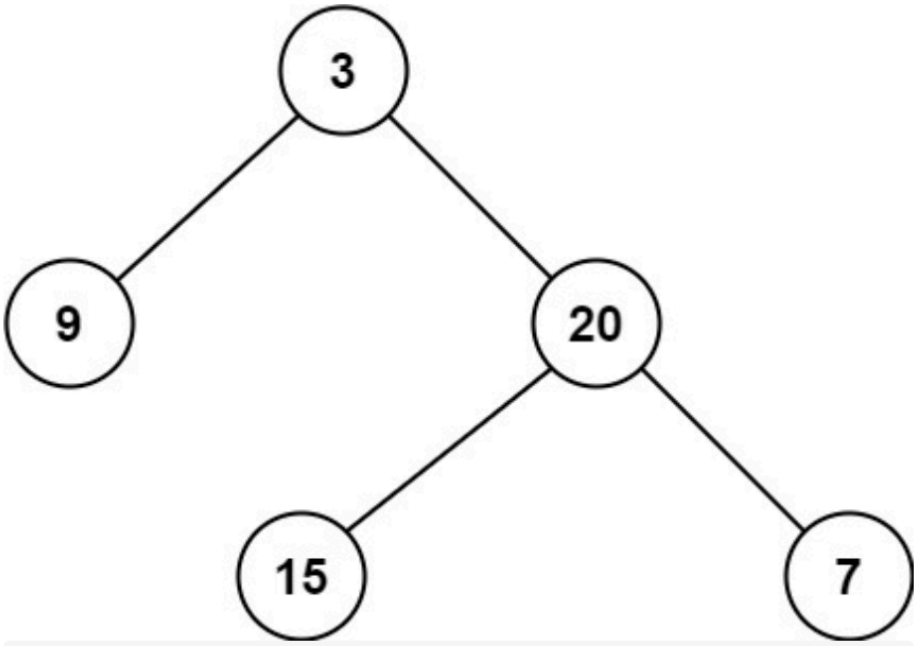
难度 简单 423

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

Note: A leaf is a node with no children.

Example 1:




```
Input: root = [3,9,20,null,null,15,7]
```

执行用时： 8 ms ，在所有 Java 提交中击败了 28.62% 的用户

内存消耗： 59.3 MB ，在所有 Java 提交中击败了 19.10% 的用户

```java
    public int minDepth(TreeNode root) {
        if(root == null)                                return 0;
        if(root.left == null || root.right == null){
            if(root.left == null && root.right == null)
                return 1;
            else
                return 1 + (root.left == null ? minDepth(root.right) :
minDepth(root.left));
        }

        return 1 + Math.min(minDepth(root.left), minDepth(root.right));
    }
```

```cpp
/*
    BFS 方法
*/
class Solution {
public:
    int minDepth(TreeNode *root) {
        if (root == nullptr) {
            return 0;
        }

        queue<pair<TreeNode *, int> > que;
        que.emplace(root, 1);

        while (!que.empty()) {
            TreeNode *node = que.front().first;
            int depth = que.front().second;
            que.pop();
            if (node->left == nullptr && node->right == nullptr) {
                return depth;
            }
            if (node->left != nullptr) {
                que.emplace(node->left, depth + 1);
            }
            if (node->right != nullptr) {
                que.emplace(node->right, depth + 1);
            }
        }

        return 0;
    }
```

```
};

作者：LeetCode-Solution
链接：https://leetcode-cn.com/problems/minimum-depth-of-binary-tree/solution/er-cha-shu-de-zui-xiao-shen-du-by-leetcode-solutio/
```

112 Path Sum

112. Path Sum

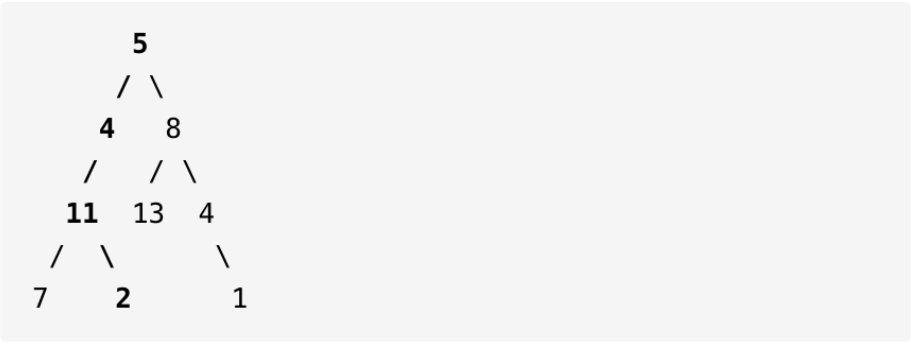
难度 简单 485

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

Note: A leaf is a node with no children.

Example:

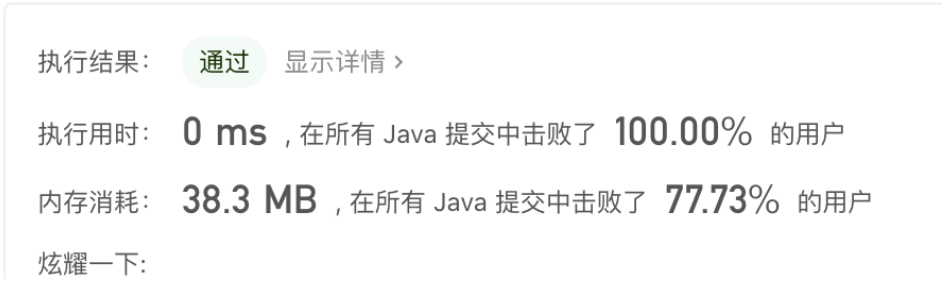
Given the below binary tree and `sum = 22`,



return true, as there exist a root-to-leaf path `5->4->11->2` which sum is 22.

```java
//二刷 狗
  public boolean hasPathSum(TreeNode root, int targetSum) {
        return getRes(root, targetSum);
    }

    public boolean getRes(TreeNode root, int remains){
        if(root == null)
            return false;
        else if(root.left == null && root.right == null){
            if(root.val == remains)
                return true;
            else
```

```
                return false;
        }

        if(getRes(root.left, remains - root.val) ||
            getRes(root.right, remains - root.val))
            return true;

        return false;
    }
```




```
    public boolean hasPathSum(TreeNode root, int sum) {
        if(root == null)                            return false;
        if(root.left == null && root.right == null) return root.val == sum;

        return dfs(root, sum);
    }

    public boolean dfs(TreeNode root, int remains){
        if(remains == 0){
            if(root == null)
                return true;
        }
        if(root == null)        return false;

        if(root.left == null && root.right == null)
            return remains - root.val == 0;

        boolean l = false;
        if(root.left != null)
            l = dfs(root.left, remains - root.val);
        boolean r = false;
        if(root.right != null)
            r = dfs(root.right, remains - root.val);

        return l || r;
    }
```

```java
//这个思路不错
public class Solution {
    public boolean hasPathSum(TreeNode root, int sum) {
        if(root == null){
            return false;
        }
        if(root.left == null && root.right == null){
            return root.val == sum;
        }
        return hasPathSum(root.left, sum - root.val) || hasPathSum(root.right, sum - root.val);

    }
}
```

作者：fuxuemingzhu
链接：https://leetcode-cn.com/problems/path-sum/solution/lu-jing-zong-he-de-si-chong-jie-fa-dfs-hui-su-bfs-/
来源：力扣（LeetCode）

113 Path Sum II 回溯不错的题目

113. Path Sum II

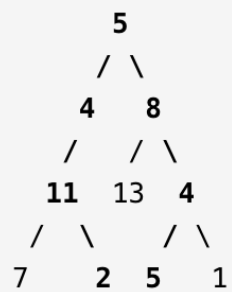
难度 中等 396

Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.

Note: A leaf is a node with no children.

Example:

Given the below binary tree and `sum = 22`,



```
      5
     / \
    4   8
   /   / \
  11  13  4
 /  \    / \
7    2  5   1
```

Return:

```
[
   [5,4,11,2],
   [5,8,4,5]
]
```

执行结果: 通过 显示详情 >

执行用时: 1 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 38.9 MB , 在所有 Java 提交中击败了 80.64% 的用户

```java
class Solution {
    List<List<Integer>> res = new ArrayList<>();
    public List<List<Integer>> pathSum(TreeNode root, int sum) {
        if(root == null)        return res;

        dfs(new ArrayList<>(), sum, root);
        return res;
    }

```

```java
    public void dfs(List<Integer> path, int remains, TreeNode root){
        if(root == null)    return;

        if(remains == root.val && root.left == null && root.right == null){
            path.add(root.val);
            res.add(new ArrayList<>(path));
            path.remove(path.size() - 1);  /*注意回溯过程中及时移除元素*/
            return;
        }

        path.add(root.val);
        dfs(path, remains - root.val, root.left);
        dfs(path, remains - root.val, root.right);
        path.remove(path.size() - 1);

    }
}

```

114 Flatten Binary Tree to Linked List

执行结果： 通过 显示详情 >

执行用时： 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 37.7 MB , 在所有 Java 提交中击败了 91.52% 的用户

```java
class Solution {
    public void flatten(TreeNode root) {
        if(root == null)        return;

        TreeNode temp = root.left;
        root.left     = root.right;
        root.right    = temp;

        TreeNode cur = root;
        while(cur.right != null)
            cur = cur.right;

        cur.right = root.left;
        root.left = null;

        flatten(root.right);
```

```
    }
}

    1
   / \
  2   5
 / \   \
3   4   6

//将 1 的左子树插入到右子树的地方
    1
  /   \
 5     2
  \   / \
   6  3   4
//将原来的右子树接到左子树的最右边节点
    1
     \
      2
     / \
    3   4
         \
          5
           \
            6

 //将 2 的左子树插入到右子树的地方
    1
     \
      2
       \
        3       4
                 \
                  5
                   \
                    6

 //将原来的右子树接到左子树的最右边节点
    1
     \
      2
       \
        3
         \
          4
           \
            5
             \
              6
```

```
66
```

116 / 117 Populating Next Pointers in Each Node

116. Populating Next Right Pointers in Each Node

难度 中等 362

You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to `NULL`.

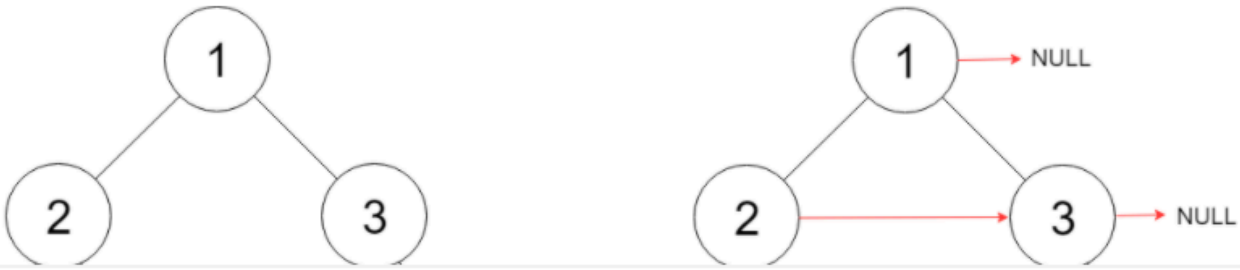
Initially, all next pointers are set to `NULL`.

Follow up:

- You may only use constant extra space.

- Recursive approach is fine, you may assume implicit stack space does not count as extra space for this problem.

Example 1:



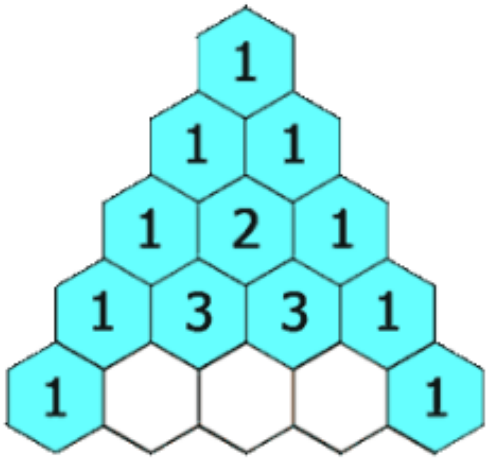

```java
class Solution {
    public Node connect(Node root) {
        if(root == null)    return null;
        Deque<Node> queue = new ArrayDeque<>();

        queue.add(root);

        while(!queue.isEmpty()){
            int size = queue.size();
            for(int i = 0; i < size; i++){
                Node cur = queue.pollFirst();
                if(i == size - 1)
                    cur.next = null;
                else
                    cur.next = queue.peekFirst();

                if(cur.left != null)
                    queue.add(cur.left);
                if(cur.right != null)
                    queue.add(cur.right);
            }
        }

        return root;
    }
}
```

118 Pascal's Triangle

118. Pascal's Triangle

难度 简单 327

Given a non-negative integer numRows, generate the first numRows of Pascal's triangle.



In Pascal's triangle, each number is the sum of the two numbers directly above it.

Example:

```
Input: 5
Output:
[
     [1],
    [1,1],
   [1,2,1],
  [1,3,3,1],
 [1,4,6,4,1]
]
```

执行结果: 通过 显示详情 >

执行用时: 1 ms , 在所有 Java 提交中击败了 77.33% 的用户

内存消耗: 37.2 MB , 在所有 Java 提交中击败了 9.09% 的用户

炫耀一下:

```
/*
[
  [1],
  [1,1],
  [1,2,1],
  [1,3,3,1],
```

```
  [1,4,6,4,1]
]
*/
public List<List<Integer>> generate(int numRows) {
    List<List<Integer>> res = new ArrayList<>();

    for(int i = 0; i < numRows; i++)
    {
        List<Integer> path = new LinkedList<>();
        for(int j = 0; j < i+1; j++)
        {
            if(j == 0 || j == i || i == 0)
                path.add(1);
            else
                path.add(res.get(i-1).get(j) + res.get(i-1).get(j-1));
        }
        res.add(path);
    }
    return res;
}
```

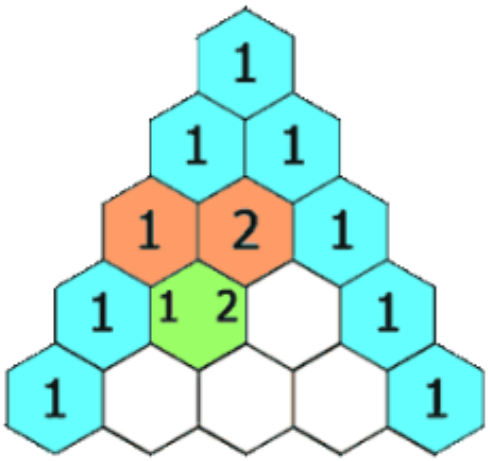
119 Pascal's Triangle II

119. Pascal's Triangle II

难度 简单 160

Given a non-negative index k where k ≤ 33, return the k^{th} index row of the Pascal's triangle.

Note that the row index starts from 0.



In Pascal's triangle, each number is the sum of the two numbers directly above it.

Example:

```
Input: 3
Output: [1,3,3,1]
```

Follow up:

Could you optimize your algorithm to use only O(k) extra space?

通过次数 58,532 | 提交次数 95,489

```go
func getRow(rowIndex int) []int {
  res := make([]int, 0)

  if rowIndex == 0{
    res = append(res, 1)
    return res
  }else if rowIndex == 1{
    res = append(res, 1)
    res = append(res, 1)
    return res
  }

  temp := make([]int, 0)
  temp = append(temp, 1)
  temp = append(temp, 1)

```

```
  for i := 2; i <= rowIndex; i++{
    res = append(res, 1)

    for j := 0; j < len(temp) - 1; j++{
      t := temp[j] + temp[j + 1]
      res = append(res, t)
    }

    res = append(res, 1)

    temp = make([]int, len(res))
    copy(temp, res)
    res = res[:0]
  }

  return temp
}
```

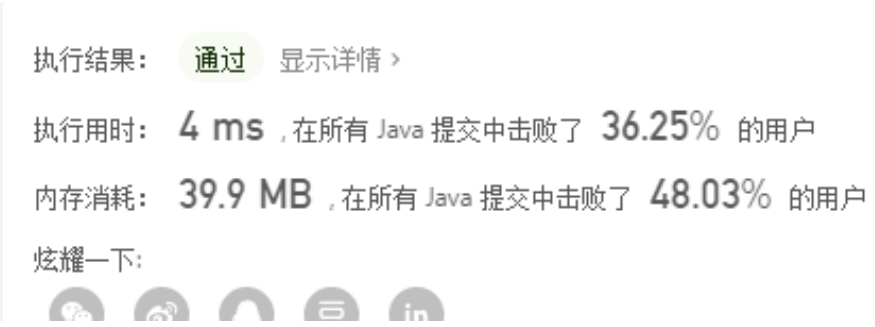
```
/*
  笨办法，拿到所有行，返回最后一行
*/
public List<Integer> getRow(int rowIndex) {
    List<List<Integer>> res = new ArrayList<>();

    for(int i = 0; i <= rowIndex; i++)
    {
        List<Integer> path = new LinkedList<>();
        for(int j = 0; j < i+1; j++)
        {
            if(j == 0 || j == i || i == 0)
                path.add(1);
            else
                path.add(res.get(i-1).get(j) + res.get(i-1).get(j-1));
        }
        res.add(path);
    }
    return res.get(rowIndex);
}
```

```
/*
  动态规划
*/
public List<Integer> getRow(int rowIndex) {
```

```
    List<Integer> list = new ArrayList<>();
    while (rowIndex-- >= 0) {
        list.add(1);
        //原地更新，属实牛逼
        for (int i = list.size() - 2; i > 0; i--)
            list.set(i, list.get(i) + list.get(i - 1));
    }

    return list;
}
https://leetcode-cn.com/problems/pascals-triangle-ii/solution/119yang-hui-san-jiao-ii-dong-tai-gui-hua-by-ceng-j/
```

120 Triangle

```
func minimumTotal(triangle [][]int) int {
  row := len(triangle)

  for i := row - 2; i >= 0; i--{
    for j := 0; j < len(triangle[i]); j++{
      triangle[i][j] = min(triangle[i + 1][j], triangle[i + 1][j + 1]) + triangle[i][j]
    }
  }

  return triangle[0][0]
}

func min(a int, b int) int{
  if a < b{
    return a
  }

  return b
}
```



```java
/*
  典型DP
*/
    public int minimumTotal(List<List<Integer>> triangle) {
        if(triangle.size() == 0)        return 0;
        int[][] dp = new int[triangle.size()][triangle.get(triangle.size() - 1).size()];
        for(int k = 0; k < triangle.get(triangle.size() - 1).size(); k++)
            dp[triangle.size() - 1][k] = triangle.get(triangle.size() - 1).get(k);
        for(int i = triangle.size() - 2; i >= 0; i--)
        {
            for(int j = 0; j < triangle.get(i).size(); j++)
            {
                if(j == 0)
                    dp[i][j] = Math.min(dp[i + 1][j], dp[i+1][j+1]) + triangle.get(i).get(j);
                else
                    dp[i][j] = Math.min(dp[i+1][j], dp[i+1][j+1]) +
                        triangle.get(i).get(j);
            }
        }


        return dp[0][0];
    }
```

124 Binary Tree Maximum Path Sum

124. Binary Tree Maximum Path Sum

难度 困难 585

Given a **non-empty** binary tree, find the maximum path sum.

For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain **at least one node** and does not need to go through the root.

