面试题03. 数组中重复的数字

2020年5月5日 1:28

找出数组中重复的数字。

在一个长度为 n 的数组 nums 里的所有数字都在 0 ~ n-1 的范围内。数组中某些数字是重复的，但不知道有几个数字重复了，也不知道每个数字重复了几次。请找出数组中任意一个重复的数字。

示例 1：

输入：
[2, 3, 1, 0, 2, 5, 3]
输出：2 或 3

限制：

2 <= n <= 100000

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/shu-zu-zhong-zhong-fu-de-shu-zi-lcof


```
class Solution {
  public int findRepeatNumber(int[] nums) {
    //hash表
    Set<Integer> set = new HashSet<Integer>();
    int res = -1;
    for(int num : nums)
      if(!set.add(num))//add失败 返回flase 找到了某个重复元素
      {
        res = num;
        break;
      }

    return res;
  }
}
```

```
class Solution {
  public void swap(int[] nums, int i, int j)
  {
    int temp = nums[i];
```

```
    nums[i] = nums[j];
    nums[j] = temp;
  }
  public int findRepeatNumber(int[] nums) {
    int n = nums.length;
    //1~ n -1
    for(int num : nums)
      if(num < 0 || num >= n)
        return -1;
    //利用下标交换 保证下标 == 元素值
    for(int i = 0; i < n; i ++)
    {
      while(nums[i] != i && nums[nums[i]] != nums[i])
        swap(nums, i, nums[i]);//交换nums[i]  nums[nums[i]]
      if(nums[i] != i)
        return nums[i];
    }
    return -1;
  }
}
```

面试题04. 二维数组中的查找

2020年5月5日 2:26

在一个 n * m 的二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。

示例:

现有矩阵 matrix 如下：

```
[
  [1,  4,  7, 11, 15],
  [2,  5,  8, 12, 19],
  [3,  6,  9, 16, 22],
  [10, 13, 14, 17, 24],
  [18, 21, 23, 26, 30]
]
```

给定 target = 5，返回 true。

给定 target = 20，返回 false。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/er-wei-shu-zu-zhong-de-cha-zhao-lcof


```
class Solution {
  public boolean findNumberIn2DArray(int[][] array, int target) {
    if((array==null||array.length==0)||(array.length==1&&array[0].length==0))
      return false;

    int i = 0, j = array[0].length - 1;
    while(i <= array.length - 1  && j >= 0)
    {
      if(array[i][j] == target) return true;
      if(array[i][j] > target) j --;
      else i ++;
    }

    return false;
  }
}
```

面试题05. 替换空格

2020年5月5日　　2:39

面试题05. 替换空格

难度简单16

请实现一个函数，把字符串 `s` 中的每个空格替换成"%20"。

示例 1：

输入： s = "We are happy."

输出： "We%20are%20happy."

限制：

0 <= s 的长度 <= 10000

通过次数29,427

提交次数38,337

来自 <https://leetcode-cn.com/problems/ti-huan-kong-ge-lcof/>

```
class Solution {
  public String replaceSpace(String s) {
    int n = s.length();
    char[] array = new char[n * 3];
    int size = 0;
    for(int i = 0; i < n; i ++)
    {
      char c = s.charAt(i);
      if(c == ' ')
      {
        array[size ++] = '%';
        array[size ++] = '2';
        array[size ++] = '0';
      }
      else
        array[size ++] = c;
    }
    String news = new String(array, 0 , size);
    return news;
  }
}
```

面试题06. 从尾到头打印链表

2020年5月5日 2:43

面试题06. 从尾到头打印链表

难度简单22

输入一个链表的头节点，从尾到头反过来返回每个节点的值（用数组返回）。

示例 1：

输入： head = [1,3,2]

输出： [2,3,1]

限制：

0 <= 链表长度 <= 10000

来自 <https://leetcode-cn.com/problems/cong-wei-dao-tou-da-yin-lian-biao-lcof/>

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public int[] reversePrint(ListNode head) {
    //压栈
    Stack<ListNode> stk = new Stack<ListNode>();
    ListNode temp = head;
    while(temp != null)//一直指向最后一个结点
    {
      stk.push(temp);
      temp = temp.next;
    }
    int n = stk.size();
    int[] res = new int[n];
    for(int i = 0; i < n; i ++)
      res[i] = stk.pop().val;

    return res;
  }
}
```

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
```

```
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public int[] reversePrint(ListNode head) {
    //不用栈
    ListNode temp = head;
    int size = 0;
    while(temp != null)
    {
      size ++;
      temp = temp.next;
    }
    int[] res = new int[size];
    temp = head;
    for(int i = size - 1; i >= 0; i --)
    {
      res[i] = temp.val;
      temp = temp.next;
    }
     return res;
  }
}
```

面试题07. 重建二叉树

2020年5月5日　　3:02

面试题07. 重建二叉树

难度中等76

输入某二叉树的前序遍历和中序遍历的结果，请重建该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。

例如，给出

前序遍历 preorder = [3,9,20,15,7]

中序遍历 inorder = [9,3,15,20,7]

返回如下的二叉树：

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

来自 <https://leetcode-cn.com/problems/zhong-jian-er-cha-shu-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  //树 递归
  int preindex = 0;
  int inindex = 0;
  public TreeNode buildTree(int[] preorder, int[] inorder) {
    return dfs(preorder, inorder, null);
  }
  private TreeNode dfs(int[] preorder, int[] inorder, TreeNode finish)
  {
    if(preindex == preorder.length || (finish !
= null &&  inorder[inindex] == finish.val))
      return null;
    //遍历过程
    //前序 跟左右
    TreeNode root = new TreeNode(preorder[preindex ++]);
    //左子树
```

```
        root.left = dfs(preorder, inorder, root);
        inindex ++;
        //右子树
        root.right = dfs(preorder, inorder, finish);
        return root;
    }
}
```

面试题08.二叉树的下一个节点

2020年5月5日 4:50

给定一棵二叉树的其中一个节点，请找出中序遍历序列的下一个节点。

注意：

- 如果给定的节点是中序遍历序列的最后一个，则返回空节点;
- 二叉树一定不为空，且给定的节点一定不是空节点;

样例

假定二叉树是：[2, 1, 3, null, null, null, null]， 给出的是值等于2的节点。
则应返回值等于3的节点。

解释：该二叉树的结构如下，2的后继节点是3。

```
 2
/\
1  3
```

来自 <https://www.acwing.com/problem/content/31/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode father;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public TreeNode inorderSuccessor(TreeNode p) {
    //左根右
    //中序遍历特点
    //一个结点 有右子树  返回一定是 右子树 最左边的结点
    if(p.right != null)
    {
      p = p.right;
      while(p.left != null) p = p.left;
      return p;
    }
    //没有右子树 返回的是 父亲结点
    while(p.father != null)
```

```
    {
      if(p == p.father.left)
        return p.father;
      p = p.father;
    }
    return null;
  }
}
```

面试题09. 用两个栈实现队列

2020年5月5日　　3:27

面试题09. 用两个栈实现队列

难度简单34

用两个栈实现一个队列。队列的声明如下，请实现它的两个函数 appendTail 和 deleteHead ，分别完成在队列尾部插入整数和在队列头部删除整数的功能。(若队列中没有元素，deleteHead 操作返回 -1)

示例 1：

输入：

["CQueue","appendTail","deleteHead","deleteHead"]

[[],[3],[],[]]

输出：[null,null,3,-1]

示例 2：

输入：

["CQueue","deleteHead","appendTail","appendTail","deleteHead","deleteHead"]

[[],[],[5],[2],[],[]]

输出：[null,-1,null,null,5,2]

来自 <https://leetcode-cn.com/problems/yong-liang-ge-zhan-shi-xian-dui-lie-lcof/>

```
class CQueue {
  //栈 先进后出
  //队列 先进先出
  Stack<Integer> stk1, stk2;
  int size;
  public CQueue() {
    stk1 = new Stack<Integer>();
    stk2 = new Stack<Integer>();
    size = 0;
  }

  public void appendTail(int value) {
    //插入一个元素
    //stk1保存 底部存新插入的 顶部存老的
    while(!stk1.isEmpty())
      stk2.push(stk1.pop());

    stk1.push(value);
    while(!stk2.isEmpty())
      stk1.push(stk2.pop());
    size ++;
  }

  public int deleteHead() {
```

```
    //删除队列首部元素
    //删除栈顶
    if(size == 0)
      return -1;

    int res = stk1.pop();
    size --;
    return res;
  }
}
/**
 * Your CQueue object will be instantiated and called as such:
 * CQueue obj = new CQueue();
 * obj.appendTail(value);
 * int param_2 = obj.deleteHead();
 */
```

面试题10- I. 斐波那契数列

2020年5月5日　　3:40

面试题10- I. 斐波那契数列

难度简单20

写一个函数，输入 `n` ，求斐波那契（Fibonacci）数列的第 `n` 项。斐波那契数列的定义如下：

`F(0) = 0,  F(1) = 1`

`F(N) = F(N - 1) + F(N - 2), 其中 N > 1.`

斐波那契数列由 0 和 1 开始，之后的斐波那契数就是由之前的两数相加而得出。

答案需要取模 1e9+7（1000000007），如计算初始结果为：1000000008，请返回 1。

示例 1：

输入： `n = 2`

输出： `1`

示例 2：

输入： `n = 5`

输出： `5`

来自 <https://leetcode-cn.com/problems/fei-bo-na-qi-shu-lie-lcof/>

```
class Solution {
  public int fib(int n) {
    if(n == 0) return 0;
    if(n == 1) return 1;
    int first = 0, second = 1;
    int res = 0;
    for(int i = 2; i <= n; i ++)
    {
      res = (first + second) % 1000000007 ;
      first = second % 1000000007;
      second = res % 1000000007;
    }

    return res % 1000000007;
  }
}
```

面试题10- II. 青蛙跳台阶问题

2020年5月5日　　3:47

面试题10- II. 青蛙跳台阶问题

难度简单23

一只青蛙一次可以跳上1级台阶，也可以跳上2级台阶。求该青蛙跳上一个 n 级的台阶总共有多少种跳法。

答案需要取模 1e9+7（1000000007），如计算初始结果为：1000000008，请返回 1。

示例 1：

输入： n = 2

输出： 2

示例 2：

输入： n = 7

输出： 21

提示：

- 0 <= n <= 100

来自 <https://leetcode-cn.com/problems/qing-wa-tiao-tai-jie-wen-ti-lcof/>

```
class Solution {
  public int numWays(int n) {
    if(n == 0) return 1;
    if(n == 1) return 1;
    int first = 1, second = 1;
    int res = 0;
    for(int i = 2; i <= n; i ++)
    {
      res = (first + second) % 1000000007 ;
      first = second % 1000000007;
      second = res % 1000000007;
    }

    return res % 1000000007;
  }
}
```

面试题11. 旋转数组的最小数字

2020年5月5日 3:49

面试题11. 旋转数组的最小数字

难度简单47

把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。输入一个递增排序的数组的一个旋转，输出旋转数组的最小元素。例如，数组 [3,4,5,1,2] 为 [1,2,3,4,5] 的一个旋转，该数组的最小值为1。

示例 1：

输入：[3,4,5,1,2]

输出：1

示例 2：

输入：[2,2,2,0,1]

输出：0

注意：本题与主站 154 题相同：https://leetcode-cn.com/problems/find-minimum-in-rotated-sorted-array-ii/

来自 <https://leetcode-cn.com/problems/xuan-zhuan-shu-zu-de-zui-xiao-shu-zi-lcof/>

```
class Solution {
  public int minArray(int[] nums) {
    int n = nums.length - 1;
    if(n < 0) return -1;
    while(n > 0 && nums[n] == nums[0]) n --;
    if(nums[n] >= nums[0]) return nums[0];
    int l = 0, r = n;
    while(l < r)
    {
      int mid = l + r >> 1; //[l, mid]  [mid + 1, r]
      if(nums[mid] < nums[0]) r = mid;
      else l = mid + 1;
    }
    return nums[l];
  }
}
```

面试题12. 矩阵中的路径

2020年5月5日 4:08

面试题12. 矩阵中的路径

难度中等60

请设计一个函数，用来判断在一个矩阵中是否存在一条包含某字符串所有字符的路径。路径可以从矩阵中的任意一格开始，每一步可以在矩阵中向左、右、上、下移动一格。如果一条路径经过了矩阵的某一格，那么该路径不能再次进入该格子。例如，在下面的3×4的矩阵中包含一条字符串“bfce”的路径（路径中的字母用加粗标出）。

[["a","**b**","c","e"],
["s","**f**","**c**","s"],
["a","d","**e**","e"]]

但矩阵中不包含字符串“abfb”的路径，因为字符串的第一个字符b占据了矩阵中的第一行第二个格子之后，路径不能再次进入这个格子。

示例 1：

输入： board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"

输出： true

示例 2：

输入： board = [["a","b"],["c","d"]], word = "abcd"

输出： false

来自 <https://leetcode-cn.com/problems/ju-zhen-zhong-de-lu-jing-lcof/>

```
class Solution {
  //回溯法 dfs
  public boolean exist(char[][] board, String word) {
    for(int i = 0; i < board.length; i ++)
      for(int j = 0; j < board[0].length; j ++)
        if(dfs(board, word, 0, i, j))
          return true;

    return false;
  }
  private boolean dfs(char[][] board, String word, int u, int x, int y)
  {
    //先判断边界 后比较相等
    if(x >= board.length || x < 0 || y >= board[0].length || y < 0 || board[x][y] !
= word.charAt(u))
      return false;
    if(u == word.length() - 1) return true;
    char temp = board[x][y];
    board[x][y] = '*';
    //递归遍历
    boolean res = dfs(board, word, u + 1, x - 1, y) || dfs(board, word, u + 1, x +
1, y) || dfs(board, word, u + 1, x, y - 1) || dfs(board, word, u + 1, x, y + 1);
```

```
        board[x][y] = temp;
        return res;
    }
}
```

