LeetCodeCrowdsource

Release 1.0

Mar 09, 2020

Contents:

1 题解 **1**

1.1 [0001] 两数之和 . . . 1
1.2 [0002] 两数相加 . . . 4
1.3 [0003] 无重复字符的最长子串 . . . 6
1.4 [0004] 寻找两个有序数组的中位数 . . . 7
1.5 [0005] 最长回文子串 . . . 9
1.6 [0006] Z 字形变换 . . . 11
1.7 [0007] 整数反转 . . . 12
1.8 [0008] 字符串转换整数 (atoi) . . . 14
1.9 [0009] 回文数 . . . 16
1.10 [0010] 正则表达式匹配 . . . 18
1.11 [0011] 盛最多水的容器 . . . 20
1.12 [0012] 整数转罗马数字 . . . 21
1.13 [0013] 罗马数字转整数 . . . 23
1.14 [0014] 最长公共前缀 . . . 25
1.15 [0015] 三数之和 . . . 27
1.16 [0016] 最接近的三数之和 . . . 29
1.17 [0017] 电话号码的字母组合 . . . 30
1.18 [0018] 四数之和 . . . 32
1.19 [0019] 删除链表的倒数第 N 个节点 . . . 34
1.20 [0020] 有效的括号 . . . 36

2 专题 **39**

2.1 专题-BFS 算法 . . . 39
2.2 专题-DFS 算法 . . . 39
2.3 专题-DP 算法 . . . 39
2.4 专题-STL 算法 . . . 40

2.5 专题-二分算法 . . . 40
2.6 专题-二叉树 . . . 40
2.7 专题-回溯算法 . . . 40
2.8 专题-图论 . . . 41
2.9 专题-堆 . . . 41
2.10 专题-字符串 . . . 41
2.11 专题-排序算法 . . . 41
2.12 专题-搜索算法 . . . 41
2.13 专题-数学 . . . 42
2.14 专题-数组 . . . 42
2.15 专题-数论 . . . 42
2.16 专题-枚举算法 . . . 42
2.17 专题-查找算法 . . . 43
2.18 专题-栈 . . . 43
2.19 专题-模拟算法 . . . 43
2.20 专题-贪心算法 . . . 43
2.21 专题-递归算法 . . . 43
2.22 专题-链表 . . . 44
2.23 专题-队列 . . . 44
2.24 专题-随机化 . . . 44

CHAPTER 1

题解

1.1 [0001] 两数之和

- https://leetcode-cn.com/problems/two-sum

1.1.1 题目描述

给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。

你可以假设每种输入只会对应一个答案。但是，你不能重复利用这个数组中同样的元素。

示例:

Related Topics

数组

哈希表

1.1.2 题目代码

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
```

(continues on next page)

(continued from previous page)

```
    }
};
```

1.1.3 题目解析

方法一

分析

- 这道题是一个查找问题。可以通**过哈希表**加速。
- 可以通过遍历数组中的每个元素 `nums[i]`，查找 `target-nums[i]`。
- 查找 `target-nums[i]`，可以遍历整个数组 `nums`，总的时间复杂度为 `$O(n^2)$`；也可以使用哈希表记录已查元素及其索引，中的时间复杂度为 `$O(n)$`。

思路

- 创建哈希表：`<nums[i], index>`。用于存储元素及其索引。
- 遍历数组的每个元素 `nums[i]`。检查哈希表中是否存在 `target-nums[i]`，如果存在，返回两个元素的索引；否则，将当前元素存储到哈希表中，处理下一个元素。
- 如果遍历完所有元素，查找失败，返回空数组。

注意

- 边界检查：数组为空；
- 处理查找失败情况；

知识点

- 数组
- 查找
- 哈希表

复杂度

- 时间复杂度：O(n)
- 空间复杂度：O(n)

参考

- https://www.cnblogs.com/grandyang/p/4130379.html

答案

```
//#include <unordered_map>
//
//using namespace std;

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        // 输入检查
        if(nums.empty())
            return {};
        // 创建哈希表 <num, index>, 加速查找
        unordered_map<int, int> m;
        for(int i=0; i<nums.size(); i++){
            if(m.count(target - nums[i]))
                return {i, m[target-nums[i]]};
            m[nums[i]] = i; // <num, index>
        }
        return {};
    }
};
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.2 [0002] 两数相加

- https://leetcode-cn.com/problems/add-two-numbers

1.2.1 题目描述

给出两个 非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照 逆序 的方式存储的，并且它们的每个节点只能存储 一位 数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例：

Related Topics

链表

数学

1.2.2 题目代码

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
```

(continues on next page)

(continued from previous page)

```
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {

    }
};
```

1.2.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.3 [0003] 无重复字符的最长子串

- https://leetcode-cn.com/problems/longest-substring-without-repeating-characters

1.3.1 题目描述

给定一个字符串，请你找出其中不含有重复字符的 最长子串 的长度。

示例 1:

示例 2:

示例 3:

Related Topics

哈希表

双指针

字符串

Sliding Window

1.3.2 题目代码

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {

    }
};
```