Example 1:

```
Input: [1,2,3]

       1
      / \
     2   3

Output: 6
```

Example 2:

```
Input: [-10,9,20,null,null,15,7]

   -10
   / \
  9  20
    /  \
   15   7

Output: 42
```

```java
/*
  全局变量和递归结合的题目

  对这个题做一个深度剖析
*/
class Solution {
    int max = Integer.MIN_VALUE;
    public int maxPathSum(TreeNode root) {
        helper(root);
        return max;
    }

  //该函数返回的是包含当前节点的最大单边路径
    private int helper(TreeNode root)
    {
```

```
16          if(root == null)        return 0;
17
18          int left = Math.max(helper(root.left)  , 0);
19          int right = Math.max(helper(root.right), 0);
20
21          //这里更新 包含横跨我们root 的max 路径,  包含左右
22          max = Math.max(max, root.val + left + right);
23
24            //这里返回单边路径
25          return root.val + Math.max(left, right);
26      }
27  }
```

125 Valid Palindrome

125. Valid Palindrome

难度 简单 245

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

```
Input: "A man, a plan, a canal: Panama"
Output: true
```

Example 2:

```
Input: "race a car"
Output: false
```

Constraints:

- `s` consists only of printable ASCII characters.


```java
public boolean isPalindrome(String s) {
    char[] chars = s.toCharArray();
    HashSet<Character> set = new HashSet<>();
    HashSet<Character> SET = new HashSet<>();
    for(char ch = 'a'; ch <= 'z'; ch++)
        set.add(ch);
    for(char ch = '0'; ch <= '9'; ch++)
        set.add(ch);
    for(char ch = 'A'; ch <= 'Z'; ch++)
        SET.add(ch);

    int left = 0;
    int right = chars.length-1;

    while(left < right)
    {
        while(left <= right && !set.contains(chars[left]) &&
!SET.contains(chars[left]))
            left++;
        while(left <= right && !set.contains(chars[right]) &&
!SET.contains(chars[right]))
            right--;

        if(left >= right)    break;

        if(SET.contains(chars[left]))
            chars[left] += 32;
        if(SET.contains(chars[right]))
            chars[right] += 32;

        if(chars[right] != chars[left])
            return false;
        left++;
        right--;
    }
    return true;
}
```

126 Word ladder II 回溯 + bfs综合

```java
class Solution {
    List<List<String>> graph = new ArrayList<>();
    List<List<String>> res = new ArrayList<>();
    HashSet<String> marked = new HashSet<>();

    public List<List<String>> findLadders(String beginWord, String endWord, List<String> wordList) {
        if(!wordList.contains(endWord))
            return res;
        wordList.remove(beginWord);
        HashSet<String> wordSet = new HashSet<>();
        wordSet.addAll(wordList);

        ArrayDeque<String> queue = new ArrayDeque<>();
        queue.addLast(beginWord);
        graph = construcGraph(queue, wordSet);
        List<String> path =  new ArrayList<>();
        path.add(beginWord);

        backtrack(beginWord, endWord, path, 0);

        return res;
    }

    private void backtrack(String beginWord, String endWord, List<String> path, int level)
    {
        if(beginWord.equals(endWord))
        {
            res.add(new ArrayList<>(path));
            return;
        }

        for(int i = 0; level < graph.size() && i < graph.get(level).size(); i++)
        {
            String s = graph.get(level).get(i);
            if(!isDifOne(s, path.get(path.size()-1)))
                continue;
            path.add(s);

            backtrack(s, endWord, path, level+1);

            path.remove(path.size()-1);
        }
```

```
43
44      }
45
46      private boolean isDifOne(String a, String b) {
47          int count = 0;
48          for(int i = 0; i < a.length(); i++)
49              if(a.charAt(i) != b.charAt(i))
50                  count++;
51          return  count == 1;
52      }
53
54      private List<List<String>> construcGraph(ArrayDeque<String> queue,
    HashSet<String> wordList)
55      {
56          while(!queue.isEmpty())
57          {
58              int size = queue.size();
59              List<String> path = new ArrayList<>();
60
61              for(int i = 0; i < size; i++)
62              {
63                  String temp = queue.removeFirst();
64                  path.add(temp);
65                  char[] chars = temp.toCharArray();
66                  for(int j = 0; j < chars.length; j++)
67                  {
68                      char ch = chars[j];
69                      for(char c = 'a'; c <= 'z'; c++)
70                      {
71                          if(c == ch)
72                              continue;
73                          chars[j] = c;
74                          String cs = toStr(chars);
75                          if(wordList.contains(cs) && !marked.contains(cs))
76                          {
77                              queue.addLast(cs);
78                              marked.add(cs);
79                          }
80                      }
81                      chars[j] = ch;
82                  }
83              }
84              graph.add(path);
85          }
86          graph.remove(0);
87          return graph;
88      }
89
90      private String toStr(char[] chars)
```

```
    {
        StringBuilder sb = new StringBuilder();
        for(char ch : chars)
            sb.append(ch);
        return sb.toString();
    }

}

作者：venturekwok
链接：https://leetcode-cn.com/problems/word-ladder-ii/solution/java-jie-gou-qing-xi-shi-pin-jiang-jie-zhu-yao-zhi/
```

128 Longest Consecutive Sequence

128. Longest Consecutive Sequence

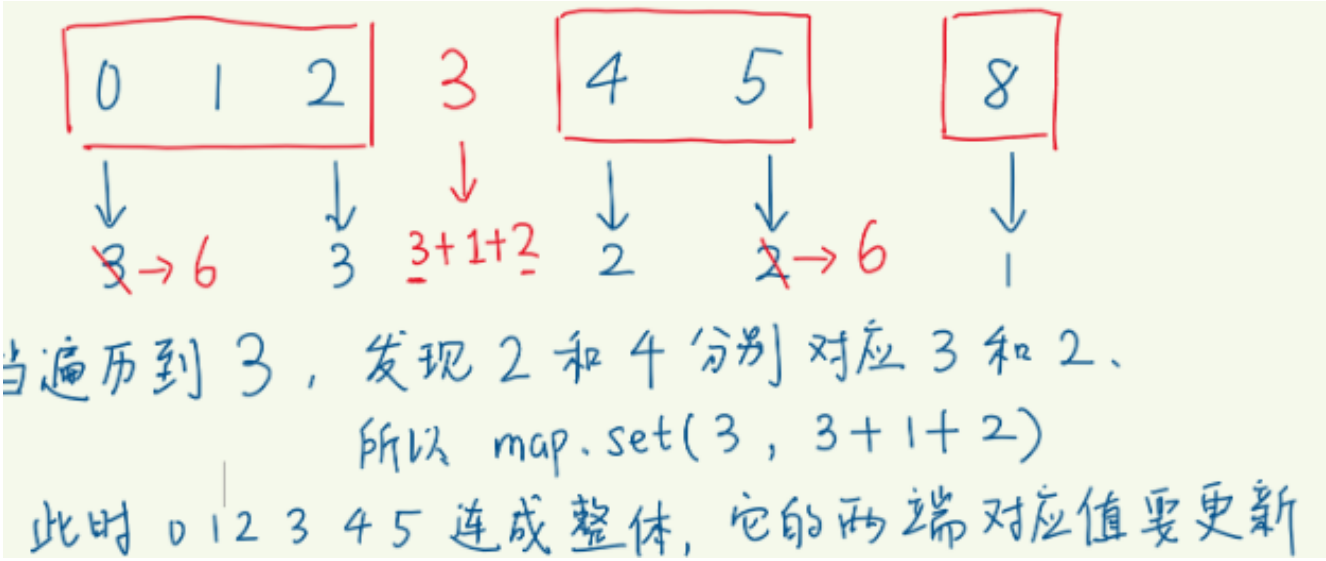
难度 困难 455

Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

Your algorithm should run in O(n) complexity.

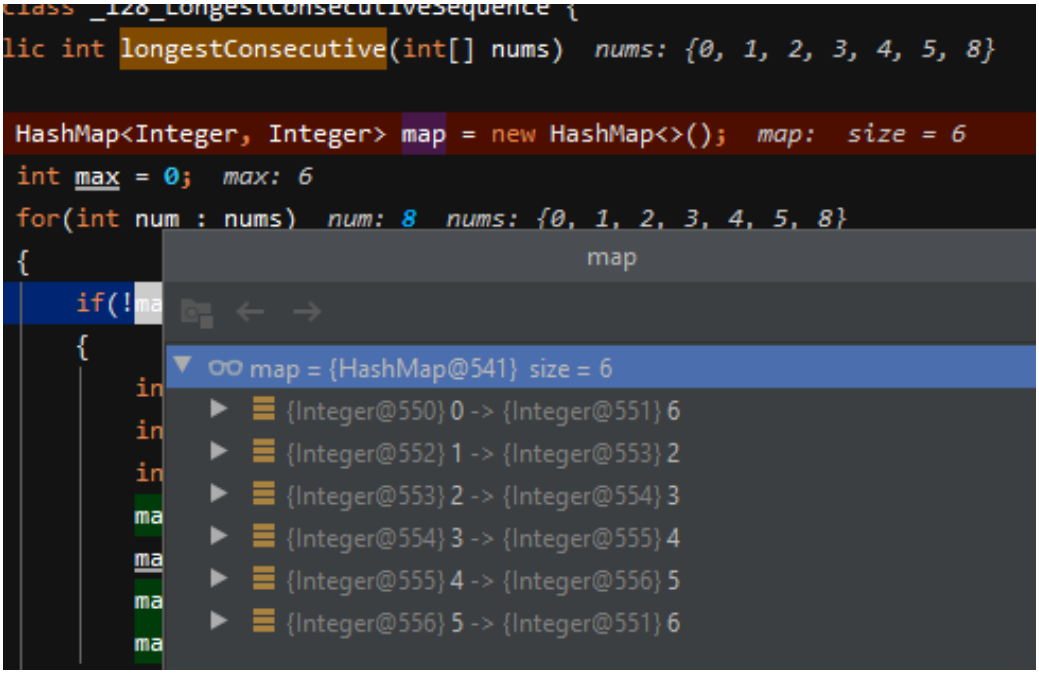
Example:

```
Input: [100, 4, 200, 1, 3, 2]
Output: 4
Explanation: The longest consecutive elements sequence is
[1, 2, 3, 4]. Therefore its length is 4.
```



方法二：

```java
/*哈希表的value存什么
key存数字，value存什么?
新存入的数字，如果它找到相邻的数，它希望从邻居数那里获取什么信息?
很显然它希望，左邻居告诉它左边能提供的连续长度，右邻居告诉它右边能提供的连续长度
加上它自己的长度，就有了自己处在的连续序列的长度

作者：hyj8
链接：https://leetcode-cn.com/problems/longest-consecutive-sequence/solution/fang-fa-cong-yi-dao-nan-bing-cha-ji-fang-fa-bu-hui/
*/

/*
  这是动态规划的方法，注意理解
*/
public int longestConsecutive(int[] nums)
{
  HashMap<Integer, Integer> map = new HashMap<>();
    int max = 0;
    for(int num : nums)
    {
        if(!map.containsKey(num))
        {
            int preLen = map.get(num-1) == null ? 0 : map.get(num-1);
            int nextLen = map.get(num+1) == null ? 0 : map.get(num+1);
            int curLen = preLen + 1 + nextLen;
            map.put(num, curLen);
            max = Math.max(max, curLen);
            map.put(num - preLen, curLen);    //对序列左边进行更新
            map.put(num + nextLen, curLen);   //对序列右边进行更新
        }
    }
    return max;
}
```



可以看到，运行后，map 连续序列中只记录两边的，并不断更新，而该序列比如1,2,3,4,等的值并没有同样更新为6

129 Sum Root to Leaf Numbers

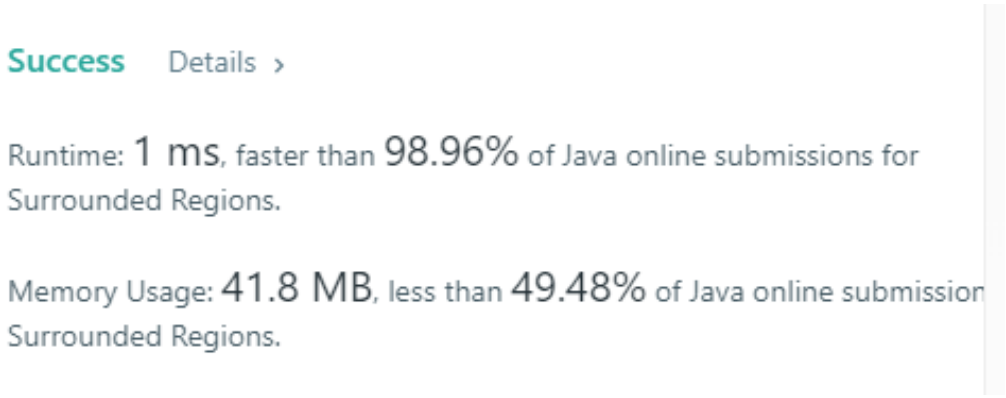

```java
/*
    整体思路采用回溯,

    记得在每一步都进行
        做出选择
        进入回溯
        撤销选择
      包括在特殊判定部分
*/
class Solution {
    int res = 0;
```

```java
    public int sumNumbers(TreeNode root) {
        dfs(root, new StringBuilder());
        return res;
    }

    private void dfs(TreeNode root, StringBuilder path){
        if(root == null)    return;
        if(root.left == null && root.right == null){
            path.append(root.val);
            res += Integer.parseInt(path.toString());
             path.setLength(path.length() - 1);
            return;
        }

        path.append(root.val);

        dfs(root.left , path);
        dfs(root.right, path);

        path.setLength(path.length() - 1);
    }
}
```

130 Surrounded Regions


```java
class Solution {
    int row =  0;
```

```
    int column = 0;
    public void solve(char[][] board) {
        row = board.length;
        column = row == 0 ? 0 : board[0].length;
        if(column == 0)     return;

        for(int i = 0; i < row; i++){
            dfs(board, i, 0);
            dfs(board, i, column - 1);
        }

        for(int j = 0; j < column; j++){
            dfs(board, 0, j);
            dfs(board, row - 1, j);
        }

        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(board[i][j] == '@')
                    board[i][j] = 'O';
                else
                    board[i][j] = 'X';

    }

    private boolean isInRange(int i, int j){
        return i >= 0 && j >= 0 && i < row && j < column;
    }

    public void dfs(char[][] board, int i, int j){
        if(isInRange(i, j) && board[i][j] == 'O'){
            board[i][j] = '@';

            dfs(board, i + 1, j);
            dfs(board, i - 1, j);
            dfs(board, i, j + 1);
            dfs(board, i, j - 1);
        }
    }
}
```




```java
private char[][] board;
public void solve(char[][] board) {
    this.board = board;
    int row = board.length;
    if(row == 0)        return;
    int column = board[0].length;

    for(int i = 0; i < row; i++)
        for(int j = 0; j < column; j++)
        {
            if(isInEdge(i,j))
                if(board[i][j] == 'O')
                    dfs(board, i, j);
        }

    for(int i = 0; i < row; i++)
        for(int j = 0; j < column; j++)
        {
            if(board[i][j] == 'O')
                board[i][j] = 'X';
            else if(board[i][j] == '@')
                board[i][j] = 'O';
        }
}

private void dfs(char[][] board, int i, int j)
{
    if(i < 0 || j < 0 || i >= board.length || j >= board[0].length)
        return;

    if(board[i][j] == 'O')
        board[i][j] = '@';
    else
        return;

    dfs(board, i, j+1);
    dfs(board, i+1, j);
    dfs(board, i, j-1);
```

```
    dfs(board, i-1, j);

}

private boolean isInEdge(int i, int j)
{
    return i == 0 || i == board.length - 1 || j == 0 || j == board[0].length-1;
}
```

执行结果： 通过 显示详情 >

执行用时： 19 ms , 在所有 Java 提交中击败了 5.08% 的用户

内存消耗： 40.5 MB , 在所有 Java 提交中击败了 71.44% 的用户

炫耀一下：

```
/*
    Union Find 这个思路蛮好的
*/
class Solution {
    int row;
    int column;
    public void solve(char[][] board) {
         row    = board.length;
         column = row == 0 ? 0  : board[0].length;

        if(column == 0) return;
        UnionFind uf = new UnionFind(row * column + 1);
        int dummy = row * column;

        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(board[i][j] == 'O'){
                    if(i == 0 || j == 0 || i == row - 1 || j == column - 1)
                        uf.union(dummy, i * column + j);
                    else{
                        if (i > 0 && board[i - 1][j] == 'O')
                            uf.union(node(i, j), node(i - 1, j));
                        if (i < row - 1 && board[i + 1][j] == 'O')
                            uf.union(node(i, j), node(i + 1, j));
                        if (j > 0 && board[i][j - 1] == 'O')
                            uf.union(node(i, j), node(i, j - 1));
                        if (j < column - 1 && board[i][j + 1] == 'O')
                            uf.union(node(i, j), node(i, j + 1));
                    }
```

```
                }

        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(board[i][j] == 'O')
                    if(!uf.connected(dummy, i * column + j))
                        board[i][j] = 'X';

    }

    int node(int i, int j) {
        return i * column + j;
    }
}

//手写并查集
class UnionFind{
    private int[] size;
    private int[] id;
    private int count;

    public UnionFind(){}
    public UnionFind(int N){
        count = N;
        id = new int[N];
        size = new int[N];

        for(int i = 0; i < N; i++){
            id[i]   = i;
            size[i] = 1;
        }
    }

    public boolean connected(int p, int q){
        return find(p) == find(q);
    }

    private int find(int p){
        while(p != id[p]){
            id[p] = id[id[p]];
            p = id[p];
        }

        return p;
    }

    public void union(int p, int q){
        int pRoot = find(p);
        int qRoot = find(q);
```

```
79
80          if(pRoot == qRoot)  return;
81
82          if(size[p] > size[q]){
83              id[qRoot] = pRoot;
84              size[p] += size[q];
85          }else{
86              id[pRoot] = qRoot;
87              size[q] += size[p];
88          }
89
90          count--;
91      }
92
93      public int count(){return count;}
94  }
```

```
1   /*
2       BFS 方法
3   */
4       int row, column;
5       const int dirx[4] = {1, -1, 0, 0};
6       const int diry[4] = {0, 0, 1, -1};
7       void solve(vector<vector<char>>& board) {
8           row = board.size();
9           column = row == 0 ? 0 : board[0].size();
10          if(column == 0)     return;
11
12          queue<pair<int, int>> que;
13          for(int i = 0; i < row; i++)
14              for(int j = 0; j < column; j++)
15                  if(board[i][j] == 'O')
16                      if(i == 0 || j == 0 || i == row - 1 || j == column - 1)
17                          que.emplace(i, j);
18          while(!que.empty()){
19              int x = que.front().first;
20              int y = que.front(). second;
21              que.pop();
22
23              board[x][y] = '@';
24              for(int k = 0; k < 4; k++){
25                  int newX = x + dirx[k];
26                  int newY = y + diry[k];
27                  if(newX < 0 || newY < 0 || newX >= row || newY >= column ||
    board[newX][newY] != 'O')
28                      continue;
29
```

```
                que.emplace(newX, newY);
            }
        }

        for (int i = 0; i < row; i++)
            for (int j = 0; j < column; j++)
                if (board[i][j] == '@')
                    board[i][j] = 'O';
                 else if (board[i][j] == 'O')
                    board[i][j] = 'X';
    }

```

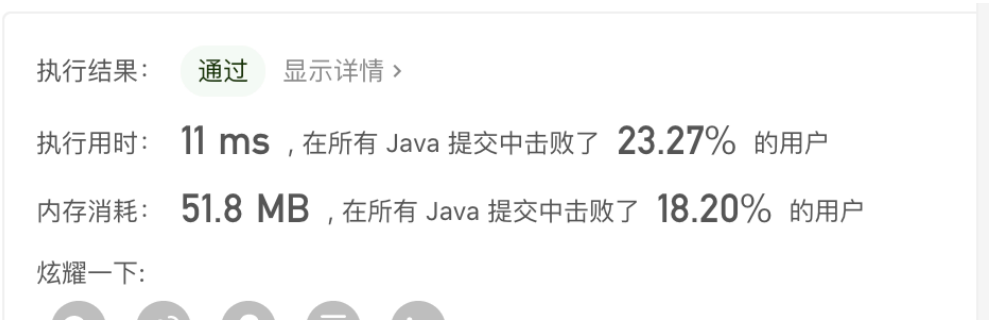
131 Palindrome Patitioning

```
    List<List<String>> res;
    public List<List<String>> partition(String s) {
        res = new ArrayList<>();

        backtrack(s, 0, new ArrayList<>());
        return res;
    }

    private void backtrack(String s, int index, List<String> path) {
        if(index == s.length()){
            res.add(new ArrayList<>(path));
            return;
        }

        for(int i = index + 1; i <= s.length(); i++){
            String frac = s.substring(index, i);

            if(isPalindrome(frac)){
                path.add(frac);
                backtrack(s, i, path);
                path.remove(path.size() - 1);
            }
        }
    }

    private boolean isPalindrome(String s){
        int left = 0, right = s.length() - 1;
        while(left < right){
```

```
            if(s.charAt(left) != s.charAt(right))
                return false;
            left++;
            right--;
        }

        return true;
    }
```

```
//二刷，需要记忆化优化
List<List<String>> res = new ArrayList<>();
    public List<List<String>> partition(String s) {
        backtrack(s.toCharArray(), new ArrayList<>(), 0);

        return res;
    }

    private void backtrack(char[] chars, List<String> path, int index) {
        if(index == chars.length){
            res.add(new ArrayList<>(path));
            return;
        }

        for(int i = index; i < chars.length; i++){
            if(isPalindrome(chars, index, i)){
                String str = toStr(chars, index, i);

                path.add(str);
                backtrack(chars, path, i + 1);
                path.remove(path.size() - 1);
            }
        }
    }

    private String toStr(char[] chars, int left, int right){
        StringBuilder res = new StringBuilder();
        for(int i = left; i <= right; i++)
            res.append(chars[i]);

        return res.toString();
    }

    private boolean isPalindrome(char[] chars, int left, int right){
        while(left < right){
            if(chars[left] != chars[right])
```