面试题13. 机器人的运动范围

2020年5月5日　　5:42

面试题13. 机器人的运动范围

难度中等90

地上有一个m行n列的方格，从坐标 [0,0] 到坐标 [m-1,n-1] 。一个机器人从坐标 [0, 0] 的格子开始移动，它每次可以向左、右、上、下移动一格（不能移动到方格外），也不能进入行坐标和列坐标的数位之和大于k的格子。例如，当k为18时，机器人能够进入方格 [35, 37] ，因为3+5+3+7=18。但它不能进入方格 [35, 38]，因为3+5+3+8=19。请问该机器人能够到达多少个格子？

示例 1：

输入： m = 2, n = 3, k = 1

输出： 3

示例 2：

输入： m = 3, n = 1, k = 0

输出： 1

提示：

- 1 <= n,m <= 100
- 0 <= k <= 20

通过次数32,814

提交次数68,287

来自 <https://leetcode-cn.com/problems/ji-qi-ren-de-yun-dong-fan-wei-lcof/>

```
class Solution {
  int m, n, k;
  boolean[][] visited;
  public int movingCount(int m1, int n1, int k1) {
    //找规律
    //19 20   小8
    //39 40   小8
    //x9 x+1 0 小8
    //普通情况 12  13  +1
    m = m1;
    n = n1;
    k = k1;
    visited = new boolean[m][n];//false
    return dfs(0, 0, 0, 0);
  }
  private int dfs(int x, int y, int sx, int sy)
  {
    //sx sy对应 x y数位之和
```

```
    // x 16 y 20 sx 7 sy 2
    //边界优先 m- 1 n -1
    if(x >= m || y >= n || k < sx + sy || visited[x][y]) return 0;
    visited[x][y] = true;
    //sx + sy <= k
    return 1 +  dfs(x + 1, y, (x + 1) % 10 != 0 ? sx + 1 : sx - 8 , sy)
        + dfs(x, y + 1, sx, (y + 1) % 10 != 0 ? sy + 1 : sy - 8);
  }
}
```

面试题14- I. 剪绳子

2020年5月5日 5:55

面试题14- I. 剪绳子

难度中等38

给你一根长度为 n 的绳子，请把绳子剪成整数长度的 m 段（m、n都是整数，n>1并且m>1），每段绳子的长度记为 k[0],k[1]...k[m] 。请问 k[0]*k[1]*...*k[m] 可能的最大乘积是多少？例如，当绳子的长度是8时，我们把它剪成长度分别为2、3、3的三段，此时得到的最大乘积是18。

示例 1：

输入: 2

输出: 1

解释: 2 = 1 + 1, 1 × 1 = 1

示例 2:

输入: 10

输出: 36

解释: 10 = 3 + 3 + 4, 3 × 3 × 4 = 36

提示：

- 2 <= n <= 58

注意：本题与主站 343 题相同：https://leetcode-cn.com/problems/integer-break/

来自 <https://leetcode-cn.com/problems/jian-sheng-zi-lcof/>

```
class Solution {
  public int cuttingRope(int n) {
    if (n < 2) {
      return 0;
    }
    int[] dp = new int[n + 1];
    dp[2] = 1;
    for (int i = 3; i <= n; i++) {
      for (int j = 1; j < i; j++) {
        dp[i] = Math.max(Math.max(j * dp[i - j], j * (i - j)), dp[i]);
      }
    }
    return dp[n];
  }
}
```

面试题15. 二进制中1的个数

2020年5月5日 6:43

面试题15. 二进制中1的个数

难度简单18

请实现一个函数，输入一个整数，输出该数二进制表示中 1 的个数。例如，把 9 表示成二进制是 1001，有 2 位是 1。因此，如果输入 9，则该函数输出 2。

示例 1：

输入： 00000000000000000000000000001011

输出： 3

解释： 输入的二进制串 **00000000000000000000000000001011** 中，共有三位为 '1'。

示例 2：

输入： 00000000000000000000000010000000

输出： 1

解释： 输入的二进制串 **00000000000000000000000010000000** 中，共有一位为 '1'。

示例 3：

输入： 11111111111111111111111111111101

输出： 31

解释： 输入的二进制串 **11111111111111111111111111111101** 中，共有 31 位为 '1'。

注意：本题与主站 191 题相同：https://leetcode-cn.com/problems/number-of-1-bits/

来自 <https://leetcode-cn.com/problems/er-jin-zhi-zhong-1de-ge-shu-lcof/>

```
public class Solution {
  // you need to treat n as an unsigned value
  public int hammingWeight(int n) {
    //n & (n - 1)
    //每执行一次 最右边的1变成0
    //unsigned int
    int count = 0;
    while(n != 0)
    {
      count ++;
      n = n & (n - 1);
    }
    return count;
  }
}
```

面试题16. 数值的整数次方

2020年5月5日　　23:55

面试题16. 数值的整数次方

难度中等22

实现函数double Power(double base, int exponent)，求base的exponent次方。不得使用库函数，同时不需要考虑大数问题。

示例 1:
输入: 2.00000, 10
输出: 1024.00000
示例 2:
输入: 2.10000, 3
输出: 9.26100
示例 3:
输入: 2.00000, -2
输出: 0.25000
解释: 2-2 = 1/22 = 1/4 = 0.25

说明:

- -100.0 < *x* < 100.0
- *n* 是 32 位有符号整数，其数值范围是 [−231, 231 − 1] 。

注意：本题与主站 50 题相同：https://leetcode-cn.com/problems/powx-n/

通过次数13,413

提交次数41,820

来自 <https://leetcode-cn.com/problems/shu-zhi-de-zheng-shu-ci-fang-lcof/>

```
class Solution {
  public double myPow(double x, int n) {
    //快速幂
    //n用二进制表示
    if(x == 0) return 0;
    long b = n;//−2147483648,2147483647
    double res = 1.0;
    if(b < 0)
    {
      b = -b;//用long防止溢出
      x = 1 / x;
    }
    //快速幂
    while(b > 0)
    {
```

```
            if((b & 1) == 1) res *= x;//累乘
            x *= x; //2的次方
            b >>= 1;
        }
        return res;
    }
}
```

面试题17. 打印从1到最大的n位数

2020年5月6日 0:14

面试题17. 打印从1到最大的n位数

难度简单20

输入数字 n，按顺序打印出从 1 到最大的 n 位十进制数。比如输入 3，则打印出 1、2、3 一直到最大的 3 位数 999。

示例 1:

输入: n = 1

输出: [1,2,3,4,5,6,7,8,9]

说明：

- 用返回一个整数列表来代替打印
- n 为正整数

通过次数19,088

提交次数24,206

来自 <https://leetcode-cn.com/problems/da-yin-cong-1dao-zui-da-de-nwei-shu-lcof/>

```
class Solution {
  public int[] printNumbers(int n) {
    if(n <= 0) return new int[0];
    int result = 1;
    int x = 10;
    //该题考查快速幂;
    while(n != 0){
      if((n & 1) == 1){
        result *= x;
      }
      n >>= 1;
      x *= x;
    }
    int len = result - 1;
    int[]array = new int[len];
    for(int i = 0;i < len;i++){
      array[i] = i + 1;
    }
    return array;
  }
}
```

面试题18. 删除链表的节点

2020年5月6日 0:22

面试题18. 删除链表的节点

难度简单17

给定单向链表的头指针和一个要删除的节点的值，定义一个函数删除该节点。

返回删除后的链表的头节点。

注意：此题对比原题有改动

示例 1:

输入: head = [4,5,1,9], val = 5

输出: [4,1,9]

解释: 给定你链表中值为 5 的第二个节点，那么在调用了你的函数之后，该链表应变为 4 -> 1 -> 9.

示例 2:

输入: head = [4,5,1,9], val = 1

输出: [4,5,9]

解释: 给定你链表中值为 1 的第三个节点，那么在调用了你的函数之后，该链表应变为 4 -> 5 -> 9.

说明:

- 题目保证链表中节点的值互不相同
- 若使用 C 或 C++ 语言，你不需要 free 或 delete 被删除的节点

来自 <https://leetcode-cn.com/problems/shan-chu-lian-biao-de-jie-dian-lcof/>

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public ListNode deleteNode(ListNode head, int val) {
    //前一个结点 指向 当前结点的下一个
    //则当前结点被删除
    //删除第一个结点 单独讨论
    //if(head.val == val) return head.next;
    ListNode dummy = new ListNode(0);//虚拟头结点
    dummy.next = head;
    ListNode pre = dummy;
    ListNode cur = dummy.next;
    //找到val结点
    while(cur.val != val && cur != null)
    {
      pre = cur;
```

```
            cur = cur.next;
        }
        //找到val 或者走到头了
        if(cur != null)
        {
            pre.next = cur.next;
        }
        return dummy.next;
    }
}
```

面试题19. 正则表达式匹配

2020年5月6日 0:35

面试题19. 正则表达式匹配

难度困难44

请实现一个函数用来匹配包含'. '和'*'的正则表达式。模式中的字符'.'表示任意一个字符，而'*'表示它前面的字符可以出现任意次（含0次）。在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab*ac*a"匹配，但与"aa.a"和"ab*a"均不匹配。

示例 1:

输入:

s = "aa"

p = "a"

输出: false

解释: "a" 无法匹配 "aa" 整个字符串。

示例 2:

输入:

s = "aa"

p = "a*"

输出: true

解释: 因为 '*' 代表可以匹配零个或多个前面的那一个元素, 在这里前面的元素就是 'a'。因此，字符串 "aa" 可被视为 'a' 重复了一次。

示例 3:

输入:

s = "ab"

p = ".*"

输出: true

解释: ".*" 表示可匹配零个或多个（'*'）任意字符（'.'）。

示例 4:

输入:

s = "aab"

p = "c*a*b"

输出: true

解释: 因为 '*' 表示零个或多个，这里 'c' 为 0 个, 'a' 被重复一次。因此可以匹配字符串 "aab"。

示例 5:

输入:

s = "mississippi"

p = "mis*is*p*."

输出: false

- s 可能为空，且只包含从 a-z 的小写字母。
- p 可能为空，且只包含从 a-z 的小写字母以及字符 . 和 *，无连续的 '*'。

注意：本题与主站 10 题相同：https://leetcode-cn.com/problems/regular-expression-matching/

来自 <https://leetcode-cn.com/problems/zheng-ze-biao-da-shi-pi-pei-lcof/>

```
class Solution {
```

```
  public boolean isMatch(String s, String p) {
    //dp思路
    //f[i][j]  s的前i个字符 和 p 的前j个字符 匹配
    int n = s.length();
    int m = p.length();
    boolean[][] f = new boolean[n + 1][m + 1];
    for(int i = 0; i <= n; i ++)
      for(int j = 0; j <= m ; j ++)
      {
        if(j == 0) // i == j == 0
          f[i][j] = (i == 0);
        else
        {
          //*
          if(p.charAt(j - 1) != '*'){
            //没有碰到*
            if(i > 0 && (s.charAt(i - 1) == p.charAt(j - 1) || p.charAt(j - 1) == '
.'))
              f[i][j] = f[i - 1][j - 1];
          }else{
            //碰到*
            //* = 0
            if(j >= 2)
              f[i][j] |= f[i][j - 2];
            //* != 0
            if(i >= 1 && j >= 2 && (s.charAt(i - 1) == p.charAt(j - 2) || p.charAt(
j - 2) == '.'))
              f[i][j] |= f[i - 1][j];
          }
        }
      }

    return f[n][m];
  }
}
```

面试题20. 表示数值的字符串

2020年5月6日 0:35

面试题20. 表示数值的字符串

难度中等11

请实现一个函数用来判断字符串是否表示数值（包括整数和小数）。例如，字符串"+100"、"5e2"、"-123"、"3.1416"、"0123"都表示数值，
但"12e"、"1a3.14"、"1.2.3"、"+-5"、"-1E-16"及"12e+5.4"都不是。

注意：本题与主站 65 题相同：https://leetcode-cn.com/problems/valid-number/

来自 <https://leetcode-cn.com/problems/biao-shi-shu-zhi-de-zi-fu-chuan-lcof/>

```
class Solution {
  public boolean isNumber(String s) {
    if(s == null || s.length() == 0)
      return false;

    boolean num = false;
    boolean dot = false;
    boolean e = false;
    char[] str = s.trim().toCharArray();//trim必须要 去掉两边空格
    for(int i = 0; i < str.length; i ++)
    {
      if(str[i] >= '0' && str[i] <= '9')
        num = true;
      else if(str[i] == '.')
      {
        // .之前 不能有. e
        if(dot || e)
          return false;

        dot = true;
      }
      else if(str[i] == 'e' || str[i] == 'E')
      {
        // e 之前不能有e  必须有数字 1234e10
        if(e || !num)
          return false;
        e = true;
        //12e 12e+
        num = false;
      }
      else if(str[i] == '+' || str[i] == '-')
```

```
        {
          //+-  第一个位置 或者 e后面的第一个位置
          if(i != 0 && str[i - 1] != 'e' && str[i - 1] != 'E')
            return false;
        }
        else//出现其他特殊字符 buvjyvj
        {
          return false;
        }
      }
      return num;
    }
}
```

面试题21. 调整数组顺序使奇数位于偶数前面

2020年5月6日 0:36

面试题21. 调整数组顺序使奇数位于偶数前面

难度简单18

输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有奇数位于数组的前半部分，所有偶数位于数组的后半部分。

示例：

输入： nums = [1,2,3,4]

输出： [1,3,2,4]

注： [3,1,2,4] 也是正确的答案之一。

提示：

1. 1 <= nums.length <= 50000
2. 1 <= nums[i] <= 10000

来自 <https://leetcode-cn.com/problems/diao-zheng-shu-zu-shun-xu-shi-qi-shu-wei-yu-ou-shu-qian-mian-lcof/>