1.3.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.4 [0004] 寻找两个有序数组的中位数

- https://leetcode-cn.com/problems/median-of-two-sorted-arrays

1.4.1 题目描述

给定两个大小为 m 和 n 的有序数组 nums1 和 nums2。

请你找出这两个有序数组的中位数，并且要求算法的时间复杂度为 O(log(m + n))。

你可以假设 nums1 和 nums2 不会同时为空。

示例 1:

示例 2:

Related Topics

数组

二分查找

分治算法

1.4.2 题目代码

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

    }
};
```

1.4.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.5 [0005] 最长回文子串

- https://leetcode-cn.com/problems/longest-palindromic-substring

1.5.1 题目描述

给定一个字符串 s，找到 s 中最长的回文子串。你可以假设 s 的最大长度为 1000。

示例 1:

示例 2:

Related Topics

字符串

动态规划

1.5.2 题目代码

```
class Solution {
public:
    string longestPalindrome(string s) {

    }
};
```

1.5.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.6 [0006] Z 字形变换

- https://leetcode-cn.com/problems/zigzag-conversion

1.6.1 题目描述

将一个给定字符串根据给定的行数，以从上往下、从左到右进行 Z 字形排列。

比如输入字符串为 “LEETCODEISHIRING” 行数为 3 时，排列如下：

之后，你的输出需要从左往右逐行读取，产生出一个新的字符串，比如：”LCIRETOESIIGEDHN”。

请你实现这个将字符串进行指定行数变换的函数：

示例 1:

示例 2:

Related Topics

字符串

1.6.2 题目代码

```
class Solution {
public:
    string convert(string s, int numRows) {

    }
};
```

1.6.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.7 [0007] 整数反转

- https://leetcode-cn.com/problems/reverse-integer

1.7.1 题目描述

给出一个 32 位的有符号整数，你需要将这个整数中每位上的数字进行反转。

示例 1:

示例 2:

示例 3:

注意:

假设我们的环境只能存储得下 32 位的有符号整数，则其数值范围为 $[-231, 231 - 1]$。请根据这个假设，如果反转后整数溢出那么就返回 0。

Related Topics

数学

1.7.2 题目代码

```
class Solution {
public:
    int reverse(int x) {

    }
};
```

1.7.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.8 [0008] 字符串转换整数 (atoi)

- atoi

1.8.1 题目描述

请你来实现一个 atoi 函数，使其能将字符串转换成整数。

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:

示例 2:

示例 3:

示例 4:

示例 5:

Related Topics

数学

字符串

1.8.2 题目代码

```
class Solution {
public:
    int myAtoi(string str) {

    }
};
```

1.8.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.9 [0009] 回文数

- https://leetcode-cn.com/problems/palindrome-number

1.9.1 题目描述

判断一个整数是否是回文数。回文数是指正序（从左向右）和倒序（从右向左）读都是一样的整数。

示例 1:

示例 2:

示例 3:

进阶:

你能不将整数转为字符串来解决这个问题吗?

Related Topics

数学

1.9.2 题目代码

```
class Solution {
public:
    bool isPalindrome(int x) {

    }
};
```

1.9.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.10 [0010] 正则表达式匹配

- https://leetcode-cn.com/problems/regular-expression-matching

1.10.1 题目描述

给你一个字符串 s 和一个字符规律 p，请你来实现一个支持 ‘.’ 和 ’*’ 的正则表达式匹配。

所谓匹配，是要涵盖 整个 字符串 s 的，而不是部分字符串。

说明:

```
<li><code>s</code>  可能为空，且只包含从 <code>a-z</code>  的小写字母。</li>
<li><code>p</code>  可能为空，且只包含从 <code>a-z</code>  的小写字母，以及
字符 <code>.</code>  和 <code>*</code>。</li>
```

示例 1:

示例 2:

示例 3:

示例 4:

示例 5:

Related Topics

字符串

动态规划

回溯算法

1.10.2 题目代码

```
class Solution {
public:
    bool isMatch(string s, string p) {

    }
};
```

1.10.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.11 [0011] 盛最多水的容器

- https://leetcode-cn.com/problems/container-with-most-water

1.11.1 题目描述

给定 n 个非负整数 a1，a2，…，an，每个数代表坐标中的一个点 (i, ai) 。在坐标内画 n 条垂直线，垂直线 i 的两个端点分别为 (i, ai) 和 (i, 0)。找出其中的两条线，使得它们与 x 轴共同构成的容器可以容纳最多的水。

说明：你不能倾斜容器，且 n 的值至少为 2。

图中垂直线代表输入数组 [1,8,6,2,5,4,8,3,7]。在此情况下，容器能够容纳水（表示为蓝色部分）的最大值为 49。

示例:

Related Topics

数组

双指针

1.11.2 题目代码