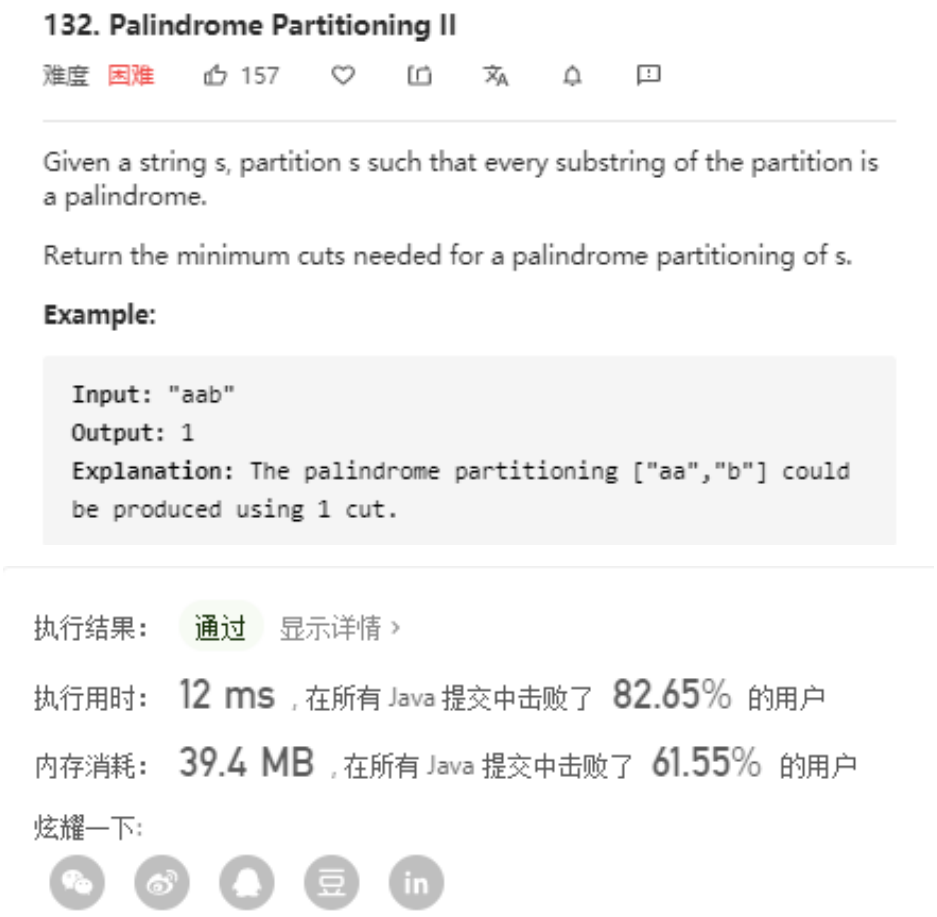
```
                return false;

            left++;
            right--;
        }

        return true;
    }
```




```
class Solution {
    List<List<String>> res = new ArrayList<>();
    public List<List<String>> partition(String s) {

        backtrack(s, new ArrayList<>());
        return res;
    }

    public void backtrack(String s, List<String> path){
        if(s.equals("")){
            res.add(new ArrayList<>(path));
            return;
        }

        for(int i = 0; i < s.length(); i++){
            String frac = s.substring(0, i + 1);

            if(isPalindrome(frac)){
                path.add(frac);

                backtrack(s.substring(i + 1), path);

                path.remove(path.size() - 1);
            }
        }

    }

```

```java
    public boolean isPalindrome(String s){
        int left = 0, right = s.length() - 1;
        while(left < right)
            if(s.charAt(left++) != s.charAt(right--))
                return false;

        return true;
    }
}
```

132 Palindrome Partitioning II



132. Palindrome Partitioning II

难度 困难 157

Given a string s, partition s such that every substring of the partition is a palindrome.

Return the minimum cuts needed for a palindrome partitioning of s.

Example:

```
Input: "aab"
Output: 1
Explanation: The palindrome partitioning ["aa","b"] could be produced using 1 cut.
```

执行结果： 通过 显示详情 >

执行用时： 12 ms , 在所有 Java 提交中击败了 82.65% 的用户

内存消耗： 39.4 MB , 在所有 Java 提交中击败了 61.55% 的用户

炫耀一下:

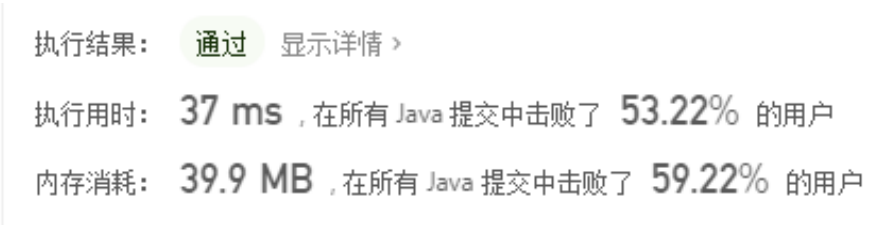
```java
/*
  动态规划
  dp[i]：表示前缀子串 s[0:i] 分割成若干个回文子串所需要最小分割次数。
  同时预处理s， 利用Leetcode 第五题， 从而快速判断
*/

    public int minCut(String s) {
        int len = s.length();
        if(len < 2)     return 0;

        int[] dp = new int[len];
        for(int i = 0; i < len; i++)
            dp[i] = i;
        boolean[][] test = longestPalinrome(s.toCharArray());

        for(int i = 1; i < len; i++)
        {
            if(test[0][i])
            {
                dp[i] = 0;
                continue;
            }

            for(int j = 0; j < i; j++)
            {
                if(test[j + 1][i])
                    dp[i] = Math.min(dp[i], dp[j] + 1);
            }
        }
         return dp[len - 1];
    }

    private boolean[][] longestPalinrome(char[] chars)
    {
        int len = chars.length;
        //dp[i][j] from s[i...j] is Palindrome
        boolean[][] dp = new boolean[len][len];
        for(int i = 0; i < len; i++)
            dp[i][i] = true;

        for(int j = 0; j < len; j++)
            for(int i = 0; i < j; i++)
            {
                if(chars[i] == chars[j])
                    dp[i][j] = (j - i <= 2 || dp[i + 1][j - 1]);
                else
                    dp[i][j] = false;
            }

```

```
        return dp;
    }
https://leetcode-cn.com/problems/palindrome-partitioning-ii/solution/dong-tai-gui-hua-by-liweiwei1419-2/
```

```
/*
  这个方法不知道为什么，只能通过27/29个案例

  想来想去，应该是算法问题
  因为我的算法基于以下假设：
    拿到该字符串中最长的回文串， 然后在回文串的两边做切割，进而再一步判断
  但事实上可能并非切割最长的回文串，才能拿到最大，因此该算法需要再次考虑
*/
    public int minCut(String s) {
        return minCut(s.toCharArray());
    }

    private int minCut(char[] chars)
    {
        if(chars.length <= 1)       return 0;
        int[] arr = longestPalinrome(chars);
        if(arr[0] == arr[1])        return chars.length - 1;

        if(arr[0] == 0 && arr[1] == chars.length - 1)   return 0;
        else if(arr[0] == 0)                            return 1 +
minCut(Arrays.copyOfRange(chars, arr[1] + 1, chars.length));
        else if(arr[1] == chars.length - 1)             return 1 +
minCut(Arrays.copyOfRange(chars, 0, arr[0]));
        else
            return 2 +minCut(Arrays.copyOfRange(chars, arr[1] + 1, chars.length)) +
                    minCut(Arrays.copyOfRange(chars, 0, arr[0]));
    }

    //this function returns the upperbound and lowerbound of the palindrome
    private int[] longestPalinrome(char[] chars)
    {
        int len = chars.length;
        //dp[i][j] from s[i...j] is Palindrome
        boolean[][] dp = new boolean[len][len];
        for(int i = 0; i < len; i++)
            dp[i][i] = true;

        for(int j = 0; j < len; j++)
            for(int i = 0; i < j; i++)
            {
                if(chars[i] == chars[j])
                    dp[i][j] = (j - i <= 2 || dp[i + 1][j - 1]);
```

```
                else
                    dp[i][j] = false;
            }

        int[] res = new int[2];
        int maxLen = 0;
        for(int i = 0; i < len ; i++)
            for(int j = i; j < len; j++)
            {
                if(dp[i][j])
                    if(j - i + 1 > maxLen)
                    {
                        maxLen = j - i + 1;
                        res[0] = i;
                        res[1] = j;
                    }
            }
        return res;
    }
```

133 Clone Graph




```
//典型BFS, 很类似链表操作
class Solution {
    public Node cloneGraph(Node node) {
        if(node == null)    return null;
        HashMap<Node, Node> map = new HashMap<>();
        Deque<Node> queue       = new ArrayDeque<>();
        HashSet<Node> visited   = new HashSet<>();
        map.put(node, new Node(node.val));
        queue.add(node);

        while(!queue.isEmpty()){
            Node cur = queue.removeFirst();
            if(visited.contains(cur)) continue;

            for(Node n : cur.neighbors){
                map.putIfAbsent(n, new Node(n.val));

                map.get(cur).neighbors.add(map.get(n));
```

```
                queue.add(n);
            }

            visited.add(cur);
        }


        return map.get(node);
    }
}
```

```
class Solution {
    public Node cloneGraph(Node node) {
        Map<Node, Node> lookup = new HashMap<>();
        return dfs(node, lookup);
    }
  //递归，  坚定不移的相信，这个函数将返回对应点的克隆点
    private Node dfs(Node node, Map<Node,Node> lookup) {
        if (node == null) return null;
        if (lookup.containsKey(node)) return lookup.get(node);
        Node clone = new Node(node.val, new ArrayList<>());
        lookup.put(node, clone);
        for (Node n : node.neighbors)clone.neighbors.add(dfs(n,lookup));
        return clone;
    }
}

作者：powcai
链接：https://leetcode-cn.com/problems/clone-graph/solution/dfs-he-bfs-by-powcai/
```

```
/*
    Cpp 版本
*/
class Solution {
public:
    Node* cloneGraph(Node* node) {
        if (node == nullptr) {
            return node;
        }

        unordered_map<Node*, Node*> visited;

        // 将题目给定的节点添加到队列
        queue<Node*> Q;
```

```
        Q.push(node);
        // 克隆第一个节点并存储到哈希表中
        visited[node] = new Node(node->val);

        // 广度优先搜索
        while (!Q.empty()) {
            // 取出队列的头节点
            auto n = Q.front();
            Q.pop();
            // 遍历该节点的邻居
            for (auto& neighbor: n->neighbors) {
                if (visited.find(neighbor) == visited.end()) {
                    // 如果没有被访问过，就克隆并存储在哈希表中
                    visited[neighbor] = new Node(neighbor->val);
                    // 将邻居节点加入队列中
                    Q.push(neighbor);
                }
                // 更新当前节点的邻居列表
                visited[n]->neighbors.emplace_back(visited[neighbor]);
            }
        }

        return visited[node];
    }
};

作者：LeetCode-Solution
链接：https://leetcode-cn.com/problems/clone-graph/solution/ke-long-tu-by-leetcode-solution/
```

134 Gas Station

134. Gas Station

难度 中等 336

There are N gas stations along a circular route, where the amount of gas at station i is `gas[i]`.

You have a car with an unlimited gas tank and it costs `cost[i]` of gas to travel from station i to its next station (i+1). You begin the journey with an empty tank at one of the gas stations.

Return the starting gas station's index if you can travel around the circuit once in the clockwise direction, otherwise return -1.

Note:

- If there exists a solution, it is guaranteed to be unique.
- Both input arrays are non-empty and have the same length.
- Each element in the input arrays is a non-negative integer.

Example 1:

```
Input:
gas  = [1,2,3,4,5]
cost = [3,4,5,1,2]

Output: 3

Explanation:
Start at station 3 (index 3) and fill up with 4 unit of
gas. Your tank = 0 + 4 = 4
Travel to station 4. Your tank = 4 - 1 + 5 = 8
```

执行用时： 0 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 38.7 MB , 在所有 Java 提交中击败了 52.41% 的用户

炫耀一下:

```java
public int canCompleteCircuit(int[] gas, int[] cost) {
    int len = gas.length;

    for(int i = 0; i < len; i++){
        int index = i;
        int oilSum = 0;
        while(gas[index] - cost[index] + oilSum >= 0){
            oilSum += gas[index] - cost[index];
            index = (index + 1) % len;

            if(index == i)
                return i;
        }

```

```
            if(index < i)
                return -1;
            i = index;
        }

        return -1;
    }
```

执行结果： 通过 显示详情 ›　　添加备

执行用时： 194 ms ，在所有 Java 提交中击败了 5.00% 的用户

内存消耗： 38.5 MB ，在所有 Java 提交中击败了 88.92% 的用户

```
//n 刷 0507
public int canCompleteCircuit(int[] gas, int[] cost) {

        for(int i = 0; i < gas.length; i++){
            if(gas[i] < cost[i])
                continue;

            int index = i;
            int oilSum = 0;

            while(true){
                oilSum += gas[index] -  cost[index];
                int step = gas[index];

                index++;
                index %= gas.length;

                if(index == i){
                    //System.out.println(i);
                    return i;
                }else if(cost[index] - gas[index] > oilSum)
                    break;
            }
        }

        return -1;
    }
```

```
public int canCompleteCircuit(int[] gas, int[] cost) {
```

```
    int volumn = 0;
    for(int i = 0; i < gas.length; i++)
    {
        volumn = gas[i] - cost[i];
        if(volumn < 0)      continue;
        int j = (i + 1) % gas.length;
        for(; j != i; j = (j + 1) % gas.length)
        {
            volumn += gas[j] - cost[j];
            if(volumn < 0)      break;
        }

        if(j == i)      return j;
    }
    return -1;
}
```

135 Candy

135. Candy

难度 困难 231

There are N children standing in a line. Each child is assigned a rating value.

You are giving candies to these children subjected to the following requirements:

- Each child must have at least one candy.
- Children with a higher rating get more candies than their neighbors.

What is the minimum candies you must give?

Example 1:

```
Input: [1,0,2]
Output: 5
Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.
```

Example 2:

```
Input: [1,2,2]
Output: 4
Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively.
             The third child gets 1 candy because it satisfies the above two conditions.
```

通过次数 24,108 | 提交次数 54,816

执行结果： 通过 显示详情 >

执行用时： 64 ms , 在所有 Java 提交中击败了 10.47% 的用户

内存消耗： 48 MB , 在所有 Java 提交中击败了 5.06% 的用户

炫耀一下：

```java

class Solution {
    public int candy(int[] ratings) {
        int len = ratings.length;
        TreeMap<Integer, List<Integer>> map = new TreeMap<>();

```

```
7
8           //for 不同索引,  分配的糖果
9           HashMap<Integer, Integer> record    = new HashMap<>();
10          for(int i = 0; i < len; i++){
11              if(!map.containsKey(ratings[i]))
12                  map.put(ratings[i], new ArrayList<>());
13
14              map.get(ratings[i]).add(i);
15          }
16
17          boolean first = true;
18          int count = 0;
19          for(Map.Entry<Integer, List<Integer>> entry : map.entrySet()){
20              if(first){
21                  for(Integer index : entry.getValue()){
22                      record.put(index, 1);
23                      count += 1;
24                  }
25                  first = !first;
26              }else{
27                  for(Integer index : entry.getValue()){
28                      int num = 1;
29                      if(index > 0 && ratings[index - 1] < ratings[index])
30                          num = Math.max(num, record.getOrDefault(index - 1, 0) + 1);
31                      if(index < len - 1 && ratings[index + 1] < ratings[index])
32                          num = Math.max(num ,record.getOrDefault(index + 1, 0) + 1);
33
34                      record.put(index, num);
35                      count += num;
36                  }
37              }
38          }
39
40          return count;
41      }
42
43  }
44
```

```
1   /*
2   Req:
3     Each child must have at least one candy.
4     Children with a higher rating get more candies than their neighbors.
5   从左遍历和从右遍历
```

```
取两次遍历结果的最大值，这样可以保证都符合规则

*/
class Solution {
    public int candy(int[] ratings) {
        int[] left = new int[ratings.length];
        int[] right = new int[ratings.length];
        Arrays.fill(left, 1);
        Arrays.fill(right, 1);
        //和左边比
        for(int i = 1; i < ratings.length; i++)
            if(ratings[i] > ratings[i - 1])
                left[i] = left[i - 1] + 1;

        int count = left[ratings.length - 1];

        //和右边比
        for(int i = ratings.length - 2; i >= 0; i--)
        {
            if(ratings[i] > ratings[i + 1])
                right[i] = right[i + 1] + 1;
            count += Math.max(left[i], right[i]);
        }
        return count;
    }
}

作者：jyd
链接：https://leetcode-cn.com/problems/candy/solution/candy-cong-zuo-zhi-you-cong-you-zhi-zuo-qu-zui-da-/
```

136 Single Number map方法的掌握以及异或的玩法

136. Single Number

难度 简单 1362

Given a **non-empty** array of integers, every element appears twice except for one. Find that single one.

Note:

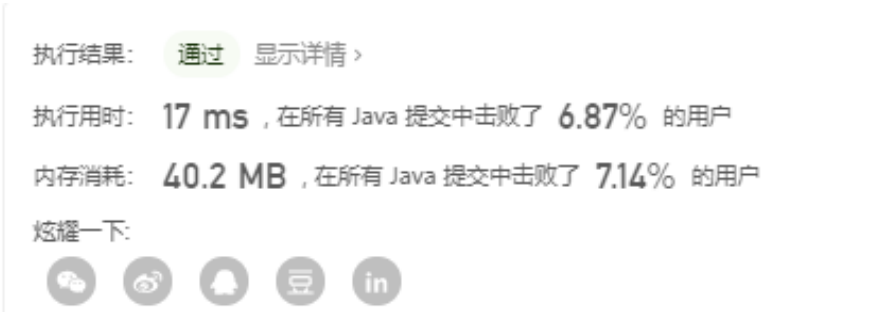
Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

Example 1:

```
Input: [2,2,1]
Output: 1
```

Example 2:

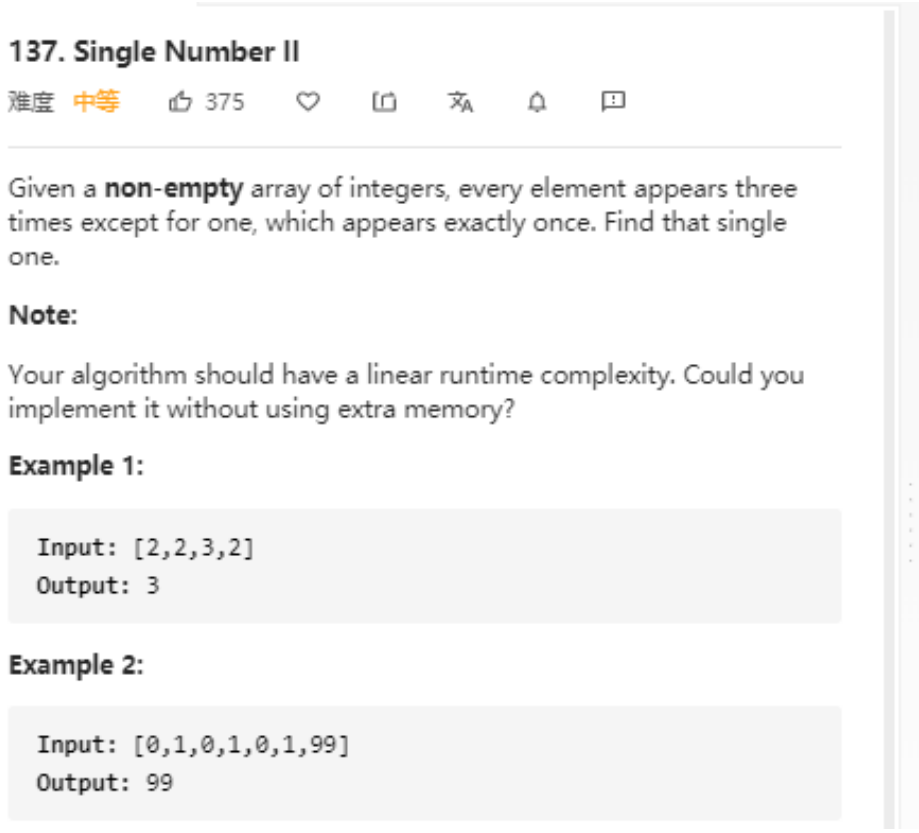
```
Input: [4,1,2,1,2]
Output: 4
```




```java
/*
  注意如何拿出map的元素方法
*/
public int singleNumber(int[] nums) {
    int len = nums.length;
    HashMap<Integer, Integer> map = new HashMap<>();

    for(int i = 0; i < len; i++)
    {
        map.put(nums[i], map.getOrDefault(nums[i],0) + 1);
        if(map.get(nums[i]) == 2)
            map.remove(nums[i]);
    }

    for(Map.Entry<Integer, Integer> entry : map.entrySet())
        return entry.getKey();
    return 0;
}
```

```
/*
牛逼解法 异或
a^b^a=a^a^b=b,因此ans相当于nums[0]^nums[1]^nums[2]^nums[3]^nums[4].
然后再根据交换律把相等的合并到一块儿进行异或（结果为0），然后再与只出现过一次的元素进行异或，这样最后的结果就是，只出现过一次的元素（0^任意值=任意值）
*/
int ans = nums[0];
if (nums.length > 1) {
   for (int i = 1; i < nums.length; i++) {
      ans = ans ^ nums[i];
   }
 }
 return ans;

作者：yinyinnie
链接：https://leetcode-cn.com/problems/single-number/solution/xue-suan-fa-jie-guo-xiang-dui-yu-guo-cheng-bu-na-y/
```

137 Single Number II



137. Single Number II

难度 中等 375

Given a **non-empty** array of integers, every element appears three times except for one, which appears exactly once. Find that single one.