```
class Solution {
  public int[] exchange(int[] nums) {
    int left = 0;
    int right = nums.length - 1;
    while(left < right)
    {
      //从左往右找偶数 找到偶数 停止
      while(left < right && nums[left] % 2 == 1) left ++;
      //从右往左找奇数 找到奇数停止
      while(left < right && nums[right] % 2 == 0) right --;
      if(left < right)
      {
        int temp = nums[left];
        nums[left] = nums[right];
        nums[right] = temp;
      }
    }
    return nums;
  }
}
```

面试题22. 链表中倒数第k个节点

2020年5月6日 1:00

面试题22. 链表中倒数第k个节点

难度简单31

输入一个链表，输出该链表中倒数第k个节点。为了符合大多数人的习惯，本题从1开始计数，即链表的尾节点是倒数第1个节点。例如，一个链表有6个节点，从头节点开始，它们的值依次是1、2、3、4、5、6。这个链表的倒数第3个节点是值为4的节点。

示例：

给定一个链表: **1->2->3->4->5**, 和 *k* = **2**.

返回链表 4->**5**.

来自 <https://leetcode-cn.com/problems/lian-biao-zhong-dao-shu-di-kge-jie-dian-lcof/>

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public ListNode getKthFromEnd(ListNode head, int k) {
    //单个指针 走到头 len
    //从头走 len - k

    //双指针
    ListNode first = head;
    ListNode second = head;
    for(int i = 0; i < k; i ++)
    {
      //k 大于链表长度
      if(first == null) return null;
      first = first.next;
    }
    while(first != null)
    {
      first = first.next;
      second = second.next;
    }
    return second;
  }
}
```

面试题23.链表中环的入口结点

2020年5月6日 1:04

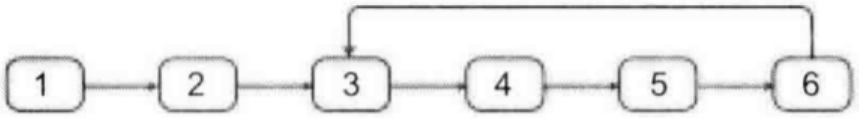
链表中环的入口结点

- 题目
- 提交记录
- 讨论
- 题解
- 视频讲解

给定一个链表，若其中包含环，则输出环的入口节点。

若其中不包含环，则输出null。

样例



给定如上所示的链表：
[1, 2, 3, 4, 5, 6]
3
注意，这里的2表示编号是2的节点，节点编号从0开始。所以编号是2的节点就是val等于3的节点。
则输出环的入口节点3.

来自 <https://www.acwing.com/problem/content/86/>

```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) {
 *         val = x;
 *         next = null;
 *     }
 * }
 */
```

```java
class Solution {
  public ListNode entryNodeOfLoop(ListNode head) {
    /*
    //hash表 从头到尾遍历
    Set<ListNode> hash = new HashSet<>();

    while(head != null)
    {
      if(!hash.add(head))
        return head;
      head = head.next;
    }

    return null;
    */
    //快慢指针
    ListNode slow = head;//走一步
    ListNode fast = head;//走两步
    while(fast != null)
    {
      slow = slow.next;
      fast = fast.next.next;
      if(fast == slow)
      {
        slow = head;
        while(slow != fast)
        {
          slow = slow.next;
          fast = fast.next;
        }
        return slow;
      }

    }

    return null;
  }
}
```

面试题24. 反转链表

2020年5月6日 1:16

面试题24. 反转链表

难度简单35

定义一个函数，输入一个链表的头节点，反转该链表并输出反转后链表的头节点。

示例:

输入: 1->2->3->4->5->NULL

输出: 5->4->3->2->1->NULL

限制:

0 <= 节点个数 <= 5000

来自 <https://leetcode-cn.com/problems/fan-zhuan-lian-biao-lcof/>

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public ListNode reverseList(ListNode head) {
    ListNode pre = null;
    ListNode cur = head;
    while(cur != null)
    {
      //让cur指向 pre
      //然后 pre  cur 都往后走一个
      ListNode temp = cur.next;
      cur.next = pre;
      pre = cur;
      cur = temp;
    }
    return pre;
  }
}
```

```
/**
 * Definition for singly-linked list.
```

```
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public ListNode reverseList(ListNode head) {
    //递归终止条件是当前为空，或者下一个节点为空
    if(head==null || head.next==null) {
      return head;
    }
    //这里的cur就是最后一个节点
    ListNode cur = reverseList(head.next);
    //这里请配合动画演示理解
    //如果链表是 1->2->3->4->5，那么此时的cur就是5
    //而head是4，head的下一个是5，下下一个是空
    //所以head.next.next 就是5->4
    head.next.next = head;
    //防止链表循环，需要将head.next设置为空
    head.next = null;
    //每层递归函数都返回cur，也就是最后一个节点
    return cur;
  }
}
```

面试题25. 合并两个排序的链表

2020年5月6日 1:17

面试题25. 合并两个排序的链表

难度简单24

输入两个递增排序的链表，合并这两个链表并使新链表中的节点仍然是递增排序的。

示例1：

输入： 1->2->4, 1->3->4

输出： 1->1->2->3->4->4

限制：

0 <= 链表长度 <= 1000

注意：本题与主站 21 题相同：https://leetcode-cn.com/problems/merge-two-sorted-lists/

来自 <https://leetcode-cn.com/problems/he-bing-liang-ge-pai-xu-de-lian-biao-lcof/>

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
  public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
    //归并排序思想
    ListNode dummy = new ListNode(0);
    ListNode cur = dummy;
    while(l1 != null && l2 != null)
    {
      if(l1.val <= l2.val)
      {
        cur.next = l1;
        l1 = l1.next;
        cur = cur.next;
      }
      else
      {
        cur.next = l2;
        l2 = l2.next;
        cur = cur.next;
      }
    }
    if(l1 != null)
    {
      cur.next = l1;
```

```
        }
        if(l2 != null)
        {
          cur.next = l2;
        }
        return dummy.next;
    }
}
```

面试题26. 树的子结构

2020年5月6日 1:26

面试题26. 树的子结构

难度中等43

输入两棵二叉树A和B，判断B是不是A的子结构。(约定空树不是任意一个树的子结构)

B是A的子结构， 即 A中有出现和B相同的结构和节点值。

例如:

给定的树 A:

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

给定的树 B：

```
 4
/
1
```

返回 true，因为 B 与 A 的一个子树拥有相同的结构和节点值。

示例 1：

输入： A = [1,2,3], B = [3,1]

输出： false

示例 2：

输入： A = [3,4,5,1,2], B = [4,1]

输出： true

限制：

0 <= 节点个数 <= 10000

来自 <https://leetcode-cn.com/problems/shu-de-zi-jie-gou-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public boolean isSubStructure(TreeNode A, TreeNode B) {
    //递归
    //A根 B根
    //A左子树 B
    //A右子树 B
    if(A == null  || B == null) return false;
```

```
    return (issubtree(A, B) || isSubStructure(A.left, B)|| isSubStructure(A.right, B))
;
  }
  public boolean issubtree(TreeNode A, TreeNode B) {
    //true?
    if(B == null) return true;
    //false?
    if(A == null || A.val != B.val) return false;
    return issubtree(A.left, B.left) && issubtree(A.right, B.right);
  }
}
```

面试题27. 二叉树的镜像

2020年5月6日 1:40

面试题27. 二叉树的镜像

难度简单19

请完成一个函数，输入一个二叉树，该函数输出它的镜像。

例如输入：

```
   4
 /  \
 2   7
/\  /\
1  3 6  9
```

镜像输出：

```
   4
 /  \
 7   2
/\  /\
9  6 3  1
```

示例 1：

输入： root = [4,2,7,1,3,6,9]

输出： [4,7,2,9,6,3,1]

限制：

0 <= 节点个数 <= 1000

注意：本题与主站 226 题相同：https://leetcode-cn.com/problems/invert-binary-tree/

来自 <https://leetcode-cn.com/problems/er-cha-shu-de-jing-xiang-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public TreeNode mirrorTree(TreeNode root) {
    //左右子树交换
    if(root == null) return null;
    TreeNode temp = root.left;
    root.left = mirrorTree(root.right);
    root.right = mirrorTree(temp);
    return root;
```

```
    }
}
```

面试题28. 对称的二叉树

2020年5月6日 1:49

面试题28. 对称的二叉树

难度简单30

请实现一个函数，用来判断一棵二叉树是不是对称的。如果一棵二叉树和它的镜像一样，那么它是对称的。

例如，二叉树 [1,2,2,3,4,4,3] 是对称的。

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

但是下面这个 [1,2,2,null,3,null,3] 则不是镜像对称的:

```
    1
   / \
  2   2
   \   \
    3   3
```

示例 1：

输入： root = [1,2,2,3,4,4,3]

输出： true

示例 2：

输入： root = [1,2,2,null,3,null,3]

输出： false

限制：

0 <= 节点个数 <= 1000

注意：本题与主站 101 题相同：https://leetcode-cn.com/problems/symmetric-tree/

来自 <https://leetcode-cn.com/problems/dui-cheng-de-er-cha-shu-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public boolean isSymmetric(TreeNode root) {
    if(root == null) return true;
```

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

面试题29. 顺时针打印矩阵

2020年5月6日　　1:50

面试题29. 顺时针打印矩阵

难度简单34

输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字。

示例 1：

输入： matrix = [[1,2,3],[4,5,6],[7,8,9]]

输出： [1,2,3,6,9,8,7,4,5]

示例 2：

输入： matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

输出： [1,2,3,4,8,12,11,10,9,5,6,7]

限制：

- 0 <= matrix.length <= 100
- 0 <= matrix[i].length <= 100

注意：本题与主站 54 题相同：https://leetcode-cn.com/problems/spiral-matrix/

通过次数12,807

提交次数29,956

来自 <https://leetcode-cn.com/problems/shun-shi-zhen-da-yin-ju-zhen-lcof/>

```
class Solution {
  public int[] spiralOrder(int[][] matrix) {
    //判断边界
    if(matrix.length == 0)
      return new int[0];
    //右下左上
    int left = 0, right = matrix[0].length -1, up = 0, down = matrix.length - 1;
    int num = 0;
    int[] res = new int[(right + 1) * (down + 1)];
    while(true)
    {
      //右
      for(int i = left; i <= right; i ++)
        res[num ++] = matrix[up][i];
      if(++ up > down) break;
      //下
      for(int i = up; i <= down; i ++)
        res[num ++] = matrix[i][right];
      if(-- right < left ) break;
      //左
```

```
        for(int i = right; i >= left; i --)
          res[num ++] = matrix[down][i];
        if(-- down < up) break;
        //上
        for(int i = down; i >= up; i --)
          res[num ++] = matrix[i][left];

        if(++ left > right) break;
      }
      return res;
    }
}
```

面试题30. 包含min函数的栈

2020年5月10日 5:13

面试题30. 包含min函数的栈

难度简单14

定义栈的数据结构，请在该类型中实现一个能够得到栈的最小元素的 min 函数在该栈中，调用 min、push 及 pop 的时间复杂度都是 O(1)。

示例:

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.min();   --> 返回 -3.
minStack.pop();
minStack.top();      --> 返回 0.
minStack.min();   --> 返回 -2.
```

提示:

1. 各函数的调用总次数不超过 20000 次

注意：本题与主站 155 题相同：https://leetcode-cn.com/problems/min-stack/

通过次数12,755

提交次数21,928

来自 <https://leetcode-cn.com/problems/bao-han-minhan-shu-de-zhan-lcof/>

```
class MinStack {
  Stack<Integer> A, B;//A正常的栈  B记录最小元素的栈
  /** initialize your data structure here. */
  public MinStack() {
    A = new Stack<>();
    B = new Stack<>();
  }

  public void push(int x) {
    //判断当前元素 和栈内最小的元素 对比
    A.add(x);
    if(B.empty() || B.peek() >= x)
      B.add(x);
  }

  public void pop() {
```

```
    //栈顶元素 是不是 最小的
    //if(A.pop().equals(B.peek()))
    //    B.pop();
    if(A.peek().equals(B.peek()))//不能用 == 必须用 equals
      B.pop();

    A.pop();
  }

  public int top() {
    //直接写
    return A.peek();
  }

  public int min() {
    //找到栈内最小的元素 直接写
    return B.peek();
  }
}
/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack obj = new MinStack();
 * obj.push(x);
 * obj.pop();
 * int param_3 = obj.top();
 * int param_4 = obj.min();
 */
```

面试题31. 栈的压入、弹出序列

2020年5月10日 5:24

面试题31. 栈的压入、弹出序列

难度中等27

输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如，序列 {1,2,3,4,5} 是某栈的压栈序列，序列 {4,5,3,2,1} 是该压栈序列对应的一个弹出序列，但 {4,3,5,1,2} 就不可能是该压栈序列的弹出序列。

示例 1：
输入： pushed = [1,2,3,4,5], popped = [4,5,3,2,1]
输出： true
解释： 我们可以按以下顺序执行：
push(1), push(2), push(3), push(4), pop() -> 4,
push(5), pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1
示例 2：
输入： pushed = [1,2,3,4,5], popped = [4,3,5,1,2]
输出： false
解释： 1 不能在 2 之前弹出。

来自 <https://leetcode-cn.com/problems/zhan-de-ya-ru-dan-chu-xu-lie-lcof/>

```
class Solution {
  public boolean validateStackSequences(int[] pushed, int[] popped) {
    Stack<Integer> temp = new Stack<>();
    int i = 0;
    for(int num : pushed)
    {
      temp.push(num);//入栈
      //模拟出栈
      while(!temp.isEmpty() && temp.peek() == popped[i])
      {
        temp.pop();
        i ++;
      }
    }
    return temp.isEmpty();
  }
}
```

Java Stack中add与push，peek与pop

add & push

共同点：

add，push都可以向stack中添加元素。

不同点：

add是继承自Vector的方法，且返回值类型是boolean。

push是Stack自身的方法，返回值类型是参数类类型。

具体的看源码：

```
public synchronized boolean add(E e) {
        modCount++;
        ensureCapacityHelper(elementCount + 1);
        elementData[elementCount++] = e;
        return true;
    }
public E push(E item) {
        addElement(item);

        return item;
    }
```

peek & pop

共同点：

peek，pop都是返回栈顶元素。

不同点：

peek()函数返回栈顶的元素，但不弹出该栈顶元素。
pop()函数返回栈顶的元素，并且将该栈顶元素出栈。

————————————————

面试题32 - I. 从上到下打印二叉树

2020年5月10日 5:38

面试题32 - I. 从上到下打印二叉树

难度中等12

从上到下打印出二叉树的每个节点，同一层的节点按照从左到右的顺序打印。

例如:

给定二叉树: [3,9,20,null,null,15,7],

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

返回:

[3,9,20,15,7]

来自 <https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public int[] levelOrder(TreeNode root) {
    if(root == null) return new int[0];

    ArrayList<Integer> ans = new ArrayList<>();
    //队列操作 保存根节点
    Queue<TreeNode> q = new LinkedList<>();
    q.add(root);
    while(!q.isEmpty())
    {
      //根结点
      TreeNode r = q.poll();
      ans.add(r.val);
      //左子树
      if(r.left != null) q.add(r.left);
      //右子树
      if(r.right != null) q.add(r.right);
    }
```

```
    int[] res = new int[ans.size()];
    for(int i = 0; i < ans.size(); i ++)
      res[i] = ans.get(i);
    return res;
  }
}
```

2020年5月10日 5:50

面试题32 - II. 从上到下打印二叉树 II

难度简单19

从上到下按层打印二叉树，同一层的节点按从左到右的顺序打印，每一层打印到一行。

例如:

给定二叉树: [3,9,20,null,null,15,7],

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

返回其层次遍历结果：

```
[
 [3],
 [9,20],
 [15,7]
]
```

提示：

1. 节点总数 <= 1000

注意：本题与主站 102 题相同：https://leetcode-cn.com/problems/binary-tree-level-order-traversal/

通过次数14,158

提交次数20,570

来自 <https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-ii-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public List<List<Integer>> levelOrder(TreeNode root) {
    List<List<Integer>> res = new ArrayList<>();
```

```
    Queue<TreeNode> q = new LinkedList<>();
    if(root != null) q.add(root);
    while(!q.isEmpty())
    {
      List<Integer> temp = new ArrayList<>();
      int s = q.size();
      for(int i = 0; i < s; i ++)
      {
        TreeNode node = q.poll();
        temp.add(node.val);
        if(node.left != null) q.add(node.left);
        if(node.right != null) q.add(node.right);
      }
      res.add(temp);
    }

    return res;
  }
}
```

面试题32 - III. 从上到下打印二叉树 III

2020年5月10日 6:03

面试题32 - III. 从上到下打印二叉树 III

难度中等16

请实现一个函数按照之字形顺序打印二叉树，即第一行按照从左到右的顺序打印，第二层按照从右到左的顺序打印，第三行再按照从左到右的顺序打印，其他行以此类推。

例如:
给定二叉树: [3,9,20,null,null,15,7],

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

返回其层次遍历结果：

```
[
 [3],
 [20,9],
 [15,7]
]
```

提示：

1. 节点总数 <= 1000

通过次数11,771

提交次数19,813

在真实的面试中遇到过这道题?

是

否

来自 <https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-iii-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
```

```java
class Solution {
  public List<List<Integer>> levelOrder(TreeNode root) {
    List<List<Integer>> res = new ArrayList<>();
    Queue<TreeNode> q = new LinkedList<>();
    if(root != null) q.add(root);
    while(!q.isEmpty())
    {
      //zigzag
      LinkedList<Integer> temp = new LinkedList<>();
      int s = q.size();
      for(int i = 0; i < s; i ++)
      {
        TreeNode node = q.poll();
        if(res.size() % 2 == 0)
          temp.addLast(node.val);
        else
          temp.addFirst(node.val);
        if(node.left != null) q.add(node.left);
        if(node.right != null) q.add(node.right);
      }
      res.add(temp);
    }
    return res;
  }
}
```

面试题33. 二叉搜索树的后序遍历序列

2020年5月10日 6:13

面试题33. 二叉搜索树的后序遍历序列

难度中等42

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历结果。如果是则返回 true，否则返回 false。假设输入的数组的任意两个数字都互不相同。

参考以下这颗二叉搜索树：

```
     5
    / \
   2   6
  / \
 1   3
```

示例 1：

输入: [1,6,3,2,5]

输出: false

示例 2：

输入: [1,3,2,6,5]

输出: true

提示：

1. 数组长度 <= 1000

来自 <https://leetcode-cn.com/problems/er-cha-sou-suo-shu-de-hou-xu-bian-li-xu-lie-lcof/>

```
class Solution {
  public boolean verifyPostorder(int[] postorder) {
    //左<根<右 二叉检索树
    //后序遍历
    //左右根
    //递归
    return dfs(postorder, 0, postorder.length - 1);
  }
  public boolean dfs(int[] arr, int l, int r)
  {
    if(l >= r) return true;//遍历到最后 都是true
    //r对应 当前子树 根节点
    //找到左右子树的临界点
    int p = l;
    while(arr[p] < arr[r]) p ++;
    int m = p;
    while(arr[p] > arr[r]) p ++;
    return (p == r) && dfs(arr, l, m - 1) && dfs(arr, m, r - 1);
  }
}
```

面试题34. 二叉树中和为某一值的路径

2020年5月10日 19:36

面试题34. 二叉树中和为某一值的路径

难度中等33

输入一棵二叉树和一个整数，打印出二叉树中节点值的和为输入整数的所有路径。从树的根节点开始往下一直到叶节点所经过的节点形成一条路径。

示例:

给定如下二叉树，以及目标和 `sum = 22`，

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

返回:

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

提示:

1. 节点总数 <= 10000

注意：本题与主站 113 题相同：https://leetcode-cn.com/problems/path-sum-ii/

通过次数14,018

提交次数25,238

来自 <https://leetcode-cn.com/problems/er-cha-shu-zhong-he-wei-mou-yi-zhi-de-lu-jing-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
   //前序遍历 根左右
```

```
    LinkedList<List<Integer>>  res = new LinkedList<>();//所有路径
    LinkedList<Integer> path = new LinkedList<>();//单条路径
    public List<List<Integer>> pathSum(TreeNode root, int sum) {
      //递归函数
      recur(root, sum);
      return res;
    }
    void recur(TreeNode root, int target){
      //找到 target = 0时候的叶节点
      if(root == null) return;
      path.add(root.val);
      target -= root.val;
      if(target == 0 && root.left == null && root.right == null)
        res.add(new LinkedList(path));
      recur(root.left, target);
      recur(root.right, target);
      //每次结束后 path必须清空
      path.removeLast();
    }
}
```

面试题35. 复杂链表的复制

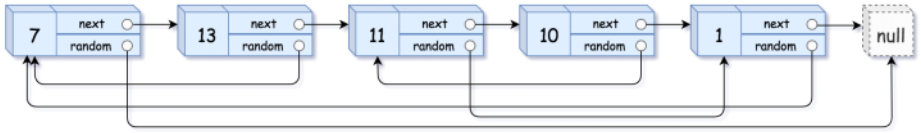
2020年5月23日　　12:06

面试题35. 复杂链表的复制

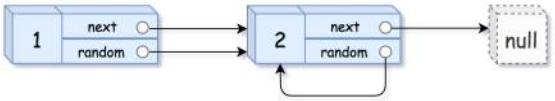
难度中等38

请实现 copyRandomList 函数，复制一个复杂链表。在复杂链表中，每个节点除了有一个 next 指针指向下一个节点，还有一个 random 指针指向链表中的任意节点或者 null。

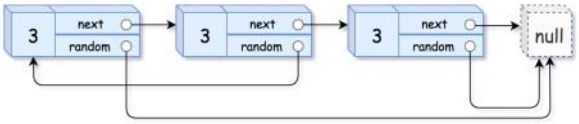
示例 1：



输入： head = [[7,null],[13,0],[11,4],[10,2],[1,0]]
输出： [[7,null],[13,0],[11,4],[10,2],[1,0]]
示例 2：



输入： head = [[1,1],[2,1]]
输出： [[1,1],[2,1]]
示例 3：



输入： head = [[3,null],[3,0],[3,null]]
输出： [[3,null],[3,0],[3,null]]
示例 4：
输入： head = []
输出： []
解释： 给定的链表为空（空指针），因此返回 null。

提示：

- -10000 <= Node.val <= 10000
- Node.random 为空（null）或指向链表中的节点。
- 节点数目不超过 1000 。

来自 <https://leetcode-cn.com/problems/fu-za-lian-biao-de-fu-zhi-lcof/>

```
/*
// Definition for a Node.
class Node {
  int val;
  Node next;
```

```
  Node random;
  public Node(int val) {
    this.val = val;
    this.next = null;
    this.random = null;
  }
}
*/
/*
思路:
1.复制 结点 7  8 9 10
      7 7’ 8 8‘ 9 9‘ 10 10’
2.复制 random指针
      7 random 9
      7 ‘random 9 next 9’
3.分离出来
      7 ‘8’ 9 ‘10’
*/
class Solution {
  public Node copyRandomList(Node head) {
    if(head == null)
      return null;

    copy(head);//复制结点
    copyrandom(head);//复制random指针
    return divide(head);//分离出来
  }
  private void copy(Node head){
    while(head != null)
    {
      Node clone = new Node(head.val);
      Node nextNode = head.next;
      head.next = clone;
      clone.next = nextNode;
      head = clone.next;
    }
  }
  private void copyrandom(Node head){
    while(head != null){
      Node clone = head.next;
      if(head.random != null)
      {
        Node randomnode = head.random;
        clone.random = randomnode.next;
      }
      head = clone.next;
    }
```

```
    }
//7  8 9 10
//7 7’ 8 8‘ 9 9‘ 10 10’
    private Node divide(Node head)
    {
        Node cloneNode = head.next;
        Node cloneHead = cloneNode;
        head.next = cloneNode.next;
        head = head.next;
        while(head != null)
        {
            cloneNode.next = head.next; // 8'
            head.next = head.next.next;//8 9
            head = head.next;
            cloneNode = cloneNode.next;
        }
        return cloneHead;
    }


}
```

面试题36. 二叉搜索树与双向链表

2020年5月24日 0:41

面试题36. 二叉搜索树与双向链表

难度中等46

输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的循环双向链表。要求不能创建任何新的节点，只能调整树中节点指针的指向。

为了让您更好地理解问题，以下面的二叉搜索树为例：

我们希望将这个二叉搜索树转化为双向循环链表。链表中的每个节点都有一个前驱和后继指针。对于双向循环链表，第一个节点的前驱是最后一个节点，最后一个节点的后继是第一个节点。
下图展示了上面的二叉搜索树转化成的链表。“head” 表示指向链表中有最小元素的节点。

特别地，我们希望可以就地完成转换操作。当转化完成以后，树中节点的左指针需要指向前驱，树中节点的右指针需要指向后继。还需要返回链表中的第一个节点的指针。

来自 <https://leetcode-cn.com

```
/*
// Definition for a Node.
class Node {
  public int val;
  public Node left;
  public Node right;
  public Node() {}
  public Node(int _val) {
    val = _val;
  }
  public Node(int _val,Node _left,Node _right) {
    val = _val;
    left = _left;
    right = _right;
  }
};
*/

class Solution {
  Node pre,head;
  public Node treeToDoublyList(Node root) {
    if(root == null) return null;
    dfs(root);//左根右 递增
    //头尾结点处理一下
    head.left = pre;
    pre.right = head;
```

```
    return head;
  }
  private void dfs(Node cur){
    if(cur == null) return;
    //左子树
    dfs(cur.left);
    //根结点
    //left right指针
    if(pre != null) pre.right = cur;
    else head = cur;
    cur.left = pre;
    pre = cur;
    //右子树
    dfs(cur.right);
  }
}
```
/problems/er-cha-sou-suo-shu-yu-shuang-xiang-lian-biao-lcof/>

面试题37. 序列化二叉树

2020年5月24日 1:06

面试题37. 序列化二叉树

难度困难29

请实现两个函数，分别用来序列化和反序列化二叉树。

示例:

你可以将以下二叉树：

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

序列化为 "[1,2,3,null,null,4,5]"

来自 <https://leetcode-cn.com/problems/xu-lie-hua-er-cha-shu-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Codec {
  // Encodes a tree to a single string.
  public String serialize(TreeNode root) {
    if(root == null) return "[]";
    //序列化为 "[1,2,3,null,null,4,5]"
    StringBuilder res = new StringBuilder("[");
    Queue<TreeNode> q = new LinkedList<>();
    q.add(root);
    while(!q.isEmpty())
    {
      TreeNode t = q.poll();
      if(t != null)
      {
        res.append(t.val + ",");
        q.add(t.left);
        q.add(t.right);
      }
      else
        res.append("null,");
    }
```

```
        res.append("]");
        return res.toString();
    }
    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        if(data.equals("[]")) return null;
        String[] vals = data.substring(1,data.length()-1).split(",");
        TreeNode root = new TreeNode(Integer.parseInt(vals[0]));
        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);
        int i = 1;
        while(!q.isEmpty())
        {
            TreeNode node = q.poll();
            if(!vals[i].equals("null")){
                node.left = new TreeNode(Integer.parseInt(vals[i]));
                q.add(node.left);
            }
            i ++;
            if(!vals[i].equals("null")){
                node.right = new TreeNode(Integer.parseInt(vals[i]));
                q.add(node.right);
            }
            i ++;
        }
        return root;
    }
}
// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.deserialize(codec.serialize(root));
```

面试题38. 字符串的排列

2020年5月24日 1:27

面试题38. 字符串的排列

难度中等41

输入一个字符串，打印出该字符串中字符的所有排列。

你可以以任意顺序返回这个字符串数组，但里面不能有重复元素。

示例:

输入： s = "abc"

输出： ["abc","acb","bac","bca","cab","cba"]

限制：

1 <= s 的长度 <= 8

来自 <https://leetcode-cn.com/problems/zi-fu-chuan-de-pai-lie-lcof/>

