

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 $0 \sim n-1$ 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:

$2 \leq n \leq 100000$

来源: 力扣 (LeetCode)

链接: <https://leetcode-cn.com/problems/shu-zu-zhong-zhong-fu-de-shu-zi-lcof>

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

```
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false 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>

著作权归领扣网络所有。商业转载请联系官方授权, 非商业转载请注明出处。

```
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:

- $0 \leq n \leq 100$

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;  
    }  
}
```