Note:

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

Example 1:

```
Input: [2,2,3,2]
Output: 3
```

Example 2:

```
Input: [0,1,0,1,0,1,99]
Output: 99
```


```java
public int singleNumber(int[] nums) {
    int len = nums.length;
    HashMap<Integer, Integer> map = new HashMap<>();

    for(int i = 0; i < len; i++)
    {
        map.put(nums[i], map.getOrDefault(nums[i],0) + 1);
        if(map.get(nums[i]) == 3) //这里是与上一个题唯一不同的地方
            map.remove(nums[i]);
    }

    for(Map.Entry<Integer, Integer> entry : map.entrySet())
        return entry.getKey();
    return 0;
}
```

138 Copy List With Random Pointer 哈希map的灵活应用

138. Copy List with Random Pointer

难度 中等 312

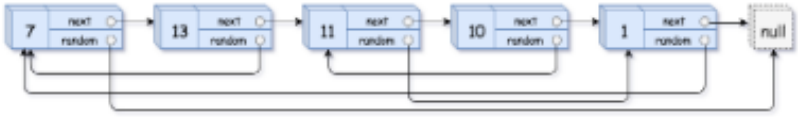
A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a deep copy of the list.

The Linked List is represented in the input/output as a list of `n` nodes. Each node is represented as a pair of `[val, random_index]` where:

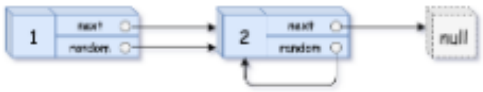
- `val`: an integer representing `Node.val`
- `random_index`: the index of the node (range from `0` to `n-1`) where random pointer points to, or `null` if it does not point to any node.

Example 1:




```
Input: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]
Output: [[7,null],[13,0],[11,4],[10,2],[1,0]]
```

Example 2:




```
Input: head = [[1,1],[2,1]]
Output: [[1,1],[2,1]]
```

Example 3:


```
Input: head = [[3,null],[3,0],[3,null]]
Output: [[3,null],[3,0],[3,null]]
```

Example 4:

```
Input: head = []
Output: []
Explanation: Given linked list is empty (null pointer),
so return null.
```

Constraints:

```java
/*
  使用HashMap 进行复制
*/
public Node copyRandomList(Node head) {
    if (head == null) return null;
    HashMap<Node, Node> map = new HashMap<>();
    Node p = head;

    while (p != null) {
        map.put(p, new Node(p.val));
        p = p.next;
    }

```

```
    p = head;

    while (p != null) {
        map.get(p).next = map.get(p.next);
        map.get(p).random = map.get(p.random);
        p = p.next;
    }

    return map.get(head);
}

作者：jerrymouse1998
链接：https://leetcode-cn.com/problems/copy-list-with-random-pointer/solution/javatu-jie-cong-hashmapdao-chang-shu-kong-jian-by-/
```

140 Word Break II

140. Word Break II

难度 困难 193

Given a **non-empty** string s and a dictionary wordDict containing a list of **non-empty** words, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences.

Note:

- The same word in the dictionary may be reused multiple times in the segmentation.
- You may assume the dictionary does not contain duplicate words.

Example 1:

```
Input:
s = "catsanddog"
wordDict = ["cat", "cats", "and", "sand", "dog"]
Output:
[
  "cats and dog",
  "cat sand dog"
]
```

Example 2:

```
Input:
s = "pineapplepenapple"
wordDict = ["apple", "pen", "applepen", "pine",
"pineapple"]
Output:
[
  "pine apple pen apple",
  "pineapple pen apple",
  "pine applepen apple"
]
Explanation: Note that you are allowed to reuse a
dictionary word.
```

Example 3:

```
Input:
s = "catsandog"
wordDict = ["cats", "dog", "sand", "and", "cat"]
Output:
[]
```

通过次数 17,451 | 提交次数 45,692

在真实的面试中遇到过这道题?

```java
/*
  以下代码可以通过 31/36 个测试案例
  但是时间太高了，需要优化
  本质是dp， dfs， 我把他放入dp-part中

  下面附一个记忆化回溯
*/
class Solution {
    public List<String> res;
    public char[] charArray;
    public List<String> wordBreak(String s, List<String> wordDict) {
        Set<String> dict = new HashSet<>(wordDict);
        res = new ArrayList<>();
        StringBuilder sb = new StringBuilder();
        charArray = s.toCharArray();

        backtrack(0, charArray, dict, sb);
        return res;
    }

    private void backtrack(int start, char[] charArray, Set<String> dict,
StringBuilder sb) {
        if(start == charArray.length)
        {
            StringBuilder fruit = new StringBuilder(sb);
            fruit.setLength(fruit.length()-1);
            res.add(fruit.toString());
            return;
        }

        for(int i = 0; i < charArray.length; i++)
        {
            if(start + i+1 <= charArray.length)
            {
                StringBuilder frac = new StringBuilder();
                frac.append(charArray, start, i+1); //  就算这样优化，回溯时间也还是很多

                if(dict.contains(frac.toString()))
                {
                    sb.append(frac).append(" ");
                    backtrack(start + i+1, charArray, dict, sb);
                    sb.setLength(sb.length() - frac.length() - 1);
                }
            }
        }
    }
}
```

```java
/*
    更好理解的一种方法，  就是记忆化回溯

    将map 作为记忆 数据结构，
    workBreakHelper 本质上是递归
*/
class Solution {
    private HashMap<String, List<String>> map = new HashMap<>();
    public List<String> wordBreak(String s, List<String> wordDict) {
        HashSet<String> set = new HashSet<>();
        for(String str : wordDict)
            set.add(str);

        wordBreakHelper(s, set);
        return map.get(s);
    }

    private List<String> wordBreakHelper(String s, HashSet<String> set)
    {
        if(map.containsKey(s))
            return map.get(s);
        if(s.length() == 0)         return new ArrayList<>();
        List<String> res = new ArrayList<>();

        for(int j = 1; j <= s.length(); j++)
        {
            String frac = s.substring(0, j);

            if(set.contains(frac))
            {
                if(j == s.length())
                    res.add(frac);
                else
                {
                    List<String> temp = wordBreakHelper(s.substring(j), set);
                    for(String ss : temp)
                        res.add(frac + " " + ss);
                }
            }
        }
        map.put(s, res);
        return res;
    }

    public static void main(String[] args){
        String s = "catsanddog";
```

```
47          List<String> wordDict = new ArrayList<>(Arrays.asList("cat", "cats", "and",
    "sand", "dog"));
48          Solution ss = new Solution();
49
50          ss.wordBreak(s, wordDict);
51      }
52  }
```

141 Linked List Cycle

141. Linked List Cycle

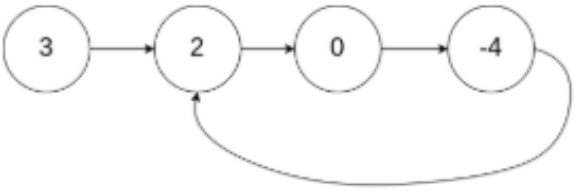
难度 简单 670

Given a linked list, determine if it has a cycle in it.

To represent a cycle in the given linked list, we use an integer `pos` which represents the position (0-indexed) in the linked list where tail connects to. If `pos` is `-1`, then there is no cycle in the linked list.

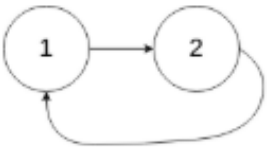
Example 1:

```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where
tail connects to the second node.
```



Example 2:

```
Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where
tail connects to the first node.
```



Example 3:

```
Input: head = [1], pos = -1
Output: false
Explanation: There is no cycle in the linked list.
```


```java
public boolean hasCycle(ListNode head) {
    if(head == null)    return false;

    ListNode fast = head;
    ListNode slow = head;

    while(fast != null && fast.next != null)
    {
        fast = fast.next.next;
        slow = slow.next;
        if(fast == slow)
            return true;
    }
    return false;
}
```

142 Linked List Cycle II

142. Linked List Cycle II

难度 中等 531

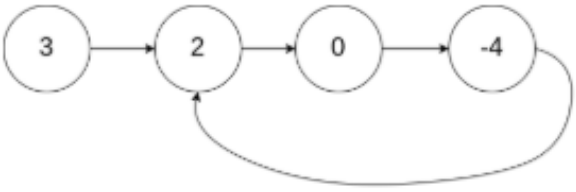
Given a linked list, return the node where the cycle begins. If there is no cycle, return `null`.

To represent a cycle in the given linked list, we use an integer `pos` which represents the position (0-indexed) in the linked list where tail connects to. If `pos` is `-1`, then there is no cycle in the linked list.

Note: Do not modify the linked list.

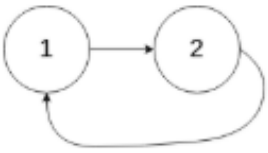
Example 1:

```
Input: head = [3,2,0,-4], pos = 1
Output: tail connects to node index 1
Explanation: There is a cycle in the linked list, where
tail connects to the second node.
```



Example 2:

```
Input: head = [1,2], pos = 0
Output: tail connects to node index 0
Explanation: There is a cycle in the linked list, where
tail connects to the first node.
```



Example 3:

```
Input: head = [1], pos = -1
Output: no cycle
Explanation: There is no cycle in the linked list.
```

Follow-up:
Can you solve it without using extra space?

通过次数 92,336 | 提交次数 181,507


```java
public ListNode detectCycle(ListNode head) {
    if(head == null)        return null;
    if(head.next == null)    return null;
    if(head.next.next == head)     return head;

    ListNode slow = head;
    ListNode fast = head;
    while(fast != null && fast.next != null)
    {
        fast = fast.next.next;
        slow = slow.next;
        if(fast == slow)
            break;
    }
    if(fast != slow)        return null;

    slow = head;
    while(slow != fast)
    {
        slow = slow.next;
        fast = fast.next;
    }

    return fast;
}
```

143 Reorder List

143. Reorder List

难度 中等 247

Given a singly linked list L: $L_0 \rightarrow L_1 \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n$,
reorder it to: $L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \dots$

You may **not** modify the values in the list's nodes, only nodes itself may be changed.

Example 1:

```
Given 1->2->3->4, reorder it to 1->4->2->3.
```

Example 2:

```
Given 1->2->3->4->5, reorder it to 1->5->2->4->3.
```

执行结果: 通过 显示详情 >

执行用时: 4 ms , 在所有 Java 提交中击败了 31.36% 的用户

内存消耗: 42.1 MB , 在所有 Java 提交中击败了 9.09% 的用户

炫耀一下:

```java
public void reorderList(ListNode head) {
    if(head == null)        return;

    Deque<ListNode> stack = new ArrayDeque<>();
    ListNode cur = head;
    while(cur != null)
    {
        stack.push(cur);
        cur = cur.next;
    }

    cur = head;
    ListNode stack_cur = new ListNode(Integer.MAX_VALUE);

    while(cur.next != stack_cur.next)
    {
        stack_cur = stack.poll();
        stack_cur.next = cur.next;
        cur.next = stack_cur;

        cur = cur.next.next;
```

```
    }

    stack_cur.next = null;    //to avoid the cycle
}
```


```java
public void reorderList(ListNode head) {
    Deque<ListNode> queue = new ArrayDeque<>();
    if(head == null)        return;

    ListNode cur = head;
    while(cur != null)
    {
        queue.addLast(cur);
        cur = cur.next;
    }

    ListNode dummy = new ListNode(0);
    cur = dummy;
    ListNode left = null, right = null;
    while(queue.size() >= 2)
    {
        left = queue.removeFirst();
        right = queue.removeLast();

        cur.next = left;
        cur.next.next = right;
        cur = cur.next.next;
        cur.next = null;
    }

    if(!queue.isEmpty()){
        cur.next = queue.removeFirst();
        cur.next.next = null;
    }


    head = dummy.next;
```

```
}
```

145 Binary Tree Postorder Traversal

```go
func postorderTraversal(root *TreeNode) []int {
  res := make([]int, 0)
  stack := make([]*TreeNode, 0)

  if root == nil{
    return res
  }

  for ; len(stack) != 0 || root != nil;{
    for;root != nil;{
      stack = append(stack, root)
      res = append(res, root.Val)
      root = root.Right
    }

    root = stack[len(stack) - 1]
    stack = stack[0 : len(stack) - 1]
        root = root.Left
  }

  for i, j := 0, len(res) - 1 ; i < j; {
    res[i], res[j] = res[j], res[i]
    i++
    j--
  }
  return res
}

```

146 LRU Cache ☆☆☆

146. LRU Cache

难度 中等 739

Design and implement a data structure for Least Recently Used (LRU) cache. It should support the following operations: `get` and `put`.

`get(key)` - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.
`put(key, value)` - Set or insert the value if the key is not already present. When the cache reached its capacity, it should invalidate the least recently used item before inserting a new item.

The cache is initialized with a **positive** capacity.

Follow up:
Could you do both operations in **O(1)** time complexity?

Example:

```
LRUCache cache = new LRUCache( 2 /* capacity */ );

cache.put(1, 1);
cache.put(2, 2);
cache.get(1);       // returns 1
cache.put(3, 3);    // evicts key 2
cache.get(2);       // returns -1 (not found)
cache.put(4, 4);    // evicts key 1
cache.get(1);       // returns -1 (not found)
cache.get(3);       // returns 3
cache.get(4);       // returns 4
```

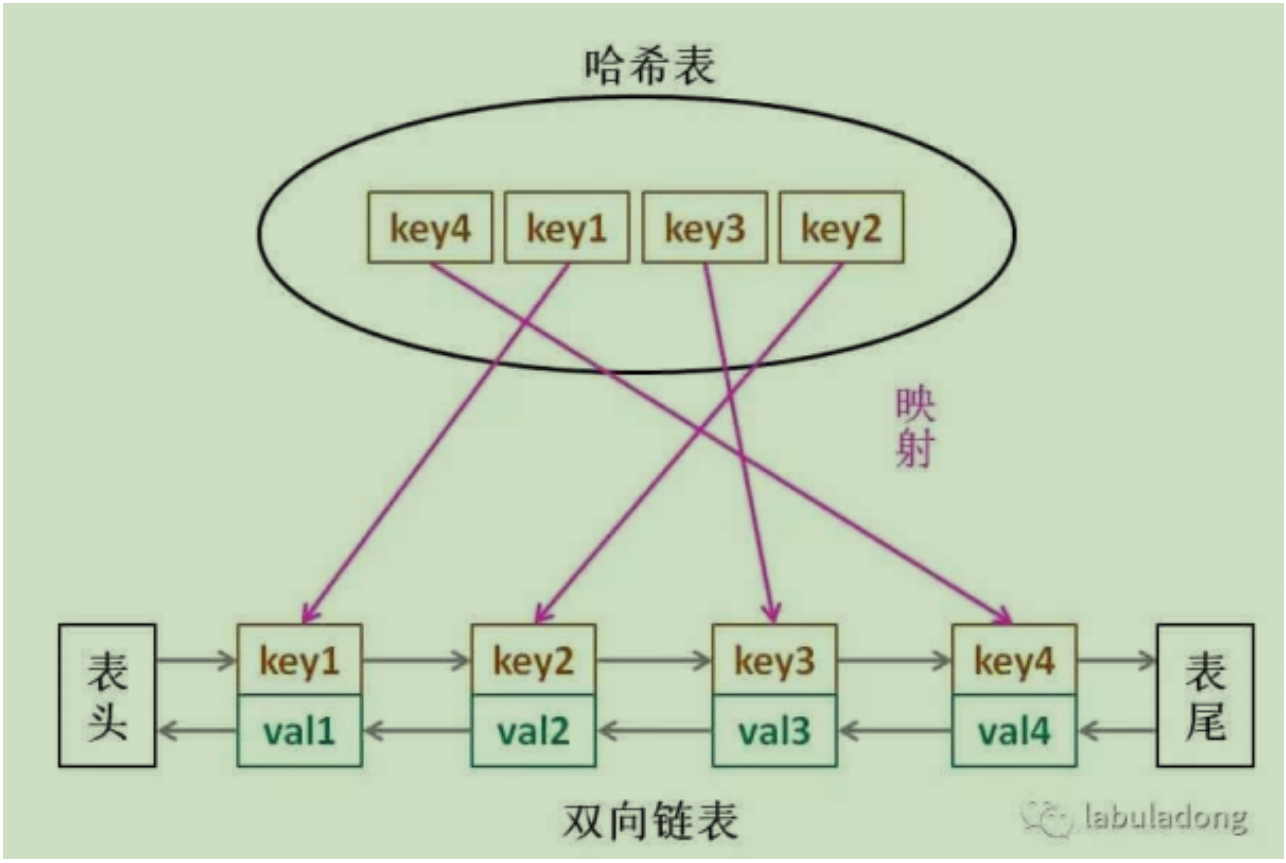
内存干净的版本， 用 valgrind 验证内存泄漏

```cpp
#include <iostream>
#include <deque>
#include <thread>
#include <unordered_map>

using namespace std;

class Node{
public:
    int key;
    int val;
    Node* prev;
    Node* next;
    Node(int _key, int _val) : key(_key), val(_val), prev(nullptr), next(nullptr)
{};
};
```

```cpp

class DoublyLinkedList{
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() : head(new Node(0, 0)), tail(new Node(0, 0)){
        head->next = tail;
        tail->prev = head;
    }

    void addFirst(Node* node){
        node->next = head->next;
        node->prev = head->prev;

        head->next->prev = node;
        head->next = node;
    }

    Node* deleteLast(){
        if(head->next != tail){
            return deleteNode(tail->prev);
        }

        return nullptr;
    }

    Node* deleteNode(Node* node){
        node->next->prev = node->prev;
        node->prev->next = node->next;

        return node;
    }

};

class LRUCache {
public:
    int cap;
    DoublyLinkedList dll;
    unordered_map<int, Node*> map;
    int count = 0;
    LRUCache(int capacity) {
        cap = capacity;
        count = 0;
    }

    int get(int key) {
        if(map.count(key) == 0)
```

```
            return -1;

        int val = map[key]->val;
        Node* node = dll.deleteNode(map[key]);
        dll.addFirst(node);

        return val;
    }

    void put(int key, int value) {
        if(map.count(key) != 0){
            map[key]->val = value;
            get(key);
        }else{
            if(count == cap){
                Node* delNode = dll.deleteLast();
                map.erase(delNode->key);

                delete delNode;
            }

            Node* newNode = new Node(key, value); //注意这里一定用 new 因为 如果采用栈新建对象，当函数结束，该对象会被回收
            map[key] = newNode;

            dll.addFirst(newNode);

            count++;
        }

    }



    ~LRUCache(){
        Node* cur = dll.head->next;
        while(cur != dll.tail){
            cur = cur->next;
            delete(cur->prev);
        }

  delete(dll.head);
  delete(dll.tail);
    }
};


int  main(){
    LRUCache lru(2);
```

```
    lru.put(1,1);
    lru.put(2,2);
    lru.get(1);
    lru.put(3,3);
    lru.get(2);
    lru.put(4,4);
    lru.get(1);
    lru.get(3);
    lru.get(4);

    return 0;
}
```




```
/*
  适合查找快，插入快，删除快，有顺序之分的数据结构
  哈希链表 => 双链表和哈希表的结合
  借助哈希表赋予链表快速查找的特性

  处理链表节点的时候，不要忘记同时更新哈希表中对节点的映射
*/

class LRUCache
{
    private HashMap<Integer, Node> map;

```

```
    private DoubleList cache;

    private int cap;

    public LRUCache(int capacity)
    {
        this.cap = capacity;
        map = new HashMap<>();
        cache = new DoubleList();
    }

    public int get(int key)
    {
        if(!map.containsKey(key))
            return -1;
        int val = map.get(key).val;
        put(key, val);
        return val;
    }

    public void put(int key, int val)
    {
        Node x = new Node(key, val);

        if(map.containsKey(key))
        {
            cache.remove(map.get(key));
            cache.addFirst(x);
            map.put(key, x);
        }
        else
        {
            if(cap == cache.size())
            {
                Node last = cache.removeLast();
                map.remove(last.key);
            }
            cache.addFirst(x);
            map.put(key, x);
        }
    }
}

class Node
{
    public int key, val;
    public Node next, prev;
    public Node(int k, int v)
    {
```

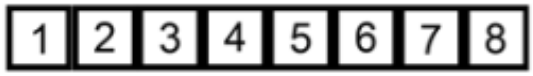
```
        this.key = k;
        this.val = v;
    }
}

class DoubleList {
    private Node head, tail; // 头尾虚节点
    private int size; // 链表元素数

    public DoubleList() {
        head = new Node(0, 0);
        tail = new Node(0, 0);
        head.next = tail;
        tail.prev = head;
        size = 0;
    }

    // 在链表头部添加节点 x
    public void addFirst(Node x) {
        x.next = head.next;
        x.prev = head;
        head.next.prev = x;
        head.next = x;
        size++;
    }

    // 删除链表中的 x 节点（x 一定存在）
    public void remove(Node x) {
        x.prev.next = x.next;
        x.next.prev = x.prev;
        size--;
    }

    // 删除链表中最后一个节点，并返回该节点
    public Node removeLast() {
        if (tail.prev == head)
            return null;
        Node last = tail.prev;
        remove(last);
        return last;
    }

    // 返回链表长度
    public int size() { return size; }
}
https://leetcode-cn.com/problems/lru-cache/solution/lru-ce-lue-xiang-jie-he-shi-xian-by-labuladong/
```

147 Insertion Sort List

147. Insertion Sort List

难度 中等 188

Sort a linked list using insertion sort.