```
class Solution {
  Set<String> res = new HashSet<>();//保存结果
  public String[] permutation(String s) {
    if(s == null) return new String[]{};
    boolean[] visited = new boolean[s.length()];
    dfs(s,"",visited);
    return res.toArray(new String[res.size()]);
  }
  private void dfs(String s, String ch, boolean[] visited){
    if(s.length() == ch.length())
    {
      res.add(ch);//abc
      return;
    }
    for(int i = 0; i < s.length(); i ++)
    {
      char temp = s.charAt(i);
      if(visited[i]) continue;
      visited[i] = true;
      dfs(s, ch + String.valueOf(temp), visited);
      visited[i] = false;
    }
  }
}
```

面试题39. 数组中出现次数超过一半的数字

2020年5月24日 1:54

面试题39. 数组中出现次数超过一半的数字

难度简单29

数组中有一个数字出现的次数超过数组长度的一半，请找出这个数字。

你可以假设数组是非空的，并且给定的数组总是存在多数元素。

示例 1:

输入: [1, 2, 3, 2, 2, 2, 5, 4, 2]

输出: 2

限制:

1 <= 数组长度 <= 50000

来自 <https://leetcode-cn.com/problems/shu-zu-zhong-chu-xian-ci-shu-chao-guo-yi-ban-de-shu-zi-lcof/>

```
class Solution {
  public int majorityElement(int[] nums) {
    int val = -1, cnt = 0;
    for(int num : nums)
    {
      //记录相同数字出现次数
      //当前数字 == val cnt ++
      //cnt == 0 val更新当前的
      //最后肯定有cnt >=1 val
      if(num == val)
        cnt ++;
      else{
        //2 2 1 1 1
        if(cnt >= 1) cnt --;
        else{
          val = num;
          cnt = 1;
        }
      }
    }
    return val;
  }
}
```

面试题40. 最小的k个数

2020年5月24日 2:07

面试题40. 最小的k个数

难度简单83

输入整数数组 `arr` ，找出其中最小的 `k` 个数。例如，输入4、5、1、6、2、7、3、8这8个数字，则最小的4个数字是1、2、3、4。

示例 1：

输入： `arr = [3,2,1], k = 2`

输出： `[1,2] 或者 [2,1]`

示例 2：

输入： `arr = [0,1,2,1], k = 1`

输出： `[0]`

来自 <https://leetcode-cn.com/problems/zui-xiao-de-kge-shu-lcof/>

```
class Solution {
  //快排
  public int[] getLeastNumbers(int[] arr, int k) {
    if(k == 0 || arr.length == 0)
      return new int[0];
    return qsk(arr,0, arr.length - 1, k - 1);
  }
  private int[] qsk(int[] nums, int l, int r, int k){
    int j = quicksort(nums, l, r);
    //j k对比得到要不要返回前k小的数组
    if(j == k)
    {
      return Arrays.copyOf(nums,j + 1);
    }
    //否则 要根据 j  k大小关系进行调整
    return j > k ? qsk(nums,l, j - 1, k) : qsk(nums, j + 1, r, k);
  }



  private int quicksort(int[] nums, int l, int r){
    int v = nums[l];
    int i = l, j = r + 1;
    while(true)
    {
      while(++i <= r && nums[i] < v);//i
      while(--j >= l && nums[j] > v);//j
```

```
            if(i >= j)
                break;
            int temp = nums[i];
            nums[i] = nums[j];
            nums[j] = temp;
        }
        nums[l] = nums[j];
        nums[j] = v;
        return j;//保证 nums[j] 左边是最小的j个数字
    }
}
```

面试题41. 数据流中的中位数

2020年5月25日 15:11

面试题41. 数据流中的中位数

难度困难26

如何得到一个数据流中的中位数？如果从数据流中读出奇数个数值，那么中位数就是所有数值排序之后位于中间的数值。如果从数据流中读出偶数个数值，那么中位数就是所有数值排序之后中间两个数的平均值。

例如，

[2,3,4] 的中位数是 3

[2,3] 的中位数是 (2 + 3) / 2 = 2.5

设计一个支持以下两种操作的数据结构：

- void addNum(int num) - 从数据流中添加一个整数到数据结构中。
- double findMedian() - 返回目前所有元素的中位数。

示例 1：

输入：

["MedianFinder","addNum","addNum","findMedian","addNum","findMedian"]

[[],[1],[2],[],[3],[]]

输出：[null,null,null,1.50000,null,2.00000]

示例 2：

输入：

["MedianFinder","addNum","findMedian","addNum","findMedian"]

[[],[2],[],[3],[]]

输出：[null,null,2.00000,null,2.50000]

限制：

- 最多会对 addNum、findMedia进行 50000 次调用。

来自 <https://leetcode-cn.com/problems/shu-ju-liu-zhong-de-zhong-wei-shu-lcof/>

```
class MedianFinder {


  //大根堆 + 小根堆
  //大根堆 堆顶 存的是最大的数字  小的那一半

  //小根堆 堆顶 存的是最小的数字  大的那一半


  Queue<Integer> A, B;
  /** initialize your data structure here. */
  public MedianFinder() {
     A = new PriorityQueue<>();//小根堆 存大的一半数字
     B = new PriorityQueue<>((x,y) -> (y - x));//大根堆 存小的一半数字
  }
```

```
    public void addNum(int num) {
        //无脑放进小根堆
        //如果 小根堆的数字数量 > 大根堆的数字数量了
        //把小根堆中较小数字 放进大根堆
        if(A.size() != B.size())
        {
            A.add(num);//无脑放进小根堆
            B.add(A.poll());//把小根堆中较小数字 放进大根堆
        }else
        {
            //堆顶比较 确保大小根堆的 一半的特性
            B.add(num);//大数字 在堆顶
            A.add(B.poll());//通过添加堆顶数字 确保 特性
        }
    }

    public double findMedian() {
        return A.size() == B.size() ? (A.peek() + B.peek())/2.0 : A.peek();
    }
}

/**
 * Your MedianFinder object will be instantiated and called as such:
 * MedianFinder obj = new MedianFinder();
 * obj.addNum(num);
 * double param_2 = obj.findMedian();
 */
```

面试题42. 连续子数组的最大和

2020年6月17日 1:54

面试题42. 连续子数组的最大和

难度简单59

输入一个整型数组，数组里有正数也有负数。数组中的一个或连续多个整数组成一个子数组。求所有子数组的和的最大值。

要求时间复杂度为O(n)。

示例1:

输入: nums = [-2,1,-3,4,-1,2,1,-5,4]

输出: 6

解释: 连续子数组 [4,-1,2,1] 的和最大，为 6。

来自 <https://leetcode-cn.com/problems/lian-xu-zi-shu-zu-de-zui-da-he-lcof/>

```
class Solution {
  public int maxSubArray(int[] nums) {
    int res = nums[0];//最大子数组的和

    for(int i = 1; i < nums.length; i ++){
      //动态规划
      //dp[i]到i位置 最大子数组的和
      //dp[i - 1]  nums[i]
      nums[i] += Math.max(nums[i - 1], 0);
      res = Math.max(res, nums[i]);
    }

    return res;
  }
}
```

面试题43. 1～n整数中1出现的次数

2020年6月17日 2:10

面试题43. 1～n整数中1出现的次数

难度中等41

输入一个整数 n ，求1～n这n个整数的十进制表示中1出现的次数。

例如，输入12，1～12这些整数中包含1 的数字有1、10、11和12，1一共出现了5次。

示例 1：

输入： n = 12

输出： 5

示例 2：

输入： n = 13

输出： 6

限制：

- 1 <= n < 2^31

注意：本题与主站 233 题相同：https://leetcode-cn.com/problems/number-of-digit-one/

来自 <https://leetcode-cn.com/problems/1nzheng-shu-zhong-1chu-xian-de-ci-shu-lcof/>

```
class Solution {
  public int countDigitOne(int n) {
    //high cur low
    //4507
    // cur == 0  high * 10^i
    //high 45 cur 0 low 7
    // 4507
    // 0010 - 4419
    // 000 - 449  450个数字
    //cur == 1 high* 10^i  + low + 1
    //4517
    //high 45  cur 1 low 7
    //0010 4517
    //000 - 457   458个数字
    //cur == 2 3 4...9   (high+1) * 10^i
    //4527
    //high 45  cur 2 low 7
    // 0010 - 4519
    // 000 - 459  460个数字
    int x = 1, res = 0;
    int high = n / 10, cur = n % 10, low = 0;
    while(high != 0 || cur != 0){
      if(cur == 0) res += high * x;
      else if(cur == 1) res += high * x + low + 1;
```

```
            else res += (high + 1) * x;
            low += cur * x;
            cur = high % 10;
            high /= 10;
            x *= 10;
        }
        return res;
    }
}
```

面试题44. 数字序列中某一位的数字

2020年6月17日 2:42

面试题44. 数字序列中某一位的数字

难度中等32

数字以0123456789101112131415...的格式序列化到一个字符序列中。在这个序列中，第5位（从下标0开始计数）是5，第13位是1，第19位是4，等等。

请写一个函数，求任意第n位对应的数字。

示例 1：

输入： n = 3

输出： 3

示例 2：

输入： n = 11

输出： 0

来自 <https://leetcode-cn.com/problems/shu-zi-xu-lie-zhong-mou-yi-wei-de-shu-zi-lcof/>

```
class Solution {
  public int findNthDigit(int n) {
    //第一步 确定n在哪个 数位的数中 1  2  3
    //1 9
    //10 99
    //100 999
    int digit = 1;
    long start = 1;
    long count = 9;
    while(n > count){//得到了digit
      n -= count;
      digit += 1;
      start *= 10;
      count = digit * start * 9;//对于 2 个数字组成的10- 99  180  2*90
    }
    //第二步：确定在哪个数字中 15
    long num = start + (n - 1)/digit;
    //第三步：找到确定的1位的数字
    return Long.toString(num).charAt((n - 1) % digit) - '0';
  }
}
```

面试题45. 把数组排成最小的数

2020年6月17日 3:10

面试题45. 把数组排成最小的数

难度中等51

输入一个非负整数数组，把数组里所有数字拼接起来排成一个数，打印能拼接出的所有数字中最小的一个。

示例 1:

输入: [10,2]

输出: "102"

示例 2:

输入: [3,30,34,5,9]

输出: "3033459"

来自 <https://leetcode-cn.com/problems/ba-shu-zu-pai-cheng-zui-xiao-de-shu-lcof/>

```
class Solution {
  public String minNumber(int[] nums) {
    String[] str = new String[nums.length];
    for(int i = 0; i < nums.length; i ++){
      str[i] = String.valueOf(nums[i]);
    }
    //字符串排序
    strsort(str, 0, str.length - 1);//快排
    //Arrays.sort(str,(x,y) -> (x + y).compareTo(y + x));//内置函数
    //字符串数组依次组合
    StringBuilder res = new StringBuilder();
    for(String s : str){
      res.append(s);
    }
    return res.toString();
  }
  void strsort(String[] str, int l, int r){
    //快排写法
    if(l >= r) return;
    int i = l, j = r;
    String temp = str[i];
    while(i < j){
      while((str[j] + str[l]).compareTo(str[l] + str[j]) >= 0 && i < j) j --;
      while((str[i] + str[l]).compareTo(str[l] + str[i]) <= 0 && i < j) i ++;
      temp = str[i];
      str[i] = str[j];
      str[j] = temp;
    }
    str[i] = str[l];
```

```
        str[l] = temp;
        strsort(str, l, i - 1);
        strsort(str, i + 1, r);
    }
}
```

面试题46. 把数字翻译成字符串

2020年6月17日 3:33

面试题46. 把数字翻译成字符串

难度中等95

给定一个数字，我们按照如下规则把它翻译为字符串：0 翻译成 “a” ，1 翻译成 “b”，……，11 翻译成 “l”，……，25 翻译成 “z”。一个数字可能有多个翻译。请编程实现一个函数，用来计算一个数字有多少种不同的翻译方法。

示例 1:

输入: 12258

输出: 5

解释: 12258有5种不同的翻译，分别是"bccfi", "bwfi", "bczi", "mcfi"和"mzi"

来自 <https://leetcode-cn.com/problems/ba-shu-zi-fan-yi-cheng-zi-fu-chuan-lcof/>

```
class Solution {
  public int translateNum(int num) {
    // 0  ~ 25有意义
    //dp【i】 = dp【i - 1】  (+ dp【i - 2】)
    String s = String.valueOf(num);
    int a = 1, b = 1;
    for(int i = 2; i <= s.length(); i ++){
      String temp = s.substring(i - 2, i);
      int c = temp.compareTo("10") >= 0 && temp.compareTo("25") <= 0 ? a + b : a;
      b = a;
      a = c;
    }
    return a;
  }
}
```

面试题47. 礼物的最大价值

2020年6月17日 3:48

面试题47. 礼物的最大价值

难度中等42

在一个 m*n 的棋盘的每一格都放有一个礼物，每个礼物都有一定的价值（价值大于 0）。你可以从棋盘的左上角开始拿格子里的礼物，并每次向右或者向下移动一格、直到到达棋盘的右下角。给定一个棋盘及其上面的礼物的价值，请计算你最多能拿到多少价值的礼物？

示例 1:

输入:

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

输出: 12

解释: 路径 1→3→5→2→1 可以拿到最多价值的礼物

来自 <https://leetcode-cn.com/problems/li-wu-de-zui-da-jie-zhi-lcof/>