A graphical example of insertion sort. The partial sorted list (black) initially c
only the first element in the list.
With each iteration one element (red) is removed from the input data and i
place into the sorted list

Algorithm of Insertion Sort:

1. Insertion sort iterates, consuming one input element e
   repetition, and growing a sorted output list.
2. At each iteration, insertion sort removes one element
   input data, finds the location it belongs within the sor
   and inserts it there.
3. It repeats until no input elements remain.

执行结果: 通过 显示详情 >

执行用时: 28 ms , 在所有 Java 提交中击败了 27.63% 的用户

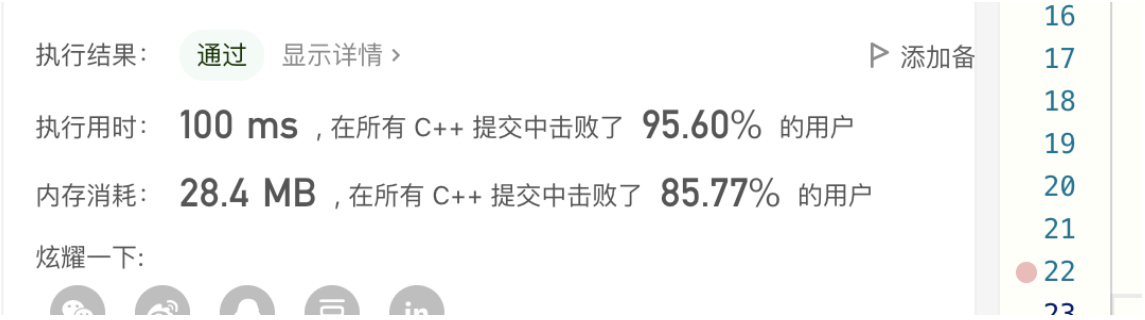
内存消耗: 39.6 MB , 在所有 Java 提交中击败了 6.67% 的用户

炫耀一下:

```java
public ListNode insertionSortList(ListNode head) {
    if(head == null)        return null;

    ListNode newHead = new ListNode(0);

    ListNode cur = head;
    ListNode curNew = newHead;

    while(cur != null)
    {
```

```
        ListNode newNode = new ListNode(cur.val);
        while(curNew.next != null && curNew.next.val < newNode.val)
            curNew = curNew.next;

        newNode.next = curNew.next;
        curNew.next = newNode;

        cur = cur.next;
        curNew = newHead;
    }
    return newHead.next;
}
```

148 Sort List




```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if(head == NULL || head->next == NULL)
            return head;

        ListNode* fast = head;
        ListNode* slow = head;

        ListNode* brk = NULL;
        while(fast != NULL && fast->next != NULL){
            fast = fast->next->next;

            if(fast == NULL || fast->next == NULL)
                brk = slow;
            slow = slow->next;
        }

        brk->next = nullptr;

        ListNode* head1 = sortList(head);
```

```
        ListNode* head2 = sortList(slow);

        ListNode dummy(0);
        ListNode* cur = &dummy;
        while(head1 != NULL || head2 != NULL){
            if(head1 == NULL || (head1 != NULL && head2 != NULL && head1->val >= head2->val)){
                cur->next = head2;
                head2 = head2->next;

                cur = cur->next;
            }else if(head2 == NULL || (head1 != NULL && head2 != NULL && head1->val < head2->val)){
                cur->next = head1;

                head1 = head1->next;
                cur = cur->next;
            }
        }

        return dummy.next;
    }
};

```

执行结果： 通过 显示详情 >

执行用时： 14 ms , 在所有 Java 提交中击败了 19.36% 的用户

内存消耗： 44.9 MB , 在所有 Java 提交中击败了 74.27% 的用户

```
class Solution {
    public ListNode sortList(ListNode head) {
        return sort(head, null);
    }

    private ListNode sort(ListNode start, ListNode end) {
        if(start == end)        return start;

        ListNode fast = start, slow = start;
        while(fast != end && fast.next != end){
            fast = fast.next.next;
            slow = slow.next;
        }
        ListNode l2 = sort(slow.next, end);
        slow.next = null;

```

```
        ListNode l1 = sort(start, slow);


        return merge(l1, l2);
    }


    private ListNode merge(ListNode l1, ListNode l2) {
        if(l1 == null || l2 == null)
            return l1 == null ? l2 : l1;

        if(l1.val < l2.val){
            l1.next = merge(l1.next, l2);
            return l1;
        }else{
            l2.next = merge(l1, l2.next);
            return l2;
        }
    }


}
```


```
public ListNode sortList(ListNode head) {
    return mergeSort(head);
}

private ListNode mergeSort(ListNode head) {
    if (head == null || head.next == null)
        return head;

    ListNode dummy = new ListNode(0);
    dummy.next = head;
    ListNode fast = dummy;
    ListNode slow = dummy;

```

```
    //快慢指针找中点
    while (fast != null && fast.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }

   /*
       [Dummy] -> [0]  -> [1]  -> [2]  -> [3]
                          △slow      △fast
        使用dummy的目的，就是让slow停在中间靠左
        然后用head2指向中间靠右，也就是第二个头
        最后slow.next = null  防止循环链表
   */
    ListNode head2 = slow.next;
    slow.next = null;
    head = mergeSort(head);
    head2 = mergeSort(head2);
    return merge(head, head2);

}

private ListNode merge(ListNode head1, ListNode head2) {
    ListNode dummy = new ListNode(0);
    ListNode tail = dummy;
    while (head1 != null && head2 != null) {
        if (head1.val < head2.val) {
            tail.next = head1;
            tail = tail.next;
            head1 = head1.next;
        } else {
            tail.next = head2;
            tail = tail.next;
            head2 = head2.next;
        }

    }
    if (head1 != null) {
        tail.next = head1;
    }

    if (head2 != null) {
        tail.next = head2;
    }

    return dummy.next;

}
```

149 Max Points On a Line

149. Max Points on a Line

难度 困难 158

Given n points on a 2D plane, find the maximum number of points that lie on the same straight line.

Example 1:

```
Input: [[1,1],[2,2],[3,3]]
Output: 3
Explanation:
^
|
|        o
|     o
|  o
+------------->
0  1  2  3  4
```

Example 2:

```
Input: [[1,1],[3,2],[5,3],[4,1],[2,3],[1,4]]
Output: 4
Explanation:
^
|
|  o
|     o        o
|        o
|  o        o
+------------------->
0  1  2  3  4  5  6
```

NOTE: input types have been changed on April 15, 2019. Please reset to default code definition to get new method signature.

通过次数 13,641 提交次数 60,265

```
/*
  注意本题设置的障碍，存在重复的元素
  先给出原始想法
  20/41
*/
```

```java

public int maxPoints(int[][] points) {
    HashMap<Integer, HashMap<Integer, Integer>> map = new HashMap<>();
    boolean[] marked = new boolean[points.length];
    if(points.length == 0)      return 0;

    //对于每一个点，创建一个hashMap
    //每个点的HashMap 代表， 前面是对应的斜率，后面是个数
    for(int i = 0; i < points.length; i++)
    {
        map.put(i, new HashMap<>());
        for(int j = i+1; j < points.length; j++)
        {
            int k = calculate(points, i, j);
            map.get(i).put(k, map.get(i).getOrDefault(k, 0) + 1);
        }
    }

    int res = 0;
    for(HashMap<Integer, Integer> m : map.values())
        for(Integer i : m.values())
            res = Math.max(res, i);

    return res+1;
}

private int calculate(int[][] points, int i, int j)
{
    int difY = points[j][1] - points[i][1];
    int difX = points[j][0] - points[i][0];

    if(difX == 0)       return Integer.MAX_VALUE;

    return difY / difX;
}

```

```java
/*
  灵感来源于点斜式，确定一个点之后，计算之后的每个点的斜率 即可唯一确定一条直线
  问题转换为 经过某个点的直线，哪条直线上的点最多
*/
public int maxPoints(int[][] points) {
    if(points.length < 3)       return points.length;

    int res = 0;

    for(int i = 0; i < points.length; i++)
```

```
    {
        int duplicates = 0;
        int max = 0;
        HashMap<String, Integer> map = new HashMap<>();
        for(int j = i + 1; j < points.length; j++)
        {
            int x = points[j][0] - points[i][0];
            int y = points[j][1] - points[i][1];

            if(x == 0 && y == 0)
            {
                duplicates++;
                continue;
            }

            /*
            解决斜率的问题， 因为有几个特殊案例， 比如求点[0,0] [95465465,95465466] [95465466,95465465]
            这个时候如果用double等等精度可能上不去
            因此采用辗转相除法， greatest common divisor
            求出最大公约数，然后相除，并作为key 保留
            */
            int gcd = greatestCommonDivisor(x, y);
            x = x / gcd;
            y = y / gcd;
            String key = x + "@" + y;
            map.put(key, map.getOrDefault(key, 0) + 1);
            max = Math.max(max, map.get(key));
        }
        //每算完一个点，就更新一下最大值
        res = Math.max(res, max + duplicates + 1);
    }
    return res;
}
//辗转相除法
private int greatestCommonDivisor(int a, int b)
{
    while(b != 0)
    {
        int temp = a % b;
        a = b;
        b =temp;
    }

    return a;
}
```

150 Evaluate Reverse Polish Notation

150. Evaluate Reverse Polish Notation

难度 中等 160

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are `+`, `-`, `*`, `/`. Each operand may be an integer or another expression.

Note:

- Division between two integers should truncate toward zero.
- The given RPN expression is always valid. That means the expression would always evaluate to a result and there won't be any divide by zero operation.

Example 1:

```
Input: ["2", "1", "+", "3", "*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

Example 2:

```
Input: ["4", "13", "5", "/", "+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6
```

Example 3:

```
Input: ["10", "6", "9", "3", "+", "-11", "*", "/", "*",
"17", "+", "5", "+"]
```

执行结果: 通过 显示详情 >

执行用时: 9 ms , 在所有 Java 提交中击败了 26.66% 的用户

内存消耗: 39.4 MB , 在所有 Java 提交中击败了 7.14% 的用户

炫耀一下:

```java
public int evalRPN(String[] tokens) {
    Deque<String> ops = new ArrayDeque<String>();
    Deque<String> nums = new ArrayDeque<String>();

```

```
    for(int i = 0; i < tokens.length; i++)
    {
        String s = tokens[i];

        if(s.equals("+") || s.equals("-") || s.equals("*") || s.equals("/"))
        {
            String s2 = nums.pop();
            String s1 = nums.pop();
            nums.push(perform(s1, s2, s) + "");
        }
        else
            nums.push(s);
    }

    return Integer.parseInt(nums.pop());
}

private int perform(String num1, String num2, String op)
{
    int a = Integer.parseInt(num1);
    int b = Integer.parseInt(num2);

    if(op.equals("+"))
        return a + b;
    if(op.equals("-"))
        return a - b;
    if(op.equals("*"))
        return a * b;
    if(op.equals("/"))
        return a / b;

    return -1;
}

```

leecode 151-200

151 Reverse Words In a String

151. Reverse Words in a String

难度 中等 194

Given an input string, reverse the string word by word.

Example 1:

```
Input: "the sky is blue"
Output: "blue is sky the"
```

Example 2:

```
Input: "  hello world!  "
Output: "world! hello"
Explanation: Your reversed string should not contain
leading or trailing spaces.
```

Example 3:

```
Input: "a good   example"
Output: "example good a"
Explanation: You need to reduce multiple spaces between
two words to a single space in the reversed string.
```

Note:

- A word is defined as a sequence of non-space characters

```go
func reverseWords(s string) string {
    res := strings.Fields(s)
    length := len(res)

    for i := 0; i < length / 2; i++{
        res[i], res[length - i - 1] = res[length - 1 - i], res[i]
    }

    return strings.Join(res, " ");
}
```


```java
public String reverseWords(String s) {
    if(s.length() == 0)     return "";


    int count = 0;
    StringBuilder sb = new StringBuilder();

    String[] words = s.split(" ");
    for(int i = words.length-1; i >= 0; i--)
    {
        if(words[i].equals(""))    continue;

        sb.append(words[i]).append(" ");
    }
    if(sb.length() == 0)      return "";
    sb.setLength(sb.length()-1);

    return sb.toString();
}
```

152 Maximum Product Subarray

```java
public int maxProduct(int[] nums) {
    int len = nums.length;

    int[] dpMax = new int[len];
    int[] dpMin = new int[len];
    int maxProduct = nums[0];

    for(int i = 0; i < nums.length; i++)
    {
        if(i == 0)
        {
            dpMax[i] = nums[i];
            dpMin[i] = nums[i];
```

```
        }
        else
        {
            if(nums[i] >= 0)
            {
                dpMax[i] = Math.max(dpMax[i-1] * nums[i], nums[i]);
                dpMin[i] = Math.min(dpMin[i-1] * nums[i], nums[i]);
            }
            else
            {
                dpMax[i] = Math.max(dpMin[i-1] * nums[i], nums[i]);
                dpMin[i] = Math.min(dpMax[i-1] * nums[i], nums[i]);
            }
        }

        maxProduct = Math.max(maxProduct, dpMax[i]);
    }

    return maxProduct;
}
```

153 Find Min in Sorted Rotated Array

153. Find Minimum in Rotated Sorted Array

难度 中等 208

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

Find the minimum element.

You may assume no duplicate exists in the array.

Example 1:

```
Input: [3,4,5,1,2]
Output: 1
```

Example 2:

```
Input: [4,5,6,7,0,1,2]
Output: 0
```


```java
/*
  结合着看
LeetCode 33 题：搜索旋转排序数组
LeetCode 81 题：搜索旋转排序数组-ii
LeetCode 153 题：寻找旋转排序数组中的最小值
LeetCode 154 题：寻找旋转排序数组中的最小值-ii
*/
public int findMin(int[] nums) {
    Arrays.sort(nums);
    return nums[0];
}
```

```java
/*
  二分正式解法
*/
   public int findMin(int[] nums) {
    int left = 0;
    int right = nums.length - 1;
    while(left <= right)
    {
        if(nums[left] == nums[right])
            return nums[left];
        int mid = (left + right)/2;

        //注意这里理解是关键
        //如果mid - right 区间是单调递增的，那么最小值就不会再这里出现
        //最小值出现的位置只可能是不连续的地方
        if(nums[mid] > nums[right])
            left = mid + 1;
        else
            right = mid;
    }

    return -1;
   }

作者：imageslr
```

```
链接：https://leetcode-cn.com/problems/find-minimum-in-rotated-sorted-array/solution/yi-wen-jie-jue-4-dao-sou-suo-xuan-zhuan-pai-xu-s-3/
```

154 Find Min in Rotate Sorted Array II

154. Find Minimum in Rotated Sorted Array II

难度 困难 128

Suppose an array sorted in ascending order is rotated at some pivo unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

Find the minimum element.

The array may contain duplicates.

Example 1:

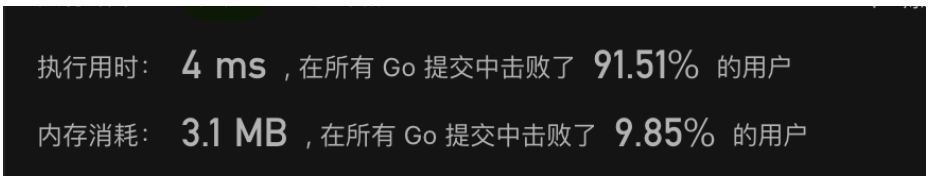
```
Input: [1,3,5]
Output: 1
```

Example 2:

```
Input: [2,2,2,0,1]
Output: 0
```

Note:

- This is a follow up problem to Find Minimum in Rotated Sort Array.
- Would allow duplicates affect the run-time complexity? How and why?




```go
func findMin(nums []int) int {
  left, right := 0, len(nums) - 1

  res := nums[0]
  for ; left <= right; {
    mid := (left + right) / 2

    res = min(res, nums[mid])
```

```

    if nums[left] == nums[mid]{
      left++
    }else if nums[left] < nums[mid]{
      res = min(nums[left], res)
      left = mid + 1
    }else{
      res = min(nums[mid], res)
      res = min(nums[right], res)

      right = mid - 1
    }
  }

  return res
}

func min(a int, b int) int{
  if a < b{
    return a
  }

  return b
}
```

```
public class Solution {
    public int findMin(int[] nums) {
        int len = nums.length;
        if (len == 0) {
            return 0;
        }
        int left = 0;
        int right = len - 1;
        while (left < right) {
            // int mid = left + (right - left) / 2;
            int mid = (left + right) >>> 1;
            if (nums[mid] > nums[right])
                left = mid + 1;
            else if (nums[mid] < nums[right])
                right = mid;
             else {
                assert nums[mid] == nums[right];
                right--;
            }
        }
```

```
21          return nums[left];
22      }
23
24  }
25
26  作者：liweiwei1419
27  链接：https://leetcode-cn.com/problems/find-minimum-in-rotated-sorted-array-
    ii/solution/er-fen-fa-fen-zhi-fa-python-dai-ma-by-liweiwei1419/
```

155 Min Stack

155. Min Stack

难度 简单 591

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

Example 1:

```
Input
["MinStack","push","push","push","getMin","pop","top","getM
[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

Constraints:

- Methods `pop`, `top` and `getMin` operations will always be called on **non-empty** stacks.

通过次数 137,240 | 提交次数 250,531

```java
/*
  采用一个Node 类结构，额外记录我们的最小值，值得学习！
*/
private static class Node{
    int val;
    int min;
    public Node(int val, int min)
    {
```

```
        this.val = val;
        this.min = min;
    }
}
    public Deque<Node> stack;
    public MinStack() {
      stack = new ArrayDeque<>();
    }
    public void push(int x)
    {

        if(stack.isEmpty())
        stack.push(new Node(x, x));
        else
        stack.push(new Node(x, Math.min(x, stack.peek().min)));
    }

    public void pop()
    {
     stack.pop();
    }

    public int top() {
      return stack.peek().val;
    }

    public int getMin() {
      return stack.peek().min;
}
```

156 Binary Tree Upside Down ? 未完成

156. Binary Tree Upside Down

难度 中等 35

Given a binary tree where all the right nodes are either leaf nodes with a sibling (a left node that shares the same parent node) or empty, flip it upside down and turn it into a tree where the original right nodes turned into left leaf nodes. Return the new root.

Example:

```
Input: [1,2,3,4,5]

    1
   / \
  2   3
 / \
4   5

Output: return the root of the binary tree
[4,5,2,#,#,3,1]

   4
  / \
 5   2
    / \
   3   1
```

Clarification:

Confused what `[4,5,2,#,#,3,1]` means? Read more below on how binary tree is serialized on OJ.

The serialization of a binary tree follows a level order traversal, where '#' signifies a path terminator where no node exists below.

Here's an example:

```
   1
  / \
 2   3
    /
   4
    \
     5
```

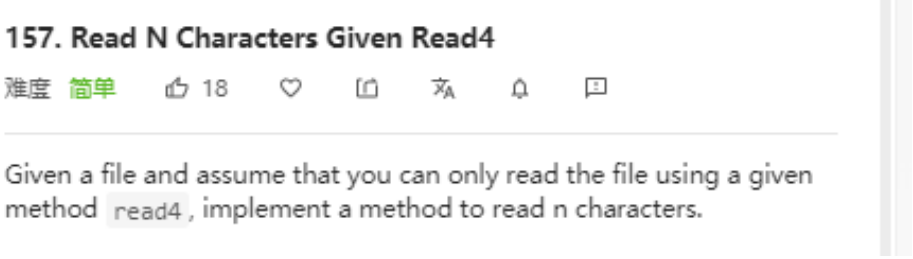
The above binary tree is serialized as `[1,2,3,#,#,4,#,#,5]`.

通过次数 2,670 提交次数 3,604

```
1 class Solution {
2     // 处理之后的根节点
3      TreeNode head;
```

```
4
5       public  TreeNode upsideDownBinaryTree(TreeNode root) {
6           if (root == null)
7               return null;
8
9           dfs(root);
10          return head;
11      }
12
13      public  TreeNode dfs(TreeNode node) {
14          if (node == null)
15              return null;
16
17          if (node.left == null && node.right == null) {
18              if (head == null) {
19                  // 最左边的节点即为实际的根节点
20                  head = node;
21              }
22              return node;
23          }
24          TreeNode left = dfs(node.left);
25          TreeNode right = dfs(node.right);
26          if (left != null) {
27              // 左孩子的左子树为当前的右节点
28              left.left = right;
29              // 左孩子的右子树为当前父节点
30              left.right = node;
31          }
32          // 清空的当前父节点的左右子树
33          node.right = null;
34          node.left = null;
35          return node;
36      }
37  }
38
39  作者：yand-3
40  链接：https://leetcode-cn.com/problems/binary-tree-upside-down/solution/hou-xu-bian-li-di-gui-chu-li-zuo-hai-zi-de-zuo-zi-/
```

157 Read N Characters Given Read 4



Method read4:

The API `read4` reads 4 consecutive characters from the file, then writes those characters into the buffer array `buf`.

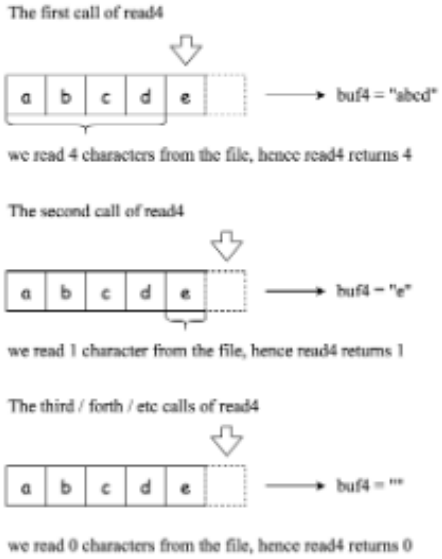
The return value is the number of actual characters read.

Note that `read4()` has its own file pointer, much like `FILE *fp` in C.

Definition of read4:

```
    Parameter:  char[] buf4
    Returns:    int

Note: buf4[] is destination not source, the results from
read4 will be copied to buf4[]
```

Below is a high level example of how `read4` works:




```
File file("abcde"); // File is "abcde", initially file
pointer (fp) points to 'a'
char[] buf4 = new char[4]; // Create buffer with enough
space to store characters
read4(buf4); // read4 returns 4. Now buf = "abcd", fp
points to 'e'
read4(buf4); // read4 returns 1. Now buf = "e", fp
points to end of file
read4(buf4); // read4 returns 0. Now buf = "", fp points
to end of file
```

Method read:

By using the `read4` method, implement the method `read` that reads n characters from the file and store it in the buffer array `buf`. Consider that you **cannot** manipulate the file directly.

The return value is the number of actual characters read.

Definition of read:

```
Parameters: char[] buf, int n
Returns:    int
```

```java
/**
 * The read4 API is defined in the parent class Reader4.
 *     int read4(char[] buf);
 */

public class Solution extends Reader4 {
    /**
     * @param buf Destination buffer
     * @param n   Number of characters to read
     * @return    The number of actual characters read
     */
  public int read(char[] buf, int n) {
        int tmp;
        int length = 0;
        char [] bufTmp = new char[4];
        while ((tmp = read4(bufTmp)) != 0) {
            for (int i = 0; i < tmp && length < n; i++) {
                buf[length++] = bufTmp[i];
            }

        }
        return length;
    }
}

作者：yand-3
    链接：https://leetcode-cn.com/problems/read-n-characters-given-read4/solution/xun-huan-du-xun-huan-tiao-jian-pan-duan-chang-du-s/
```

158 Read N Chara from Read4II 未完成

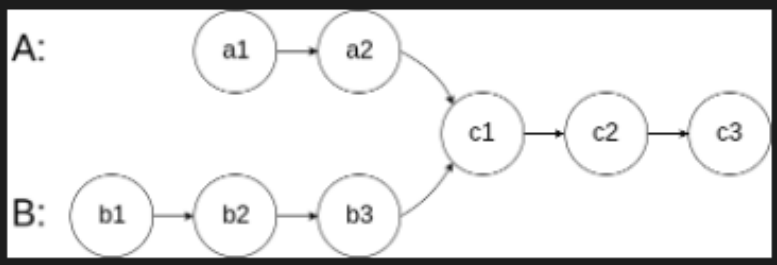
160 Intersection of Two Linked Lists

160. Intersection of Two Linked Lists

难度 简单 719

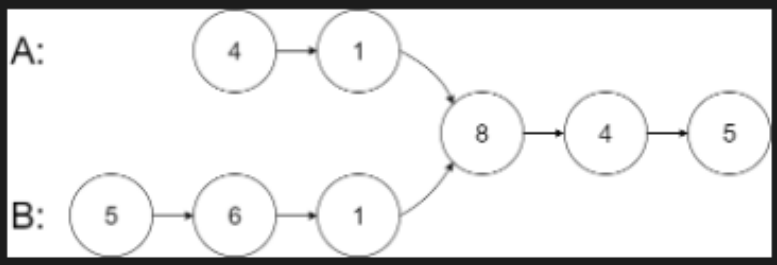
Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:



begin to intersect at node c1.

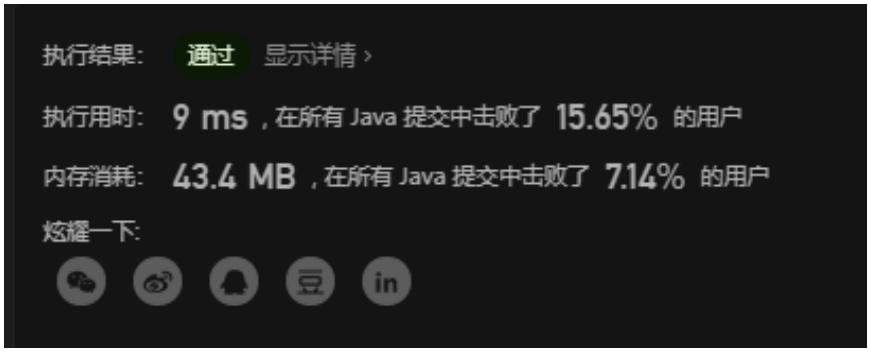
Example 1:




```
Input: intersectVal = 8, listA = [4,1,8,4,5], listB =
[5,6,1,8,4,5], skipA = 2, skipB = 3
Output: Reference of the node with value = 8
```

```java
public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
        ListNode node1 = headA, node2 = headB;
        while(node1 != node2) {
          node1 = node1.next == null ? headB : node1.next;
          node2 = node2.next == null ? headA : node2.next;
        }
        return node1;
    }
ref := https://leetcode-cn.com/problems/intersection-of-two-linked-lists/solution/intersection-of-two-linked-lists-shuang-zhi-zhen-1/
```


```java
/*
  给两个链表加个头结点，这样会方便很多
*/
public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
    if(headA == headB)      return headB;
    if(headA == null && headB != null)      return null;
    if(headA != null && headB == null)      return null;
    if(headA.next == headB)     return headB;
    if(headB.next == headA)     return headA;

    ListNode newA = new ListNode(0);
    ListNode newB = new ListNode(1);
    newA.next = headA;
    newB.next = headB;
    ListNode curA = newA;
    ListNode curB = newB;

    while(curA != null)
    {
        while(curB != null && curA.next != curB.next)
          curB = curB.next;
      if(curB != null && curB.next == curA.next)
        return curB.next;
      else
        {
            curA = curA.next;
            curB = newB;
        }
    }

    return null;
}
```




```java
/*
  哈希表会让时间快一点
*/
public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
    HashSet<ListNode> set = new HashSet<>();
    ListNode cur = headA;
    while(cur != null)
    {
        set.add(cur);
        cur = cur.next;
    }
    cur = headB;
    while(cur != null)
    {
        if(set.contains(cur))
            return cur;
        cur = cur.next;
    }
    return null;
}
```

```java
/*
  第三种方法，就是走两遍，第二遍就会相遇
*/
public class Solution {
    public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
        ListNode ha = headA, hb = headB;
        while (ha != hb) {
            ha = ha != null ? ha.next : headB;
            hb = hb != null ? hb.next : headA;
        }
        return ha;
    }
}

作者：jyd
```

链接：https://leetcode-cn.com/problems/intersection-of-two-linked-lists/solution/intersection-of-two-linked-lists-shuang-zhi-zhen-l/


```java
public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
    List<ListNode> queueOne = new ArrayList<>();
    List<ListNode> queueTwo = new ArrayList<>();

    if(headA == null || headB == null)      return null;

    ListNode cur = headA;
    while(cur != null)
    {
        queueOne.add(cur);
        cur = cur.next;
    }

    cur = headB;

    while(cur != null)
    {
        queueTwo.add(cur);
        cur = cur.next;
    }

    int indexA = queueOne.size() - 1;
    int indexB = queueTwo.size() - 1;

    while(indexA >= 0 && indexB >= 0)
    {
        if(queueOne.get(indexA) != queueTwo.get(indexB))
            if(indexA == queueOne.size() - 1)
                return null;
            else
                return queueOne.get(indexA + 1);

        indexA--;
        indexB--;
    }
    return queueOne.get(indexA + 1);
```

```
}
```

161 One Edit Distance

161. One Edit Distance

难度 中等 26

Given two strings **s** and **t**, determine if they are both one edit distance apart.

Note:

There are 3 possiblities to satisify one edit distance apart:

1. Insert a character into **s** to get **t**
2. Delete a character from **s** to get **t**
3. Replace a character of **s** to get **t**

Example 1:

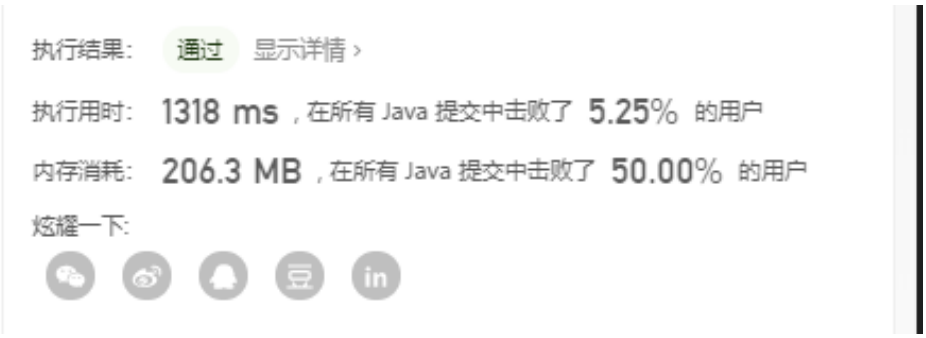
```
Input: s = "ab", t = "acb"
Output: true
Explanation: We can insert 'c' into s to get t.
```

Example 2:

```
Input: s = "cab", t = "ad"
Output: false
Explanation: We cannot get t from s by only one step.
```

Example 3:

```
Input: s = "1203", t = "1213"
Output: true
Explanation: We can replace '0' with '1' to get t.
```




```java
/*
  借鉴了同一类型题的解法,  78 Edit Distance
*/
public boolean isOneEditDistance(String s, String t) {
        if(s.length() - t.length() >= 2)    return false;

```

```
        int[][] dp = new int[s.length()+1][t.length()+1];
        for(int i = 0; i <= s.length(); i++)
            dp[i][0] = i;
        for(int j = 0; j <= t.length(); j++)
            dp[0][j] = j;

        for(int i = 1; i <= s.length(); i++)
            for(int j = 1; j <= t.length(); j++)
            {
                if(s.charAt(i-1) == t.charAt(j-1))
                    dp[i][j] =dp[i-1][j-1];
                else
                    dp[i][j] = Math.min(dp[i][j-1], Math.min(dp[i-1][j], dp[i-1][j-1])) + 1;
            }

        return dp[s.length()][t.length()] == 1;
    }
```

```
/*
  思路很清晰  确保s.length < t.length
  首先排除不可能情况
  之后对前i个进行判断，
    如果相同，往下走
    如果不同，判断剩余的是否相同
    最后如果都相等，那么必须是s.length() + 1 = t.length()
*/
class Solution {
  public boolean isOneEditDistance(String s, String t) {
    int ns = s.length();
    int nt = t.length();

    // Ensure that s is shorter than t.
    if (ns > nt)
      return isOneEditDistance(t, s);

    // The strings are NOT one edit away distance
    // if the length diff is more than 1.
    if (nt - ns > 1)
      return false;

    for (int i = 0; i < ns; i++)
      if (s.charAt(i) != t.charAt(i))
        // if strings have the same length
        if (ns == nt)
          return s.substring(i + 1).equals(t.substring(i + 1));
        // if strings have different lengths
```

```
        else
          return s.substring(i).equals(t.substring(i + 1));

    // If there is no diffs on ns distance
    // the strings are one edit away only if
    // t has one more character.
    return (ns + 1 == nt);
  }
}

作者：LeetCode
链接：https://leetcode-cn.com/problems/one-edit-distance/solution/xiang-ge-wei-1-de-bian-ji-ju-chi-by-leetcode/
```

161 Find Peak Element 二分类题目

162. Find Peak Element

难度 中等 242

A peak element is an element that is greater than its neighbors.

Given an input array `nums`, where `nums[i] ≠ nums[i+1]`, find a peak element and return its index.

The array may contain multiple peaks, in that case return the index to any one of the peaks is fine.

You may imagine that `nums[-1] = nums[n] = -∞`.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function
should return the index number 2.
```

Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 1 or 5
Explanation: Your function can return either index
number 1 where the peak element is 2,
             or index number 5 where the peak element is
6.
```

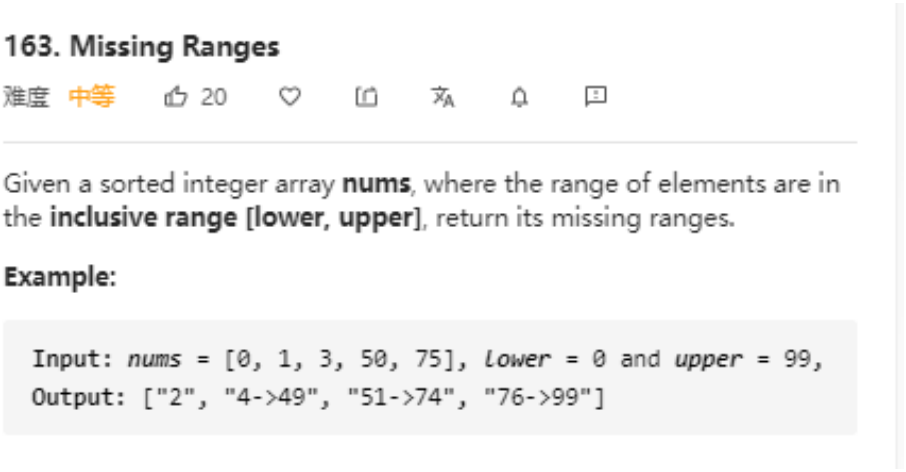
Follow up: Your solution should be in logarithmic complexity.


```java
public int findPeakElement(int[] nums) {
    if(nums.length == 1)        return 0;
    if(nums[0] > nums[1])       return 0;
    if(nums[nums.length-1] > nums[nums.length -2])      return nums.length-1;

    for(int i = 1 ; i < nums.length -1;  i++)
        if(nums[i] > nums[i-1] && nums[i] > nums[i+1])
            return i;
    return -1;
}
```

```java
/*
  正宗二分解法， logarithematic
  注意这里，由于数组两边nums[-1] = nums[n] = -∞
  因此，我们只要找到一个局部上升的梯度，就可以在左/右找到peak
  这个性质很重要
*/
class Solution {
    public int findPeakElement(int[] nums) {
        int left = 0, right = nums.length - 1;
        for (; left < right; )
        {
            int mid = left + (right - left) / 2;
            if (nums[mid] > nums[mid + 1]) {
                right = mid;  //注意这里是包含的，因为mid可能是peak
            } else {
                left = mid + 1;
            }
        }
        return left;
    }
}

作者：guanpengchn
```

链接：https://leetcode-cn.com/problems/find-peak-element/solution/hua-jie-suan-fa-162-xun-zhao-feng-zhi-by-guanpengc/

163 Missing Ranges



163. Missing Ranges

难度 中等 20

Given a sorted integer array **nums**, where the range of elements are in the **inclusive range [lower, upper]**, return its missing ranges.

Example:

```
Input: nums = [0, 1, 3, 50, 75], lower = 0 and upper = 99,
Output: ["2", "4->49", "51->74", "76->99"]
```

执行结果: 通过 显示详情 >

执行用时: 15 ms , 在所有 Java 提交中击败了 5.56% 的用户

内存消耗: 38.4 MB , 在所有 Java 提交中击败了 20.00% 的用户

炫耀一下:

```java
/*
  通过38/40
  没办法通过Integer.MAX_VALUE

  经验教训，碰到超过int情况过不去的，改成long
*/
    public List<String> findMissingRanges(int[] nums, int lower, int upper) {
        long lowerl = lower;
        long upperl = upper;
        Deque<String> stack = new ArrayDeque<>();
        List<String> res = new ArrayList<>();

        if(nums.length == 0)
        {
            if(lower == upper){
                res.add(lower + "");
```

```
                return res;
            }
            else{
                StringBuilder sb = new StringBuilder();
                sb.append(lower + "").append("->").append(upper + "");
                res.add(sb.toString());
                return res;
            }
        }

        for(int i = 0; i <nums.length-1; i++)
        {
            int dif = nums[i+1] - nums[i];
            if(dif == 0 || dif == 1)        continue;
            else if(dif == 2)
                stack.addLast((nums[i]+1) + "");
            else
            {
                StringBuilder sb = new StringBuilder();
                sb.append(nums[i] + 1).append("->").append(nums[i+1]-1);
                stack.addLast(sb.toString());
            }
        }
        long dif_low = nums[0] - lowerl;
        if(dif_low == 0)    {}
        else if (dif_low == 1)  stack.addFirst(lowerl + "");
        else if (dif_low >= 2)  stack.addFirst(lowerl + "" + "->" + (nums[0]-1));

        long  dif_hi = upperl - nums[nums.length-1];
        if(dif_hi == 0)    {}
        else if (dif_hi == 1)  stack.addLast(upperl + "");
        else if (dif_hi >= 2)  stack.addLast(nums[nums.length-1]+1 + "" + "->" +
upperl);

        for(String s : stack)
            res.add(s);

        return res;
    }
```

```
/*
  双指针蛮巧妙
*/
public List<String> findMissingRanges(int[] nums, int lower, int upper) {
    List<String> res = new ArrayList<>();
    long pre = (long)lower - 1; // prevent 'int' overflow
    for (int i = 0; i < nums.length; i++) {
```

```
        if (nums[i] - pre == 2) res.add(String.valueOf(pre + 1));
        else if (nums[i] - pre > 2) res.add((pre + 1) + "->" + (nums[i] - 1));
        pre = nums[i]; // 'int' to 'long'
    }
    if (upper - pre == 1) res.add(String.valueOf(pre + 1));
    else if (upper - pre > 1) res.add((pre + 1) + "->" + upper);
    return res;
}


作者：jyd
链接：https://leetcode-cn.com/problems/missing-ranges/solution/missing-ranges-shuang-zhi-zhen-fa-by-jyd/
```

164 Max Gap 搞懂桶排序

164. Maximum Gap

难度 困难 172

Given an unsorted array, find the maximum difference between the successive elements in its sorted form.