```
class Solution {
  public int maxValue(int[][] f) {
    //最经典二维动态规划
    //f[i][j] = max(f[i-1][j], f[i][j - 1])
    int m = f.length, n = f[0].length;
    for(int i = 0; i < m; i ++)
      for(int j = 0; j < n; j ++){
        if(i == 0 && j == 0) continue;
        if(i == 0) f[i][j] += f[i][j - 1];
        else if(j == 0) f[i][j] += f[i - 1][j];
        else f[i][j] += Math.max(f[i - 1][j], f[i][j - 1]);
      }

    return f[m - 1][n - 1];
  }
}
```

面试题48. 最长不含重复字符的子字符串

2020年6月17日 3:55

面试题48. 最长不含重复字符的子字符串

难度中等50

请从字符串中找出一个最长的不包含重复字符的子字符串，计算该最长子字符串的长度。

示例 1:

输入: "abcabcbb"

输出: 3

解释: 因为无重复字符的最长子串是 "abc"，所以其长度为 3。

示例 2:

输入: "bbbbb"

输出: 1

解释: 因为无重复字符的最长子串是 "b"，所以其长度为 1。

示例 3:

输入: "pwwkew"

输出: 3

解释: 因为无重复字符的最长子串是 "wke"，所以其长度为 3。

请注意，你的答案必须是 **子串** 的长度，"pwke" 是一个*子序列*，不是子串。

提示：

- s.length <= 40000

注意：本题与主站 3 题相同：https://leetcode-cn.com/problems/longest-substring-without-repeating-characters/

来自 <https://leetcode-cn.com/problems/zui-chang-bu-han-zhong-fu-zi-fu-de-zi-zi-fu-chuan-lcof/>

```
class Solution {
  public int lengthOfLongestSubstring(String s) {
    HashMap<Character, Integer> hash = new HashMap<>();
    int res = 0, left = 0;
    for(int i = 0; i < s.length(); i ++){
      char c = s.charAt(i);
      //判断c是否出现过
      //"abcabcbb"
      if(hash.containsKey(c)){
        left = Math.max(left, hash.get(c) + 1);
      }
      hash.put(c, i);
      res = Math.max(res, i - left + 1);
    }
    return res;
  }
}
```

面试题49. 丑数

2020年6月17日 5:03

面试题49. 丑数

难度中等36

我们把只包含因子 2、3 和 5 的数称作丑数（Ugly Number）。求按从小到大的顺序的第 n 个丑数。

示例:

输入: n = 10

输出: 12

解释: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12 是前 10 个丑数。

说明:

1. 1 是丑数。
2. n **不超过**1690。

注意：本题与主站 264 题相同：https://leetcode-cn.com/problems/ugly-number-ii/

通过次数13,406

提交次数21,203

来自 <https://leetcode-cn.com/problems/chou-shu-lcof/>

```
class Solution {
  public int nthUglyNumber(int n) {
    int[] f = new int[n];//前n个丑数
    int[] pos = new int[3];//对应 2 3 5三个质因子
    f[0] = 1;//第一个丑数1
    for(int i = 1; i < n; i ++)
    {
      int a = f[pos[0]] * 2;
      int b = f[pos[1]] * 3;
      int c = f[pos[2]] * 5;
      int mini = Math.min(Math.min(a, b), c);//一定只包含 2  3 5质因子
      if(f[pos[0]] * 2 == mini) pos[0] ++;//6 2*3  10 2*5
      if(f[pos[1]] * 3 == mini) pos[1] ++;//6 3*2
      if(f[pos[2]] * 5 == mini) pos[2] ++;// 10 5*2
      f[i] = mini;
    }
    return f[n - 1];
  }
}
```

面试题50. 第一个只出现一次的字符

2020年6月17日 6:15

面试题50. 第一个只出现一次的字符

难度简单28

在字符串 s 中找出第一个只出现一次的字符。如果没有，返回一个单空格。 s 只包含小写字母。

示例:

s = "abaccdeff"

返回 "b"

s = ""

返回 " "

来自 <https://leetcode-cn.com/problems/di-yi-ge-zhi-chu-xian-yi-ci-de-zi-fu-lcof/>

```
class Solution {
  public char firstUniqChar(String s) {
    ///hash表
    HashMap<Character, Boolean> hash = new HashMap<>();
    char[] ch = s.toCharArray();
    for(char c : ch)
      hash.put(c, !hash.containsKey(c));//先插入 true，如果重复了就变成false

    for(char c : ch)
      if(hash.get(c)) return c;

    return ' ';
  }
}
```

面试题51. 数组中的逆序对

2020年6月17日 6:22

面试题51. 数组中的逆序对

难度困难175

在数组中的两个数字，如果前面一个数字大于后面的数字，则这两个数字组成一个逆序对。输入一个数组，求出这个数组中的逆序对的总数。

示例 1:

输入: [7,5,6,4]

输出: 5

来自 <https://leetcode-cn.com/problems/shu-zu-zhong-de-ni-xu-dui-lcof/>

```
class Solution {
  //归并排序
  public int reversePairs(int[] nums) {
    int len = nums.length;
    if(len < 2)//逆序对要求两个数字以上
      return 0;
    int[] temp = new int[len];
    return mergeSort(nums, 0, len - 1, temp);
  }
  private int mergeSort(int[] nums, int left, int right, int[] temp){
    //递归终止条件
    if(left >= right)
      return 0;
    int mid = left + (right - left) / 2;//防止溢出
    //[left, mid]
    int leftPairs = mergeSort(nums, left, mid, temp);
    //[mid + 1, right]
    int rightPairs = mergeSort(nums, mid +1, right, temp);
    //[left mid] [mid + 1  right]
    if(nums[mid] <= nums[mid + 1]){
      return leftPairs + rightPairs;
    }
    //跨区间的逆序对
    int crossPairs = mergeSortCross(nums, left, mid, right, temp);
    return leftPairs + rightPairs + crossPairs;
  }
  private int mergeSortCross(int[] nums, int left,int mid, int right, int[] temp
){
    for(int i = left; i <= right; i ++)
      temp[i] = nums[i];
```

```
    int i = left;
    int j = mid + 1;
    //[left i mid]  [mid + 1 j  right]
    // mid - i + 1个逆序对
    int count = 0;
    for(int k = left; k <= right; k ++){
      if(i == mid  + 1){
        nums[k] = temp[j];
        j ++;
      }else if(j == right + 1){
        nums[k] = temp[i];
        i ++;
      }else if(temp[i] <= temp[j]){
        nums[k] = temp[i];
        i ++;
      }else{
        nums[k] = temp[j];
        j ++;
        count += mid - i + 1;
      }
    }
    return count;
  }
}
```

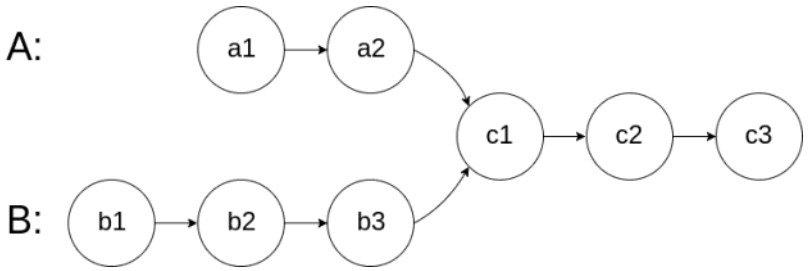
2020年6月17日 6:34

面试题52. 两个链表的第一个公共节点

难度简单54

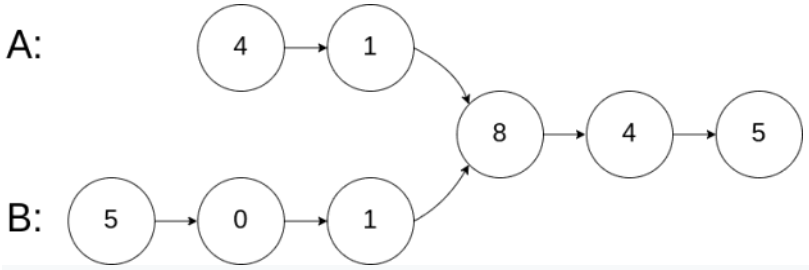
输入两个链表，找出它们的第一个公共节点。

如下面的两个链表：



在节点 c1 开始相交。

示例 1：

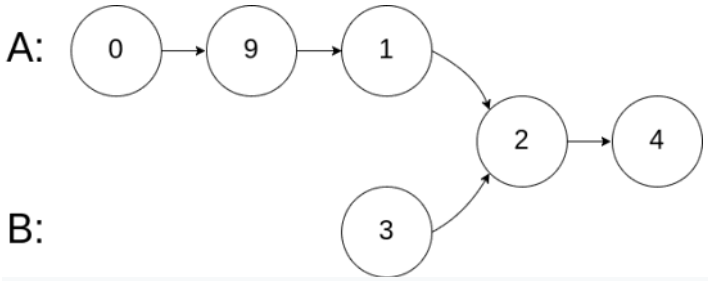


输入： intersectVal = 8, listA = [4,1,8,4,5], listB = [5,0,1,8,4,5], skipA = 2, skipB = 3

输出： Reference of the node with value = 8

输入解释： 相交节点的值为 8 （注意，如果两个列表相交则不能为 0）。从各自的表头开始算起，链表 A 为 [4,1,8,4,5]，链表 B 为 [5,0,1,8,4,5]。在 A 中，相交节点前有 2 个节点；在 B 中，相交节点前有 3 个节点。

示例 2：

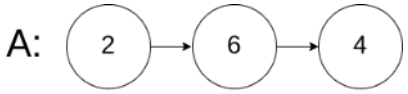


输入： intersectVal = 2, listA = [0,9,1,2,4], listB = [3,2,4], skipA = 3, skipB = 1

输出： Reference of the node with value = 2

输入解释： 相交节点的值为 2 （注意，如果两个列表相交则不能为 0）。从各自的表头开始算起，链表 A 为 [0,9,1,2,4]，链表 B 为 [3,2,4]。在 A 中，相交节点前有 3 个节点；在 B 中，相交节点前有 1 个节点。

示例 3：



输入：intersectVal = 0, listA = [2,6,4], listB = [1,5], skipA = 3, skipB = 2
输出：null
输入解释：从各自的表头开始算起，链表 A 为 [2,6,4]，链表 B 为 [1,5]。由于这两个链表不相交，所以 intersectVal 必须为 0，而 skipA 和 skipB 可以是任意值。
解释：这两个链表不相交，因此返回 null。

注意：

- 如果两个链表没有交点，返回 null.
- 在返回结果后，两个链表仍须保持原有的结构。
- 可假定整个链表结构中没有循环。
- 程序尽量满足 O(*n*) 时间复杂度，且仅用 O(*1*) 内存。
- 本题与主站 160 题相同：https://leetcode-cn.com/problems/intersection-of-two-linked-lists/

通过次数21,198

提交次数33,333

在真实的面试中遇到过这道题?

是

否

来自 <https://leetcode-cn.com/problems/liang-ge-lian-biao-de-di-yi-ge-gong-gong-jie-dian-lcof/>

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) {
 *         val = x;
 *         next = null;
 *     }
 * }
 */
public class Solution {
  public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
    ListNode idxA = headA;
    ListNode idxB = headB;
    while(idxA != idxB) {
      if (idxA == null) idxA = headB;
      else idxA = idxA.next;
```

```
            if (idxB == null) idxB = headA;
            else idxB = idxB.next;
        }
        return idxA;
    }
}
```

面试题53 - I. 在排序数组中查找数字 I

2020年6月18日 2:51

面试题53 - I. 在排序数组中查找数字 I

难度简单33

统计一个数字在排序数组中出现的次数。

示例 1:

输入: nums = [5,7,7,8,8,10], target = 8

输出: 2

示例 2:

输入: nums = [5,7,7,8,8,10], target = 6

输出: 0

来自 <https://leetcode-cn.com/problems/zai-pai-xu-shu-zu-zhong-cha-zhao-shu-zi-lcof/>

```
class Solution {
  public int search(int[] nums, int target) {
    //找到target所在位置 start
    //找到target + 1 所在位置 end
    //end - start +  (1/0)
    if(nums == null || nums.length == 0)
      return 0;

    int start = binarySearch(nums, target), end = binarySearch(nums, target + 1);
    return end - start + (nums[end] == target ? 1 : 0);
  }
  //二分
  private int binarySearch(int[] nums, int tar){
    int l = 0, r = nums.length - 1;
    while(l < r){
      int mid = l + (r - l) / 2;
      //[l  mid] [mid + 1 r]
      if(nums[mid] < tar)
        l = mid + 1;
      else
        r = mid;
    }
    return l;
  }
}
```

面试题53 - II. 0～n-1中缺失的数字

2020年6月18日 3:07

面试题53 - II. 0～n-1中缺失的数字

难度简单29

一个长度为n-1的递增排序数组中的所有数字都是唯一的，并且每个数字都在范围0～n-1之内。在范围0～n-1内的n个数字中有且只有一个数字不在该数组中，请找出这个数字。

示例 1:

输入: [0,1,3]

输出: 2

示例 2:

输入: [0,1,2,3,4,5,6,7,9]

输出: 8

来自 <https://leetcode-cn.com/problems/que-shi-de-shu-zi-lcof/>

```
class Solution {
  public int missingNumber(int[] nums) {
    //nums[i]  i
    //二分
    if(nums == null || nums.length == 0)
      return 0;

    //二分
    int l = 0, r = nums.length - 1;
    while(l < r){
      int mid = l + (r - l) / 2;
      if(nums[mid] != mid)
        r = mid;
      else
        l = mid + 1;
    }
    //[0 1 2] 3
    if(nums[r] == r)
      r ++;

    return r;
  }
}
```

面试题54. 二叉搜索树的第k大节点

2020年6月18日 3:12

面试题54. 二叉搜索树的第k大节点

难度简单32

给定一棵二叉搜索树，请找出其中第k大的节点。

示例 1:

输入: root = [3,1,4,null,2], k = 1

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

输出: 4

示例 2:

输入: root = [5,3,6,2,4,null,null,1], k = 3

```
    5
   / \
  3   6
 / \
2   4
 /
1
```

输出: 4

限制:

1 ≤ k ≤ 二叉搜索树元素个数

来自 <https://leetcode-cn.com/problems/er-cha-sou-suo-shu-de-di-kda-jie-dian-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  //中序序列 递增 左根右
  //中序序列反过来 递减
  int k1, res;
  public int kthLargest(TreeNode root, int k) {
```