Return 0 if the array contains less than 2 elements.

Example 1:

```
Input: [3,6,9,1]
Output: 3
Explanation: The sorted form of the array is [1,3,6,9], either
             (3,6) or (6,9) has the maximum difference 3.
```

Example 2:

```
Input: [10]
Output: 0
Explanation: The array contains less than 2 elements, therefore return 0.
```

Note:

- You may assume all elements in the array are non-negative integers and fit in the 32-bit signed integer range.
- Try to solve it in linear time/space.

```
/*
```

```
  将箱子能放的数字个数为interval, 给定的数字最小是min,  最大是max
  箱子划分范围:
  min + 0 * interval ~ min + 1 * interval - 1
  min + 1 * interval ~ min + 2 * interval - 1
  min + 2 * interval ~ min + 3 * interval - 1
  ....
  min + (n-2) * interval ~ min + (n-1) * interval - 1

  通过(nums[i] - min) / interval 得到当前数字应该放到的箱子编号
  同时需要保证至少有一个空箱子,  这样
    1 可以保证箱子内部不会产生最大Gap
    2 如果在计算中,  跳过某一个空箱子，得到的gap一定会大于interval

  如果有 n - 2个数字,  箱子数目多于N-2,  就一定会有空箱子。
  interval = (max - min) / 箱子数目
    因为我们想interval 尽可能大,  因此将箱子数目取得最小，但同时要满足 箱子数目 > n - 2
    因此，箱子数目为n - 1
  So interval = (max - min) / (n - 1) 向上取整
    比如原来范围是[0, 5.5], 内部最大gap = 5 - 0  向上取整，变成[0, 6)
    内部最大gap依然是5-0

    我们把 0 3 4 6 23 28 29 33 38 依次装到三个箱子中
        0             1             2             3
     -------       -------       -------       -------
    |  3 4  |     |       |     | 29    |     | 33    |
    |   6   |     |       |     |  23   |     |       |
    |  0    |     |       |     |  28   |     |  38   |
     -------       -------       -------       -------
      0 - 9         10 - 19       20 - 29       30 - 39
    我们把每个箱子的最大值和最小值表示出来
     min  max      min  max      min  max  min  max
     0     6       -     -       23   29   33   38
*/
public int maximumGap(int[] nums) {
    if (nums.length <= 1)
        return 0;

    int n = nums.length;
    int min = nums[0];
    int max = nums[0];
    //找出最大值、最小值
    for (int i = 1; i < n; i++)
    {
        min = Math.min(nums[i], min);
        max = Math.max(nums[i], max);
    }

    if(max - min == 0)
        return 0;
```

```java


    //算出每个箱子的范围
    int interval = (int) Math.ceil((double)(max - min) / (n - 1));

    //每个箱子里数字的最小值和最大值
    int[] bucketMin = new int[n - 1];
    int[] bucketMax = new int[n - 1];

    //最小值初始为 Integer.MAX_VALUE
    Arrays.fill(bucketMin, Integer.MAX_VALUE);
    //最大值初始化为 -1，因为题目告诉我们所有数字是非负数
    Arrays.fill(bucketMax, -1);

    //考虑每个数字
    for (int i = 0; i < nums.length; i++) {
        //当前数字所在箱子编号
        int index = (nums[i] - min) / interval;
        //最大数和最小数不需要考虑
        if(nums[i] == min || nums[i] == max)
            continue;

        //更新当前数字所在箱子的最小值和最大值
        bucketMin[index] = Math.min(nums[i], bucketMin[index]);
        bucketMax[index] = Math.max(nums[i], bucketMax[index]);
    }

    int maxGap = 0;
    //min 看做第 -1 个箱子的最大值
    int previousMax = min;
    //从第 0 个箱子开始计算
    for (int i = 0; i < n - 1; i++) {
        //最大值是 -1 说明箱子中没有数字，直接跳过
        if (bucketMax[i] == -1)
            continue;


        //当前箱子的最小值减去前一个箱子的最大值
        maxGap = Math.max(bucketMin[i] - previousMax, maxGap);
        previousMax = bucketMax[i];
    }

    //最大值可能处于边界，不在箱子中，需要单独考虑
    maxGap = Math.max(max - previousMax, maxGap);
    return maxGap;

}

作者：windliang
```

```
链接：https://leetcode-cn.com/problems/maximum-gap/solution/xiang-xi-tong-su-de-si-lu-fen-xi-duo-jie-fa-by--39/
```

165 Compare Version Numbers

165. Compare Version Numbers

难度 中等 91

Compare two version numbers version1 and version2.
If `version1 > version2` return `1;` if `version1 < version2` return `-1;` otherwise return `0`.

You may assume that the version strings are non-empty and contain only digits and the `.` character.

The `.` character does not represent a decimal point and is used to separate number sequences.

For instance, `2.5` is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.

You may assume the default revision number for each level of a version number to be `0`. For example, version number `3.4` has a revision number of `3` and `4` for its first and second level revision number. Its third and fourth level revision number are both `0`.

Example 1:

```
Input: version1 = "0.1", version2 = "1.1"
Output: -1
```

Example 2:

```
Input: version1 = "1.0.1", version2 = "1"
Output: 1
```

Example 3:

```
Input: version1 = "7.5.2.4", version2 = "7.5.3"
Output: -1
```

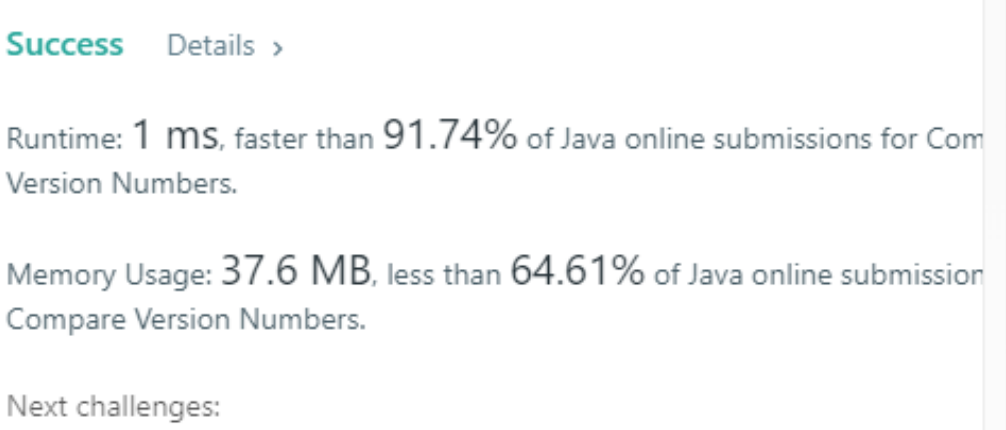
Example 4:

```
Input: version1 = "1.01", version2 = "1.001"
Output: 0
```

```
Explanation: Ignoring leading zeroes, both “01” and
“001" represent the same number “1”
```

Example 5:

```
Input: version1 = "1.0", version2 = "1.0.0"
Output: 0
Explanation: The first version number does not have a
third level revision number, which means its third level
revision number is default to "0"
```




```java
public int compareVersion(String version1, String version2) {

    String[] strs1 = version1.split("\\.");
    String[] strs2 = version2.split("\\.");

    int i = 0;
    for(; i < strs1.length && i < strs2.length; i++)
    {
        int num1 = Integer.parseInt(strs1[i]);
        int num2 = Integer.parseInt(strs2[i]);

        if(num1 > num2)
            return 1;
        else if(num1 < num2)
            return -1;
        else
            continue;
    }

    while(i < strs1.length)
        if(Integer.parseInt(strs1[i]) > 0)
            return 1;
    else
        i++;
    while(i < strs2.length)
```

```
26          if(Integer.parseInt(strs2[i]) > 0)
27              return -1;
28      else
29          i++;
30
31      return 0;
32  }
```

166 Fraction to Recurring Decimal 哈希Map的又一应用

166. Fraction to Recurring Decimal

难度 中等 143

Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.

If the fractional part is repeating, enclose the repeating part in parentheses.

Example 1:

```
Input: numerator = 1, denominator = 2
Output: "0.5"
```

Example 2:

```
Input: numerator = 2, denominator = 1
Output: "2"
```

Example 3:

```
Input: numerator = 2, denominator = 3
Output: "0.(6)"
```

```
1  /*
2    使用HashMap 来存储余数，以及对应的索引。
3    这样当相同的余数来临，我们就知道是否是重复了
4    先处理整数部分，再处理小数部分
5
```

```java

  这段代码写的相当优雅
*/

public class Solution {
    public String fractionToDecimal(int numerator, int denominator) {
        if (numerator == 0) {
            return "0";
        }
        StringBuilder res = new StringBuilder();

        // 处理符号
        res.append(((numerator > 0) ^ (denominator > 0)) ? "-" : "");
        long num = Math.abs((long)numerator);
        long den = Math.abs((long)denominator);

        // 处理整数
        res.append(num / den);
        num %= den;
        if (num == 0)
            return res.toString();


        // 处理小数部分
        res.append(".");
        HashMap<Long, Integer> map = new HashMap<Long, Integer>();
        map.put(num, res.length());
        while (num != 0) {
            num *= 10;
            res.append(num / den);
            //这里就相当于拿到每次的余数，如果在一个过程中，余数重复了，说明就是重复小数
            num %= den;
            if (map.containsKey(num)) {
                int index = map.get(num);
                res.insert(index, "(");
                res.append(")");
                break;
            }
            else {
                //记录不同的余数出现的位置是哪里，便于插入括号
                map.put(num, res.length());
            }
        }
        return res.toString();
    }
}
/*
  对于 14/17 举例,下面这个数字是num每次被den取余后的结果，一旦发现出现重复4， 就停止，因为下面的操作就会重复
```

```
      4 6 9 5 16 7 2 3 13 11 8 12 1 10 15 14 4
*/
```

168 Excel Sheet Column Title

168. Excel Sheet Column Title

难度 简单 238

Given a positive integer, return its corresponding column title as appear in an Excel sheet.

For example:

```
1 -> A
2 -> B
3 -> C
...
26 -> Z
27 -> AA
28 -> AB
...
```

Example 1:

```
Input: 1
Output: "A"
```

Example 2:

```
Input: 28
Output: "AB"
```

Example 3:

```
Input: 701
Output: "ZY"
```

```
/*
每一次除以26，  都会把右侧的数字给除掉
A   0x26 + 1     AA    1x26+ 1     BA  2x26+ 1     ...     ZA  26x26+ 1     AAA  1x26²+1x26+ 1
B   0x26 + 2     AB    1x26+ 2     BB  2x26+ 2     ...     ZB  26x26+ 2     AAB  1x26²+1x26+ 2
```

```
.   .     ..    .....     ..  .......     ...     ..  ........     ...
.............
.   .     ..    .....     ..  .......     ...     ..  ........     ...
.............
.   .     ..    .....     ..  .......     ...     ..  ........     ...
.............
Z  0x26 + 26     AZ    1x26+26     BZ  2×26+26     ...     ZZ  26×26+26     AAZ
1×26²+1×26+26

*/

//为了保证能拿到A,  我们会先给n-1,  因为 %26 ==0,  代表是26 -> Z,
//当n = 1,  bit == 0,  代表是A
//当n = 26,  bit == 25,  代表是Z
// use (n-1)%26 instead, then we get a number range from 0 to 25.
public String convertToTitle(int n) {
    StringBuilder res = new StringBuilder();
    while(n != 0)
    {
        int bit = (n-1) % 26;
        res.append((char)('A' + bit));
        n = (n-1) / 26;
    }
    return res.reverse().toString();
}
```

169 Majority Element

169. Majority Element

难度 简单 659

Given an array of size n, find the majority element. The majority element is the element that appears **more than** `⌊ n/2 ⌋` times.

You may assume that the array is non-empty and the majority element always exist in the array.

Example 1:

```
Input: [3,2,3]
Output: 3
```

Example 2:

```
Input: [2,2,1,1,1,2,2]
Output: 2
```

```java
//二刷
public int majorityElement(int[] nums) {
        int count = 1;
        int ele = nums[0];

        for(int i = 1; i < nums.length; i++){
            if(nums[i] == ele){
                count++;
            }else{
                count--;
                if(count == 0){
                    ele = nums[i];
                    count = 1;
                }
            }
        }

        return ele;
    }
```

执行结果: 通过 显示详情 >

执行用时: 14 ms , 在所有 Java 提交中击败了 33.23% 的用户

内存消耗: 45 MB , 在所有 Java 提交中击败了 5.71% 的用户

炫耀一下:

```java
public int majorityElement(int[] nums) {
    HashMap<Integer, Integer> map = new HashMap<>();

    for(int num : nums)
        map.put(num, map.getOrDefault(num, 0) + 1);

    int filter = nums.length / 2 ;

    for(Integer i : map.keySet())
        if(map.get(i) > filter)
            return i;
    return -1;
}
```

170 Two Sum III data structure

170. Two Sum III - Data structure design

难度 简单 21

Design and implement a TwoSum class. It should support the following operations: `add` and `find`.

`add` - Add the number to an internal data structure.
`find` - Find if there exists any pair of numbers which sum is equal to the value.

Example 1:

```
add(1); add(3); add(5);
find(4) -> true
find(7) -> false
```

Example 2:

```
add(3); add(1); add(2);
find(3) -> true
find(6) -> false
```

执行结果: 通过 显示详情 >

执行用时: 115 ms , 在所有 Java 提交中击败了 73.72% 的用户

内存消耗: 48.3 MB , 在所有 Java 提交中击败了 100.00% 的用户

炫耀一下:

```java
private HashSet<Integer> set;
private HashMap<Integer, Integer> list;
/** Initialize your data structure here. */
public TwoSum() {
    set = new HashSet<>();
    list = new HashMap<>();
}

/** Add the number to an internal data structure.. */
public void add(int number) {
    set.add(number);
    list.put(number, list.getOrDefault(number, 0) + 1);
}

/** Find if there exists any pair of numbers which sum is equal to the value. */
public boolean find(int value) {
    for(Integer i : set)
        if(set.contains(value-i))
            if(i != value - i)
                return true;
            else
                if(list.get(i) > 1)
                    return true;

    return false;
}
```

171 Excel Sheet Column Number

171. Excel Sheet Column Number

难度 简单 157

Given a column title as appear in an Excel sheet, return its corresponding column number.

For example:

```
    A -> 1
    B -> 2
    C -> 3
    ...
    Z -> 26
    AA -> 27
    AB -> 28
    ...
```

Example 1:

```
Input: "A"
Output: 1
```

Example 2:

```
Input: "AB"
Output: 28
```

Example 3:

```
Input: "ZY"
Output: 701
```

Constraints:

- `1 <= s.length <= 7`
- `s` consists only of uppercase English letters.
- `s` is between "A" and "FXSHRXW".

通过次数 43,852 | 提交次数 64,985

执行结果: 通过 显示详情 >

执行用时: 1 ms，在所有 Java 提交中击败了 100.00% 的用户

内存消耗: 38.2 MB，在所有 Java 提交中击败了 5.55% 的用户

炫耀一下:

```java
public int titleToNumber(String s) {
    int n= 1;
    int res = 0;
    for(int i = s.length()-1; i >= 0; i--, n *= 26)
        res += (s.charAt(i) - 'A' + 1) * n;
    return res;
}
```

172 Factorial Trailing zeroes

172. Factorial Trailing Zeroes

难度 简单 310

Given an integer n, return the number of trailing zeroes in n!.

Example 1:

```
Input: 3
Output: 0
Explanation: 3! = 6, no trailing zero.
```

Example 2:

```
Input: 5
Output: 1
Explanation: 5! = 120, one trailing zero.
```

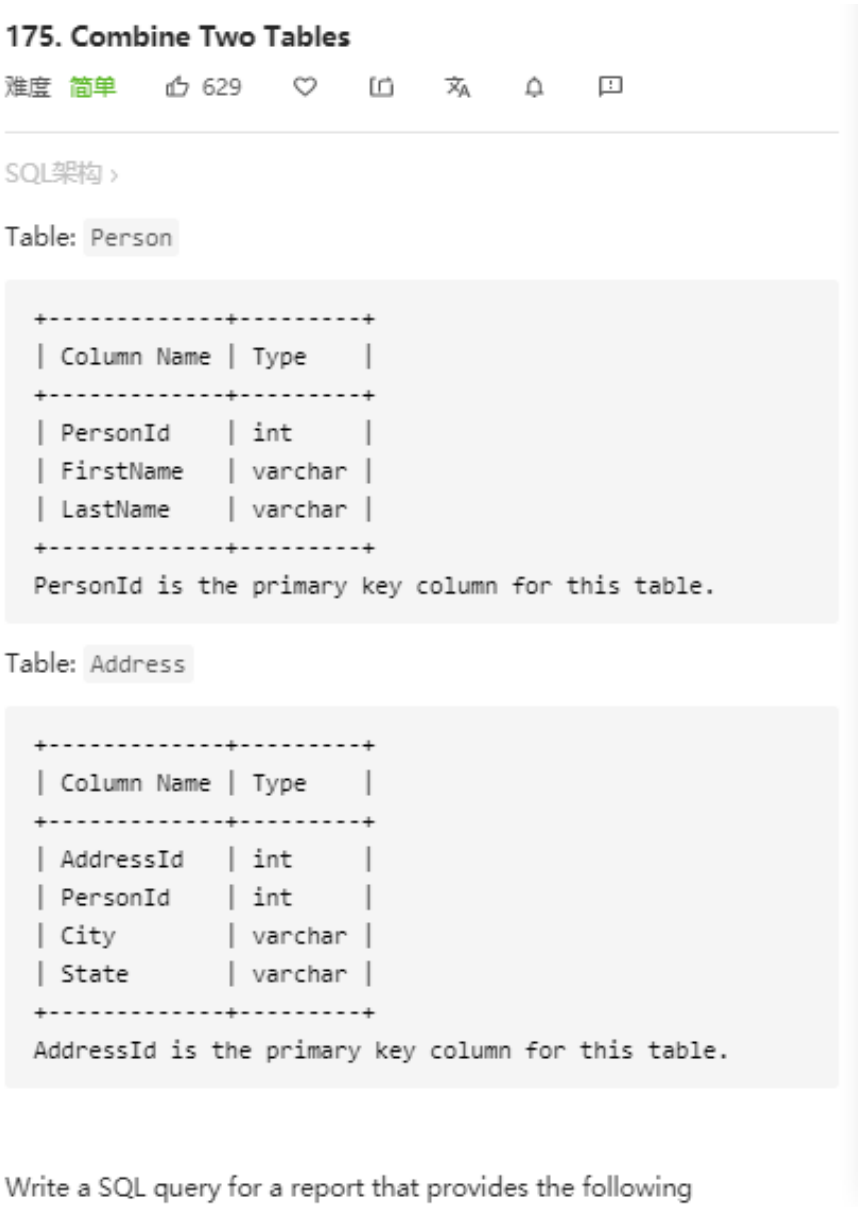
Note: Your solution should be in logarithmic time complexity.

```java
/*
  思路:
  我们用n / 5, 可以筛选出比如5,15 35 等等的个数
  注意到25， 是 5 * 5 ，也就是说，如果我们只用n /5， 会导致漏掉一个
  而25*5= 125 有 3 个5， 会漏掉2个5
  因此，我们每次让n /5 就是为了计算对应的个数，从而不漏
*/
public int trailingZeroes(int n) {
    int count = 0;
    while (n > 0) {
        count += n / 5;
        n = n / 5;
    }
```

```
    return count;
}

作者：windliang
链接：https://leetcode-cn.com/problems/factorial-trailing-zeroes/solution/xiang-xi-tong-su-de-si-lu-fen-xi-by-windliang-3/
```

174 Dungeon game 暂且略过

175 Combine Two Tables



175. Combine Two Tables

难度 简单 629

SQL架构 >

Table: `Person`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| PersonId    | int     |
| FirstName   | varchar |
| LastName    | varchar |
+-------------+---------+
PersonId is the primary key column for this table.
```

Table: `Address`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| AddressId   | int     |
| PersonId    | int     |
| City        | varchar |
| State       | varchar |
+-------------+---------+
AddressId is the primary key column for this table.
```

Write a SQL query for a report that provides the following

```sql
#注意：如果没有某个人的地址信息，使用 where 子句过滤记录将失败，因为它不会显示姓名信息
select FirstName, LastName, City, State
from Person left join Address
on Person.PersonId = Address.PersonId;
```

176 Second Highest Salary

176. Second Highest Salary

难度 简单 591

SQL架构 >

Write a SQL query to get the second highest salary from the `Employee` table.

```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

For example, given the above Employee table, the query should return `200` as the second highest salary. If there is no second highest salary, then the query should return `null`.

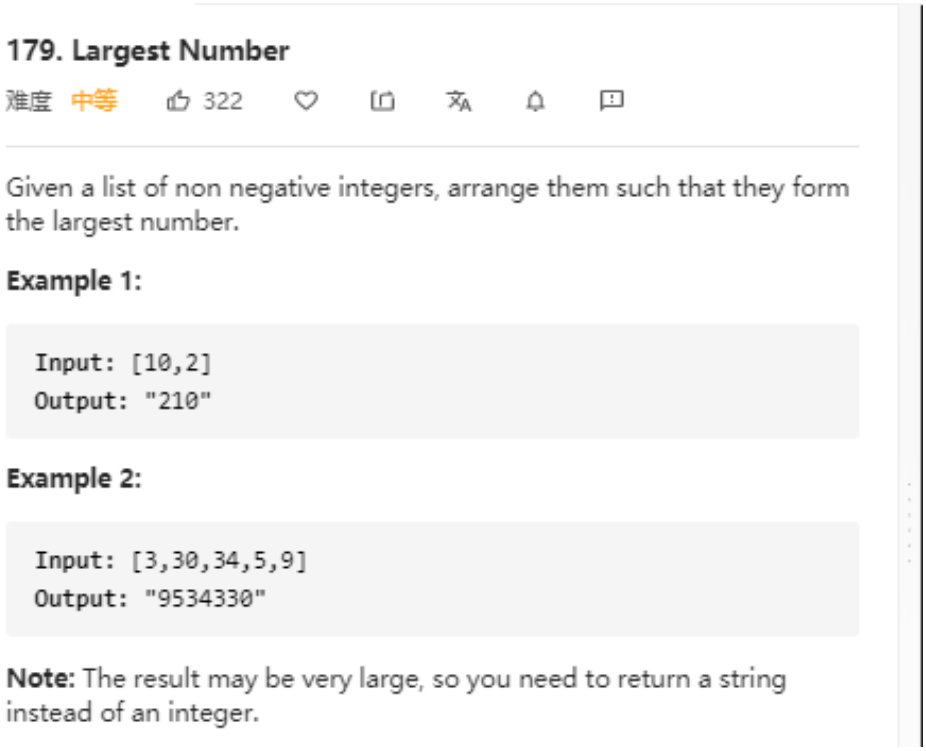
```
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

```sql
# 用if null 解决，如果没有该记录的情况
SELECT
    IFNULL(
      (SELECT DISTINCT Salary
       FROM Employee
       ORDER BY Salary DESC
        LIMIT 1 OFFSET 1),
    NULL) AS SecondHighestSalary

"作者：LeetCode
链接：https://leetcode-cn.com/problems/second-highest-salary/solution/di-er-gao-de-xin-shui-by-leetcode/"
```

177 Sql 未完成

178 Sql 未完成

179 Largest Number 重写排序规则



179. Largest Number

难度 中等 322

Given a list of non negative integers, arrange them such that they form the largest number.

Example 1:

```
Input: [10,2]
Output: "210"
```

Example 2:

```
Input: [3,30,34,5,9]
Output: "9534330"
```

Note: The result may be very large, so you need to return a string instead of an integer.

```java
public String largestNumber(int[] nums) {
    List<String> res = new ArrayList<>();
    for(int num : nums)
        res.add(num + "");

    res.sort(new Comparator<String>() {
        @Override
        public int compare(String o1, String o2) {
            String str1 = o1 + o2;
            String str2 = o2 + o1;

            return str2.compareTo(str1);
        }
    });

    StringBuilder ans = new StringBuilder();
```

```
        for(int i = 0; i < res.size(); i++)
            ans.append(res.get(i));

        if(ans.charAt(0) == '0')
            return "0";
        return ans.toString();
    }

```

```
/*
  思路是：希望拿到该数组的某种排列，使得对于每一个元素a,b都有
  a~b > b~a  这里~代表连接的意思

  那么可以理解为，需要我们重新定义排序规则
*/
class Solution {
    private class LargerNumberComparator implements Comparator<String> {
        @Override
        public int compare(String a, String b) {
            String order1 = a + b;
            String order2 = b + a;
           return order2.compareTo(order1);
        }
    }

    public String largestNumber(int[] nums) {
        // Get input integers as strings.
        String[] asStrs = new String[nums.length];
        for (int i = 0; i < nums.length; i++)
            asStrs[i] = String.valueOf(nums[i]);


        // Sort strings according to custom comparator.
        Arrays.sort(asStrs, new LargerNumberComparator());

        // If, after being sorted, the largest number is `0`, the entire number
        // is zero.
        if (asStrs[0].equals("0"))
            return "0";


        // Build largest number from sorted array.
```

```
        String largestNumberStr = new String();
        for (String numAsStr : asStrs)
            largestNumberStr += numAsStr;


        return largestNumberStr;
    }
}

作者：LeetCode
链接：https://leetcode-cn.com/problems/largest-number/solution/zui-da-shu-by-leetcode/
```

180 Consecutive Numbers Sql 未完成

181 Employees Earning More Than Their Managers

```mysql
# Write your MySQL query statement below
select e.Name as Employee
from Employee e
where e.Salary >
    (
        select Salary
        from Employee e1
        where e.ManagerId = e1.id
    )
```

182 Duplicate Emails

提交时间	提交结果	运行时间	内存消耗	语
1 分钟前	通过	255 ms	0 B	My

```mysql
SELECT Email
FROM Person
group by Email
having COUNT(Id) > 1
```

183 Customers Who Never Orders

```sql
SELECT c.Name as Customers
FROM Customers as c
WHERE c.ID not in
    (SELECT DISTINCT CustomerID
    FROM Orders)
```

184 部门工资最高的员工

184. Department Highest Salary

难度 中等 325

SQL架构 ›

The `Employee` table holds all employees. Every employee has an Id, a salary, and there is also a column for the department Id.

```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 70000  | 1            |
| 2  | Jim   | 90000  | 1            |
| 3  | Henry | 80000  | 2            |
| 4  | Sam   | 60000  | 2            |
| 5  | Max   | 90000  | 1            |
+----+-------+--------+--------------+
```

The `Department` table holds all departments of the company.

```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

Write a SQL query to find employees who have the highest salary in each of the departments. For the above tables, your SQL query should

```sql
SELECT
    Department.Name AS Department,
    Employee.Name AS Employee,
    Salary
```

```
FROM
    Employee,
    Department
WHERE Employee.DepartmentId = Department.Id
    AND
    (Employee.DepartmentId, Employee.Salary)  /*注意这种写法*/
    IN (SELECT DepartmentId, max(Salary)
        FROM Employee
        GROUP BY DepartmentId
       )
```

186 Reverse Words In a String II

186. Reverse Words in a String II

难度 中等 30

Given an input string , reverse the string word by word.

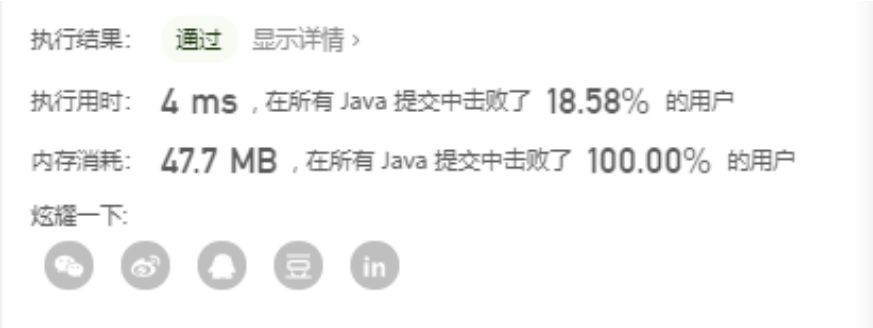
Example:

```
Input:  ["t","h","e"," ","s","k","y"," ","i","s","
","b","l","u","e"]
Output: ["b","l","u","e"," ","i","s"," ","s","k","y","
","t","h","e"]
```

Note:

- A word is defined as a sequence of non-space characters.
- The input string does not contain leading or trailing spaces.
- The words are always separated by a single space.

Follow up: Could you do it in-place without allocating extra space?




```java
/*
  这种方式就是暴力解，我们下面试试双指针
*/
public void reverseWords(char[] s) {
    List<StringBuilder> jar = new ArrayList<>();

    int index = 0;
    while(index != s.length)
    {
        StringBuilder sb = new StringBuilder();
        while(index < s.length &&  s[index] == ' ')
            index++;
        while(index < s.length && s[index] != ' ')
            sb.append(s[index++]);
        jar.add(sb);
    }

    int indexx = 0;
    int num = jar.size()-1;
    while(num >= 0 && indexx < s.length)
    {
        int pointer = 0;
        StringBuilder sb = jar.get(num--);
        char[] chars = sb.toString().toCharArray();

        for(;pointer < chars.length;)
            s[indexx++] = chars[pointer++];
        if(indexx < s.length)
            s[indexx++] = ' ';
    }
}
```

```java
/*
  思路:  翻转两次字符串，一次全局，一次单个反转
*/
private void reverse(char[] s, int start, int end) {
        while (start < end) {
```

```
            char tmp = s[start];
            s[start] = s[end];
            s[end] = tmp;
            start++;
            end--;
        }
    }

public void reverseWords(char[] s) {
    // 两次翻转即可，第一次全局翻转，第二次翻转各个单词
    int len = s.length;
    reverse(s, 0, len - 1);

    int start = 0;
    for (int i = 0; i < len; i++) {
        if (s[i] == ' ') {
            // 翻转前面的单词
            reverse(s, start, i-1);
            start = i + 1;
        }
    }

    // 翻转最后一个单词
    reverse(s, start, len - 1);
}


作者：yuruiyin
链接：https://leetcode-cn.com/problems/reverse-words-in-a-string-ii/solution/java-liang-ci-fan-zhuan-by-npe_tle/
```

187 Repeated DNA Sequences

187. Repeated DNA Sequences

难度 中等 102

All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: "ACGAATTCCG". When studying DNA, it is sometimes useful to identify repeated sequences within the DNA.

Write a function to find all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule.

Example:

```
Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC", "CCCCCAAAAA"]
```

执行结果: 通过 显示详情 >

执行用时: 21 ms , 在所有 Java 提交中击败了 71.45% 的用户

内存消耗: 48.8 MB , 在所有 Java 提交中击败了 25.00% 的用户

炫耀一下:

```java
public List<String> findRepeatedDnaSequences(String s) {
    List<String> res= new ArrayList<>();
    HashMap<String, Integer> map = new HashMap<>();

    for(int i = 0; i <= s.length() - 10; i++)
    {
        String frac = s.substring(i, i+10);
        map.put(frac, map.getOrDefault(frac, 0) + 1);
    }

    for(String str : map.keySet())
        if(map.get(str) > 1)
            res.add(str);

    return res;
}
```

189 Rotate Array

189. Rotate Array

难度 简单 617

Given an array, rotate the array to the right by k steps, where k is non-negative.

Follow up:

- Try to come up as many solutions as you can, there are at least 3 different ways to solve this problem.
- Could you do it in-place with O(1) extra space?

Example 1:

```
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
```

Example 2:

```
Input: nums = [-1,-100,3,99], k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]
```

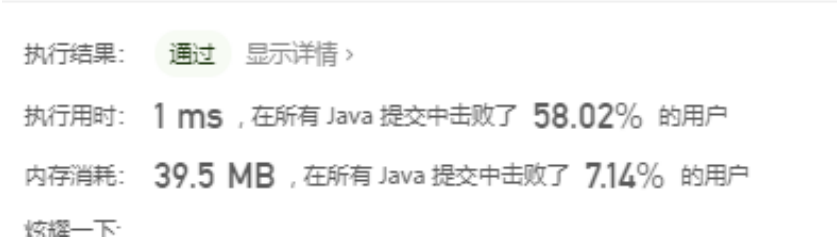
执行结果: 通过 显示详情 >

执行用时: 4 ms , 在所有 Java 提交中击败了 35.05% 的用户

内存消耗: 40.1 MB , 在所有 Java 提交中击败了 7.14% 的用户

炫耀一下:

```java
public void rotate(int[] nums, int k) {
    if(nums.length == k)        return;
    if(nums.length < k)     k = k - nums.length;
    if(k == 0)      return;

    List<Integer> record = new ArrayList<>();
    for(int i = nums.length - k; record.size() < nums.length; i = (i+1) %
nums.length)
        record.add(nums[i]);

    int index = 0;
    for(Integer i : record)
        nums[index++] = i;
}
```




```java
public void rotate(int[] nums, int k) {
    if(nums.length == k || k == 0)        return;
    if(nums.length < k)     k = k - nums.length;

    int[] newNums = new int[nums.length];

    for(int i = nums.length - k,index = 0; index <nums.length; index++,i =
(i+1)%nums.length)
        newNums[index] = nums[i];
    for(int i = 0; i < nums.length; i++)
        nums[i] = newNums[i];
}
```

190 Reverse Bits

190. Reverse Bits

难度 简单 183

Reverse bits of a given 32 bits unsigned integer.

Example 1:

```
Input: 00000010100101000001111010011100
Output: 00111001011110000010100101000000
Explanation: The input binary string
00000010100101000001111010011100 represents the unsigned
integer 43261596, so return 964176192 which its binary
representation is 00111001011110000010100101000000.
```

Example 2:

```
Input: 11111111111111111111111111111101
Output: 10111111111111111111111111111111
Explanation: The input binary string
11111111111111111111111111111101 represents the unsigned
integer 4294967293, so return 3221225471 which its
binary representation is
10111111111111111111111111111111.
```

Note:

- Note that in some languages such as Java, there is no unsigned integer type. In this case, both input and output will be given as signed integer type and should not affect your implementation, as the internal binary representation of the integer is the same whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in **Example 2** above the input represents the signed integer `-3` and the output represents the signed integer `-1073741825`.

```java
public int reverseBits(int n)
{
    int res = 0;
    int count = 0;
    while(count < 32)
    {
        res <<= 1;  //左移一位，空出位置给下面的
        res |= (n & 1); //把拿到的最低位给res
        n >>= 1;
        count++;
    }
    return res;
}
```

191 Number of 1 bits

191. Number of 1 Bits

难度 简单 183

Write a function that takes an unsigned integer and return the number of '1' bits it has (also known as the Hamming weight).

Example 1:

```
Input: 00000000000000000000000000001011
Output: 3
Explanation: The input binary string
00000000000000000000000000001011 has a total of three
'1' bits.
```

Example 2:

```
Input: 00000000000000000000000010000000
Output: 1
Explanation: The input binary string
00000000000000000000000010000000 has a total of one '1'
bit.
```

Example 3:

```
Input: 11111111111111111111111111111101
Output: 31
Explanation: The input binary string
11111111111111111111111111111101 has a total of thirty
one '1' bits.
```

Note:

- Note that in some languages such as Java, there is no unsigned integer type. In this case, the input will be given as signed integer type and should not affect your implementation, as the internal binary representation of the integer is the same whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in **Example 3** above the input represents the signed integer `-3`.

Follow up:

If this function is called many times, how would you optimize it?

通过次数 70,532 | 提交次数 103,342


```java
public int hammingWeight(int n) {
    int res = 0;
    int count = 0;
    while(count < 32)
    {
        res += (n & 1) == 1 ? 1 : 0;
        count++;
        n >>= 1;
    }
    return res;
}
```

192 Word Frequency -- Bash

192. Word Frequency

难度 中等 104

Write a bash script to calculate the frequency of each word in a text file `words.txt`.

For simplicity sake, you may assume:

- `words.txt` contains only lowercase characters and space `' '` characters.
- Each word must consist of lowercase characters only.
- Words are separated by one or more whitespace characters.

Example:

Assume that `words.txt` has the following content:

```
the day is sunny the the
the sunny is is
```

Your script should output the following, sorted by descending frequency:

```
the 4
is 3
sunny 2
day 1
```

199 Binary Tree Right Side View

执行结果: 通过 显示详情 >

执行用时: 1 ms , 在所有 Java 提交中击败了 96.12% 的用户

内存消耗: 37 MB , 在所有 Java 提交中击败了 75.45% 的用户

炫耀一下:

```java
class Solution {
    List<Integer> res = new ArrayList<>();
    public List<Integer> rightSideView(TreeNode root) {
```

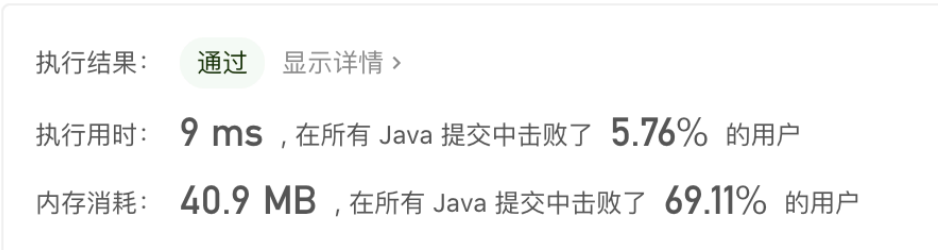
```
        if(root == null)    return res;
        postOrder(root);

        Collections.reverse(res);
        return res;
    }

    public void postOrder(TreeNode root){
        if(root == null)    return;

        postOrder(root.left);

        postOrder(root.right);

        res.add(root.val);
    }
}
```

DFS 方法很巧妙

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

作者：sweetiee
链接：https://leetcode-cn.com/problems/binary-tree-right-side-view/solution/jian-dan-bfsdfs-bi-xu-miao-dong-by-sweetiee/
```

200 Number of Islands

太慢了




```java
class Solution {
    int row;
    int column;
    int[][] dir = {{1, 0},{-1, 0},{0, -1},{0, 1}};
    public int numIslands(char[][] grid) {
        row = grid.length;
        column = row == 0 ? 0 : grid[0].length;

        if(column == 0) return 0;
        HashSet<Integer> set = new HashSet<>();
        WeightedUnionFind wuf = new WeightedUnionFind(row * column);

        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                for(int k  = 0; k < 4; k++){
                    int newX = i + dir[k][0];
                    int newY = j + dir[k][1];
                    if(isInRange(newX, newY) && grid[i][j] == grid[newX][newY]){
                        wuf.union(node(i, j), node(newX, newY));
                    }
                }


```

```
        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(grid[i][j] == '1' && !set.contains(wuf.find(node(i, j)))){
                    set.add(wuf.find(node(i, j)));
                }

        return set.size();

    }

    public boolean isInRange(int i, int j){ return i >= 0 && j >= 0 && i < row && j < column;}

    public int node(int i, int j){
        return i * column + j;
    }
}

class WeightedUnionFind{
    public int[] id;
    private int[] sz;
    private int count;

    public WeightedUnionFind(int N){
        this.count = 0;
        id = new int[N];
        sz = new int[N];
        for(int i = 0; i < N; i++){
            id[i] = i;
            sz[i] = 1;
        }
    }

    public int find(int p){
        while(p != id[p]){
            id[p] = id[id[p]];
            p = id[p];
        }
        return p;
    }

    public boolean connected(int p, int q){return find(p) == find(q);}

    public void union(int p, int q){
        int pRoot = find(p);
        int qRoot = find(q);

        if(pRoot == qRoot)  return;

```

```
        if(sz[pRoot] > sz[qRoot]){
            sz[pRoot] += sz[qRoot];
            id[qRoot] = pRoot;
        }else{
            sz[qRoot] += sz[pRoot];
            id[pRoot] = qRoot;
        }


    }
}
```

28 分钟前　通过　2 ms　40.6 MB　J

```
class Solution {
    int row;
    int column;
    int res = 0;
    public int numIslands(char[][] grid) {
        row = grid.length;
        column = row == 0 ? 0 : grid[0].length;

        if(column == 0)     return 0;

        for(int i = 0; i < row; i++)
            for(int j = 0; j < column; j++)
                if(grid[i][j] == '1'){
                    dfs(grid, i, j);
                    res += 1;
                }

        return res;
    }


    private void dfs(char[][] grid, int i, int j){
        if(isInRange(i, j) && grid[i][j] == '1'){
            grid[i][j] = '-';

            dfs(grid, i + 1, j);
            dfs(grid, i - 1, j);
            dfs(grid, i, j + 1);
            dfs(grid, i, j - 1);
```

```
        }
    }

    public boolean isInRange(int i, int j){
        return i >= 0 && j >= 0 && i < row && j < column;
    }
}
```