```
        k1 = k;
        dfs(root);
        return res;
    }
    private void dfs(TreeNode root){
        if(root == null || k1 == 0) return;
        //右子树
        dfs(root.right);
        //根节点
        if(--k1 == 0) res = root.val;
        //左子树
        dfs(root.left);
    }
}
```

面试题55 - I. 二叉树的深度

2020年6月18日 3:24

面试题55 - I. 二叉树的深度

难度简单25

输入一棵二叉树的根节点，求该树的深度。从根节点到叶节点依次经过的节点（含根、叶节点）形成树的一条路径，最长路径的长度为树的深度。

例如：

给定二叉树 [3,9,20,null,null,15,7]，

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

返回它的最大深度 3 。

来自 <https://leetcode-cn.com/problems/er-cha-shu-de-shen-du-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public int maxDepth(TreeNode root) {
    if(root == null)
      return 0;

    int left = maxDepth(root.left);
    int right = maxDepth(root.right);
    return left>right ? left + 1 : right + 1;
  }
}
```

面试题55 - II. 平衡二叉树

2020年6月18日 3:29

面试题55 - II. 平衡二叉树

难度简单30

输入一棵二叉树的根节点，判断该树是不是平衡二叉树。如果某二叉树中任意节点的左右子树的深度相差不超过1，那么它就是一棵平衡二叉树。

示例 1:

给定二叉树 `[3,9,20,null,null,15,7]`

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

返回 `true` 。

示例 2:

给定二叉树 `[1,2,2,3,3,null,null,4,4]`

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

返回 `false` 。

来自 <https://leetcode-cn.com/problems/ping-heng-er-cha-shu-lcof/>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public int treeDepth(TreeNode root) {
    if(root ==null)return 0;
    int a = treeDepth(root.left);
    int b = treeDepth(root.right);
```

```
    return a>b?a+1:b+1;
  }
  public boolean isBalanced(TreeNode root) {
     if (root == null) {
       return true;
     }
     int left = treeDepth(root.left);
     int right = treeDepth(root.right);
     return Math.abs(left - right) > 1 ? false : true && isBalanced(root.left) && is
Balanced(root.right);
  }
}
```

面试题56 - I. 数组中数字出现的次数

2020年6月18日 3:39

面试题56 - I. 数组中数字出现的次数

难度中等158

一个整型数组 nums 里除两个数字之外，其他数字都出现了两次。请写程序找出这两个只出现一次的数字。要求时间复杂度是O(n)，空间复杂度是O(1)。

示例 1：

输入： nums = [4,1,4,6]

输出： [1,6] 或 [6,1]

示例 2：

输入： nums = [1,2,10,4,1,4,3,3]

输出： [2,10] 或 [10,2]

来自 <https://leetcode-cn.com/problems/shu-zu-zhong-shu-zi-chu-xian-de-ci-shu-lcof/>

```
class Solution {
  //异或
  // 1 ^ 1 = 0
  // A ^ 0 = A
  public int[] singleNumbers(int[] nums) {
    int sum = 0;//异或 first ^ second
    for(int x : nums)
      sum ^= x;
    //sum = first ^ second
    //二进制中某一位 1  0 不同
    //找到sum的二进制中 1 所在的那个位置
    int index = 0;
    while((sum >> index & 1) == 0) index ++;//找到了位置
    int first = 0;
    for(int x : nums)
      if((x >> index & 1) == 0) first ^= x;
    //first已经找到了
    int second = sum ^ first;
    // d = a ⊕ b ⊕ c 可以推出 a = d ⊕ b ⊕ c.
    return new int[]{first, second};
  }
}
```

面试题56 - II. 数组中数字出现的次数 II

2020年6月18日 3:56

面试题56 - II. 数组中数字出现的次数 II

难度中等38

在一个数组 nums 中除一个数字只出现一次之外，其他数字都出现了三次。请找出那个只出现一次的数字。

示例 1：

输入： nums = [3,4,3,3]

输出： 4

示例 2：

输入： nums = [9,1,7,9,7,9,7]

输出： 1

限制：

- 1 <= nums.length <= 10000
- 1 <= nums[i] < 2^31

来自 <https://leetcode-cn.com/problems/shu-zu-zhong-shu-zi-chu-xian-de-ci-shu-ii-lcof/>

```
class Solution {
  public int singleNumber(int[] nums) {
    int[] count = new int[32];
    for(int num : nums)
      for(int i = 0; i < 32 ; i ++){
        count[i] += num & 1;
        num >>= 1;
      }
    int res = 0, mod = 3;
    for(int i = 0; i < 32; i ++){
      res <<= 1;
      res += count[31 - i] % mod;
    }
    return res;
  }
}
```

```
class Solution {
  public int singleNumber(int[] nums) {
     //自动机
    int ones = 0, twos = 0;
    //{ones, twos}
```

```
    //{0,0} {1,0} {0,1} {0,0}
    for(int x : nums)
    {
      ones = (ones ^ x) & ~twos;
      twos = (twos ^ x) & ~ones;
    }
    return ones;
  }
}
```

面试题57. 和为s的两个数字

2020年6月18日 4:23

面试题57. 和为s的两个数字

难度简单22

输入一个递增排序的数组和一个数字s，在数组中查找两个数，使得它们的和正好是s。如果有多对数字的和等于s，则输出任意一对即可。

示例 1：

输入： nums = [2,7,11,15], target = 9

输出： [2,7] 或者 [7,2]

示例 2：

输入： nums = [10,26,30,31,47,60], target = 40

输出： [10,30] 或者 [30,10]

来自 <https://leetcode-cn.com/problems/he-wei-sde-liang-ge-shu-zi-lcof/>

//哈希表

```
class Solution {
  public int[] twoSum(int[] nums, int target) {
        Set<Integer> s = new HashSet<>();
    for (int i=0;i<nums.length;i++){
      s.add(nums[i]);
    }
    for (int i=0;i<nums.length;i++){
      if(s.contains(target-nums[i])){
        return new int[]{nums[i],target-nums[i]};
      }
    }
    return null;
  }
}
```

双指针

```
class Solution {
  public int[] twoSum(int[] nums, int target) {
    //双指针
    //递增
    int i = 0, j = nums.length - 1;
    while(i < j){
      int s = nums[i] + nums[j];
      if(s < target) i ++;
      else if(s > target) j --;
      else return new int[]{nums[i], nums[j]};
    }
```

```
        return null;
    }
}
```

面试题57 - II. 和为s的连续正数序列

2020年6月18日 4:33

面试题57 - II. 和为s的连续正数序列

难度简单111

输入一个正整数 target ，输出所有和为 target 的连续正整数序列（至少含有两个数）。

序列内的数字由小到大排列，不同序列按照首个数字从小到大排列。

示例 1：

输入： target = 9

输出： [[2,3,4],[4,5]]

示例 2：

输入： target = 15

输出： [[1,2,3,4,5],[4,5,6],[7,8]]

来自 <https://leetcode-cn.com/problems/he-wei-sde-lian-xu-zheng-shu-xu-lie-lcof/>

```
class Solution {
  public int[][] findContinuousSequence(int target) {
    List<int[]> re = new ArrayList<>();
    for(int i = 1, j = 1, s = 1; i < target; i ++){
      while(s < target){
        j ++;
        s += j;
      }
      if(s == target && j - i >= 1){
        int[] one = new int[j - i + 1];
        for(int k = i; k <= j; k ++)
          one[k - i] = k;
        re.add(one);
      }
      s -= i;
    }
    return re.toArray(new int[re.size()][]);
  }
}
```

面试题58 - I. 翻转单词顺序

2020年6月18日 5:02

面试题58 - I. 翻转单词顺序

难度简单22

输入一个英文句子，翻转句子中单词的顺序，但单词内字符的顺序不变。为简单起见，标点符号和普通字母一样处理。例如输入字符串"I am a student. "，则输出"student. a am I"。

示例 1：
输入: "the sky is blue"
输出: "blue is sky the"
示例 2：
输入: " hello world! "
输出: "world! hello"
解释: 输入字符串可以在前面或者后面包含多余的空格，但是反转后的字符不能包括。
示例 3：
输入: "a good example"
输出: "example good a"
解释: 如果两个单词间有多余的空格，将反转后单词间的空格减少到只含一个。

说明：

- 无空格字符构成一个单词。
- 输入字符串可以在前面或者后面包含多余的空格，但是反转后的字符不能包括。
- 如果两个单词间有多余的空格，将反转后单词间的空格减少到只含一个。

注意：本题与主站 151 题相同：https://leetcode-cn.com/problems/reverse-words-in-a-string/
注意：此题对比原题有改动

来自 <https://leetcode-cn.com/problems/fan-zhuan-dan-ci-shun-xu-lcof/>

```
class Solution {
  public String reverseWords(String s) {
    s = s.trim();//去除首尾空格
    int j = s.length() - 1, i = j;
    StringBuilder res = new StringBuilder();
    while(i >= 0){
      while(i >= 0 && s.charAt(i) != ' ') i --;//找到空格
      res.append(s.substring(i + 1, j + 1) + " ");//添加单词
      while(i >= 0 && s.charAt(i) == ' ') i --;//跳过空格
      j = i;
    }
    return res.toString().trim();
  }
}
```

面试题58 - II. 左旋转字符串

2020年6月18日 5:10

面试题58 - II. 左旋转字符串

难度简单35

字符串的左旋转操作是把字符串前面的若干个字符转移到字符串的尾部。请定义一个函数实现字符串左旋转操作的功能。比如，输入字符串"abcdefg"和数字2，该函数将返回左旋转两位得到的结果"cdefgab"。

示例 1：
输入: s = "abcdefg", k = 2
输出: "cdefgab"
示例 2：
输入: s = "lrloseumgh", k = 6
输出: "umghlrlose"

限制：

- 1 <= k < s.length <= 10000

通过次数36,752

提交次数43,397

来自 <https://leetcode-cn.com/problems/zuo-xuan-zhuan-zi-fu-chuan-lcof/>

```
class Solution {
  public String reverseLeftWords(String s, int n) {
    return s.substring(n, s.length()) + s.substring(0, n);
  }
}
```

面试题59 - I. 滑动窗口的最大值

2020年6月18日　　5:14

面试题59 - I. 滑动窗口的最大值

难度简单52

给定一个数组 nums 和滑动窗口的大小 k，请找出所有滑动窗口里的最大值。

示例:

输入: *nums* = [1,3,-1,-3,5,3,6,7], 和 *k* = 3

输出: [3,3,5,5,6,7]

解释:

```
滑动窗口的位置                最大值
---------------             -----
[1  3  -1] -3  5  3  6  7     3
 1 [3  -1  -3] 5  3  6  7     3
 1  3 [-1  -3  5] 3  6  7     5
 1  3  -1 [-3  5  3] 6  7     5
 1  3  -1  -3 [5  3  6] 7     6
 1  3  -1  -3  5 [3  6  7]    7
```

提示:

你可以假设 *k* 总是有效的，在输入数组不为空的情况下，1 ≤ k ≤ 输入数组的大小。

注意：本题与主站 239 题相同：https://leetcode-cn.com/problems/sliding-window-maximum/

通过次数21,983

提交次数50,128

在真实的面试中遇到过这道题?

是

否

来自 <https://leetcode-cn.com/problems/hua-dong-chuang-kou-de-zui-da-zhi-lcof/>

```
class Solution {
  public int[] maxSlidingWindow(int[] nums, int k) {
    if(nums.length == 0 || k == 0)
      return new int[0];
    int[] res = new int[nums.length-k+1];
    Deque<Integer> q = new LinkedList<>();
    int j=0;
    for(int i=0;i<nums.length;i++){
      if(q.size()>0 && i-q.peekFirst() >= k)//判断队头是否需要出队
        q.pollFirst();
      while(q.size()>0&&nums[q.peekLast()]<nums[i])//维护队列单调性
```

```
            q.pollLast();
          q.add(i);
          if(i >= k-1){
            res[j++] = (nums[q.peekFirst()]);//取队头作为窗口最大元素
          }
        }
        return res;
    }
}
```

面试题59 - II. 队列的最大值

2020年6月18日 5:50

面试题59 - II. 队列的最大值

难度中等105

请定义一个队列并实现函数 max_value 得到队列里的最大值，要求函数max_value、push_back 和 pop_front 的**均摊**时间复杂度都是O(1)。

若队列为空，pop_front 和 max_value 需要返回 -1

示例 1：

输入:

["MaxQueue","push_back","push_back","max_value","pop_front","max_value"]

[[],[1],[2],[],[],[]]

输出: [null,null,null,2,1,2]

示例 2：

输入:

["MaxQueue","pop_front","max_value"]

[[],[],[]]

输出: [null,-1,-1]

来自 <https://leetcode-cn.com/problems/dui-lie-de-zui-da-zhi-lcof/>

```
class MaxQueue {
  Queue<Integer> que;
  Deque<Integer> deq;//记录最大值
  public MaxQueue() {
    que = new LinkedList<>();
    deq = new LinkedList<>();
  }

  public int max_value() {
    //取双端队列的队首作为做大值
    return deq.size()> 0 ? deq.peek() : -1;
  }

  public void push_back(int value) {
    que.add(value);
    while(deq.size() > 0 && deq.peekLast() < value){
      deq.pollLast();//保证双端队列存的就是最大的值，将队尾小于value的元素全部删除
    }
    deq.add(value);
  }

  public int pop_front() {
    int v = que.size() > 0 ? que.poll() : -1;
    if(deq.size() > 0 && deq.peekFirst() == v)
```

```
            deq.pollFirst();

        return v;
    }
}
/**
 * Your MaxQueue object will be instantiated and called as such:
 * MaxQueue obj = new MaxQueue();
 * int param_1 = obj.max_value();
 * obj.push_back(value);
 * int param_3 = obj.pop_front();
 */
```

面试题60. n个骰子的点数

2020年6月18日 6:02

面试题60. n个骰子的点数

难度简单66

把n个骰子扔在地上，所有骰子朝上一面的点数之和为s。输入n，打印出s的所有可能的值出现的概率。

你需要用一个浮点数数组返回答案，其中第 i 个元素代表这 n 个骰子所能掷出的点数集合中第 i 小的那个的概率。

示例 1:

输入: 1

输出: [0.16667,0.16667,0.16667,0.16667,0.16667,0.16667]

示例 2:

输入: 2

输出:

[0.02778,0.05556,0.08333,0.11111,0.13889,0.16667,0.13889,0.11111,0.08333,0.05556,0.02778]

限制:

1 <= n <= 11

来自 <https://leetcode-cn.com/problems/nge-tou-zi-de-dian-shu-lcof/>

```
class Solution {
  public double[] twoSum(int n) {
    //1 1~6   6   6   1 + 5 * 1
    //2 2~12  36  11  1 + 5 * 2
    //3 3~18  6^3 16  1 + 5 * 3
    //4 4~24  6^4 21  1 + 5 * 4
    //n n~n*6
    //方案总数 6^n
    double all = Math.pow(6,n);//方案总数
    //状态数组
    int[][] dp = new int[n + 1][6*n + 1];
    dp[0][0] = 1;
    //dp[i][j]有i个骰子 点数和为j的方案总数
    //先统计每个点数可行方案的数量
    for(int i = 1; i <= n; i ++)//n个骰子
      for(int j = 1; j <= 6 * i; j ++)
        for(int k = 1; k <= Math.min(j, 6); k ++)
          dp[i][j] += dp[i - 1][j - k];//前i次总和为j的方案总数
    double[] res = new double[1 + 5 * n];
    for(int i = n; i <= n * 6; i ++)
      res[i - n] = dp[n][i] / all;//然后再换成概率
    return res;
```

```
    }
}
```

面试题61. 扑克牌中的顺子

2020年6月19日 1:22

面试题61. 扑克牌中的顺子

难度简单31

从扑克牌中随机抽5张牌，判断是不是一个顺子，即这5张牌是不是连续的。2～10为数字本身，A为1，J为11，Q为12，K为13，而大、小王为 0 ，可以看成任意数字。A 不能视为 14。

示例 1:

输入: [1,2,3,4,5]

输出: True

示例 2:

输入: [0,0,1,2,5]

输出: True

限制:

数组长度为 5

数组的数取值为 [0, 13] .

来自 <https://leetcode-cn.com/problems/bu-ke-pai-zhong-de-shun-zi-lcof/>

```
class Solution {
  public boolean isStraight(int[] nums) {
    if(nums.length != 5) return false;
    Arrays.sort(nums);// 0 0 xxx
    //去除0
    int k = 0;
    while(nums[k] == 0){//2
      k ++; //3
    }
    //nums[k] != 0
    //看有没有重复数字 如果有 false
    for(int i = k + 1; i < nums.length; i ++)
      if(nums[i] == nums[i - 1])
        return false;
    //看除了0以外的最大 - 最小 <= 4
    return nums[4] - nums[k] <= 4;
  }
}
```

面试题62. 圆圈中最后剩下的数字

2020年6月19日 2:03

面试题62. 圆圈中最后剩下的数字

难度简单156

0,1,,n-1这n个数字排成一个圆圈，从数字0开始，每次从这个圆圈里删除第m个数字。求出这个圆圈里剩下的最后一个数字。

例如，0、1、2、3、4这5个数字组成一个圆圈，从数字0开始每次删除第3个数字，则删除的前4个数字依次是2、0、4、1，因此最后剩下的数字是3。

示例 1：

输入: n = 5, m = 3

输出: 3

示例 2：

输入: n = 10, m = 17

输出: 2

来自 <https://leetcode-cn.com/problems/yuan-quan-zhong-zui-hou-sheng-xia-de-shu-zi-lcof/>

```
class Solution {
  public int lastRemaining(int n, int m) {
    //旧编号 0    1   2 ...   m-1 m m+1  ... n-1
    //新编号 -m -m+1 -m+2 ... x   0  1  ...  n-1-m
    //递推关系 f(n,m) = (f(n - 1, m) + m) % n
    if(n == 1) return 0;
    return (lastRemaining(n - 1, m) + m) % n;
  }
}
```

面试题63. 股票的最大利润

2020年6月19日 2:16

面试题63. 股票的最大利润

难度中等30

假设把某股票的价格按照时间先后顺序存储在数组中，请问买卖该股票一次可能获得的最大利润是多少？

示例 1:

输入: [7,1,5,3,6,4]

输出: 5

解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。

注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格。

示例 2:

输入: [7,6,4,3,1]

输出: 0

解释: 在这种情况下, 没有交易完成, 所以最大利润为 0。

来自 <https://leetcode-cn.com/problems/gu-piao-de-zui-da-li-run-lcof/>

```
class Solution {
  public int maxProfit(int[] prices) {
    if(prices.length == 0)
      return 0;
    int maxv = 0;//最大利润
    int minv = prices[0];//最小价格，默认第一个价格最小
    for(int i = 1; i < prices.length; i ++){
      maxv = Math.max(prices[i] - minv, maxv);
      minv = Math.min(minv, prices[i]);
    }
    return maxv;
  }
}
```

2020年6月19日 2:22

面试题64. 求1+2+...+n

难度中等150

求 1+2+...+n，要求不能使用乘除法、for、while、if、else、switch、case等关键字及条件判断语句（A?B:C）。

示例 1：

输入: n = 3

输出: 6

示例 2：

输入: n = 9

输出: 45

来自 <https://leetcode-cn.com/problems/qiu-12n-lcof/>

//递归

```
class Solution {
  public int sumNums(int n) {
    return n == 0 ? 0 : n + sumNums(n - 1);
  }
}
//
class Solution {
 int[] test=new int[]{0};
  public int sumNums(int n) {
    try{
      return test[n];
    }catch(Exception e){
      return n+sumNums(n-1);
    }
  }
}
```

面试题65. 不用加减乘除做加法

2020年6月19日 2:35

面试题65. 不用加减乘除做加法

难度简单34

写一个函数，求两个整数之和，要求在函数体内不得使用 “+”、“-”、“*”、“/” 四则运算符号。

示例:

输入: a = 1, b = 1

输出: 2

提示:

- a, b 均可能是负数或 0
- 结果不会溢出 32 位整数

通过次数11,046

提交次数19,793

来自 <https://leetcode-cn.com/problems/bu-yong-jia-jian-cheng-chu-zuo-jia-fa-lcof/>

```
class Solution {
  public int add(int a, int b) {
    //与 异或
    while(b != 0){//进位找到0就停止 a就是和
      int c = (a & b) << 1;//c进位
      a ^= b;//a非进位部分的和
      b = c;
    }
    return a;
  }
}
```

面试题66. 构建乘积数组

2020年6月19日 2:40

面试题66. 构建乘积数组

难度简单21

给定一个数组 A[0,1,...,n-1]，请构建一个数组 B[0,1,...,n-1]，其中 B 中的元素 B[i]=A[0]×A[1]×...×A[i-1]×A[i+1]×...×A[n-1]。不能使用除法。

示例:

输入: [1,2,3,4,5]

输出: [120,60,40,30,24]

提示:

- 所有元素乘积之和不会溢出 32 位整数
- a.length <= 100000

通过次数12,214

提交次数21,118

在真实的面试中遇到过这道题?

来自 <https://leetcode-cn.com/problems/gou-jian-cheng-ji-shu-zu-lcof/>

```
class Solution {
  public int[] constructArr(int[] a) {
    //想办法在 i的时候 避开× ai
    int n = a.length;
    int[] res = new int[n];
    for(int i = 0,p = 1; i < n; i ++){
      res[i] = p;// a1 **** ai -1
      p *= a[i];
    }
    for(int i = n - 1, p = 1; i >= 0; i --){
      res[i] *= p;//ai+1 **** an-1
      p *= a[i];
    }
    return res;
  }
}
```

面试题67. 把字符串转换成整数

2020年6月19日 2:51

面试题67. 把字符串转换成整数

难度中等12

写一个函数 StrToInt，实现把字符串转换成整数这个功能。不能使用 atoi 或者其他类似的库函数。

首先，该函数会根据需要丢弃无用的开头空格字符，直到寻找到第一个非空格的字符为止。

当我们寻找到的第一个非空字符为正或者负号时，则将该符号与之后面尽可能多的连续数字组合起来，作为该整数的正负号；假如第一个非空字符是数字，则直接将其与之后连续的数字字符组合起来，形成整数。

该字符串除了有效的整数部分之后也可能会存在多余的字符，这些字符可以被忽略，它们对于函数不应该造成影响。

注意：假如该字符串中的第一个非空格字符不是一个有效整数字符、字符串为空或字符串仅包含空白字符时，则你的函数不需要进行转换。

在任何情况下，若函数不能进行有效的转换时，请返回 0。

说明：

假设我们的环境只能存储 32 位大小的有符号整数，那么其数值范围为 [−231，231 − 1]。如果数值超过这个范围，请返回 INT_MAX (231 − 1) 或 INT_MIN (−231) 。

示例 1:

输入: "42"

输出: 42

示例 2:

输入: " -42"

输出: -42

解释: 第一个非空白字符为 '-', 它是一个负号。

我们尽可能将负号与后面所有连续出现的数字组合起来，最后得到 -42 。

示例 3:

输入: "4193 with words"

输出: 4193

解释: 转换截止于数字 '3' ，因为它的下一个字符不为数字。

示例 4:

输入: "words and 987"

输出: 0

解释: 第一个非空字符是 'w', 但它不是数字或正、负号。

因此无法执行有效的转换。

示例 5:

输入: "-91283472332"

输出: -2147483648

解释: 数字 "-91283472332" 超过 32 位有符号整数范围。

因此返回 INT_MIN (−231) 。

来自 <https://leetcode-cn.com/problems/ba-zi-fu-chuan-zhuan-huan-cheng-zheng-shu-lcof/>

```
class Solution {
  public int strToInt(String str) {
```

```
    //2147483647   214748364 + 还有一个数字 7比较大小
    //-2147483648  214748364 + 还有一个数字
    char[] c = str.trim().toCharArray();
    if(c.length == 0)
      return 0;

    int res = 0, sign = 1;//res返回结果 sign 1正数 -1负数
    int temp = Integer.MAX_VALUE / 10;//214748364
    int index = 1;
    if(c[0] == '-') sign = -1;
    else if(c[0] != '+') index = 0;
    for(int i = index; i < c.length; i ++){
      //"4193 with words"  4193
      //"words and 987"  0
      if(c[i] < '0' || c[i] > '9') break;
      if(res > temp || (res == temp && c[i] > '7'))
        return sign == 1 ? Integer.MAX_VALUE : Integer.MIN_VALUE;
      res = res * 10 + c[i] - '0';
    }
    return sign * (int)res;
  }
}
```

面试题68 - I. 二叉搜索树的最近公共祖先

2020年6月19日 3:10

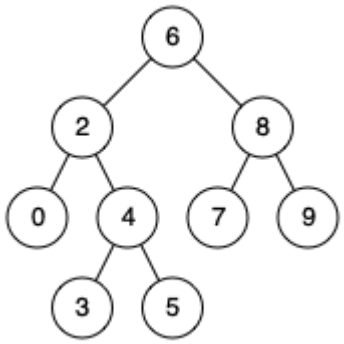
面试题68 - I. 二叉搜索树的最近公共祖先

难度简单29

给定一个二叉搜索树, 找到该树中两个指定节点的最近公共祖先。

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

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



示例 1:

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

输出: 6

解释: 节点 2 和节点 8 的最近公共祖先是 6。

示例 2:

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

输出: 2

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

说明:

- 所有节点的值都是唯一的。
- p、q 为不同节点且均存在于给定的二叉搜索树中。

注意：本题与主站 235 题相同：https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-search-tree/

通过次数17,744

提交次数26,164

在真实的面试中遇到过这道题?

来自 <https://leetcode-cn.com/problems/er-cha-sou-suo-shu-de-zui-jin-gong-gong-zu-xian-lcof/>

```
/**
 * Definition for a binary tree node.
```

```
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
//迭代
class Solution {
  public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
    if(p.val > q.val){//确保p  < q
      TreeNode temp = p;
      p = q;
      q = temp;
    }
    while(root != null){
      if(root.val < p.val)
        root = root.right;
      else if(root.val > q.val)
        root = root.left;
      else break;
    }
    return root;
  }
}
```

//递归

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
  public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
    if(root.val < p.val && root.val < q.val)
      return lowestCommonAncestor(root.right, p, q);
    if(root.val > p.val && root.val > q.val)
      return lowestCommonAncestor(root.left, p, q);
    return root;
  }
}
```

面试题68 - II. 二叉树的最近公共祖先

2020年6月19日 3:33

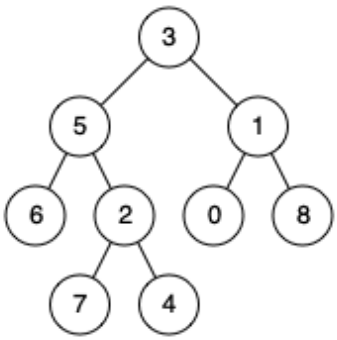
面试题68 - II. 二叉树的最近公共祖先

难度简单68

给定一个二叉树, 找到该树中两个指定节点的最近公共祖先。

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

例如，给定如下二叉树: root = [3,5,1,6,2,0,8,null,null,7,4]



示例 1:

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

输出: 3

解释: 节点 5 和节点 1 的最近公共祖先是节点 3。

示例 2:

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

输出: 5

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

说明:

- 所有节点的值都是唯一的。
- p、q 为不同节点且均存在于给定的二叉树中。

注意：本题与主站 236 题相同：https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-tree/

来自 <https://leetcode-cn.com/problems/er-cha-shu-de-zui-jin-gong-gong-zu-xian-lcof/>

```
class Solution {
  public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
    if(root == null || root == p || root == q) return root;
    TreeNode left = lowestCommonAncestor(root.left, p, q);
    TreeNode right = lowestCommonAncestor(root.right, p, q);
    if(left == null) return right;
    if(right == null) return left;
    return root;
  }
}
```