```
class Solution {
public:
    int maxArea(vector<int>& height) {

    }
};
```

1.11.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.12 [0012] 整数转罗马数字

- https://leetcode-cn.com/problems/integer-to-roman

1.12.1 题目描述

罗马数字包含以下七种字符： I， V， X， L，C，D 和 M。

例如, 罗马数字 2 写做 II , 即为两个并列的 1。12 写做 XII , 即为 X + II 。27 写做 XXVII, 即为 XX + V + II 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

```
<li><code>I</code>  可以放在 <code>V</code> (5) 和 <code>X</code>&
↪nbsp;(10) 的左边，来表示 4 和 9。</li>
<li><code>X</code>  可以放在 <code>L</code> (50) 和 <code>C</code>&
↪nbsp;(100) 的左边，来表示 40 和 90。 </li>
<li><code>C</code>  可以放在 <code>D</code> (500) 和 <code>M</code>&
↪nbsp;(1000) 的左边，来表示 400 和 900。</li>
```

给定一个整数，将其转为罗马数字。输入确保在 1 到 3999 的范围内。

示例 1:

示例 2:

示例 3:

示例 4:

示例 5:

Related Topics

数学

字符串

1.12.2 题目代码

```
class Solution {
public:
    string intToRoman(int num) {

    }
};
```

1.12.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.13 [0013] 罗马数字转整数

- https://leetcode-cn.com/problems/roman-to-integer

1.13.1 题目描述

罗马数字包含以下七种字符: I， V， X， L，C，D 和 M。

例如, 罗马数字 2 写做 II , 即为两个并列的 1。12 写做 XII , 即为 X + II 。27 写做 XXVII, 即为 XX + V + II 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

```
<li><code>I</code>  可以放在 <code>V</code> (5) 和 <code>X</code>&
↪nbsp;(10) 的左边，来表示 4 和 9。</li>
<li><code>X</code>  可以放在 <code>L</code> (50) 和 <code>C</code>&
↪nbsp;(100) 的左边，来表示 40 和 90。 </li>
<li><code>C</code>  可以放在 <code>D</code> (500) 和 <code>M</code>&
↪nbsp;(1000) 的左边，来表示 400 和 900。</li>
```

给定一个罗马数字，将其转换成整数。输入确保在 1 到 3999 的范围内。

示例 1:

示例 2:

示例 3:

示例 4:

示例 5:

Related Topics

数学

字符串

1.13.2 题目代码

```
class Solution {
public:
    int romanToInt(string s) {

    }
};
```

1.13.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.14 [0014] 最长公共前缀

- https://leetcode-cn.com/problems/longest-common-prefix

1.14.1 题目描述

编写一个函数来查找字符串数组中的最长公共前缀。

如果不存在公共前缀，返回空字符串 ”“。

示例 1:

示例 2:

说明:

所有输入只包含小写字母 a-z 。

Related Topics

字符串

1.14.2 题目代码

```
class Solution {
public:
    string longestCommonPrefix(vector<string>& strs) {

    }
};
```

1.14.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.15 [0015] 三数之和

- https://leetcode-cn.com/problems/3sum

1.15.1 题目描述

给定一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？ 找出所有满足条件且不重复的三元组。

注意：答案中不可以包含重复的三元组。

示例：

Related Topics

数组

双指针

1.15.2 题目代码

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {

    }
};
```

1.15.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.16 [0016] 最接近的三数之和

- https://leetcode-cn.com/problems/3sum-closest

1.16.1 题目描述

给定一个包括 n 个整数的数组 nums 和一个目标值 target。找出 nums 中的三个整数，使得它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在唯一答案。

Related Topics

数组

双指针

1.16.2 题目代码

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {

    }
};
```

1.16.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.17 [0017] 电话号码的字母组合

- https://leetcode-cn.com/problems/letter-combinations-of-a-phone-number

1.17.1 题目描述

给定一个仅包含数字 2-9 的字符串，返回所有它能表示的字母组合。

给出数字到字母的映射如下（与电话按键相同）。注意 1 不对应任何字母。

示例:

说明: 尽管上面的答案是按字典序排列的，但是你可以任意选择答案输出的顺序。

Related Topics

字符串

回溯算法

1.17.2 题目代码

```
class Solution {
public:
    vector<string> letterCombinations(string digits) {

    }
};
```

1.17.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.18 [0018] 四数之和

- https://leetcode-cn.com/problems/4sum

1.18.1 题目描述

给定一个包含 n 个整数的数组 nums 和一个目标值 target，判断 nums 中是否存在四个元素 a，b，c 和 d ，使得 a + b + c + d 的值与 target 相等？找出所有满足条件且不重复的四元组。

注意：

答案中不可以包含重复的四元组。

示例：

Related Topics

数组

哈希表

双指针

1.18.2 题目代码

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {

    }
};
```

1.18.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.19 [0019] 删除链表的倒数第 N 个节点

- https://leetcode-cn.com/problems/remove-nth-node-from-end-of-list

1.19.1 题目描述

给定一个链表，删除链表的倒数第 n 个节点，并且返回链表的头结点。

示例:

说明:

给定的 n 保证是有效的。

进阶:

你能尝试使用一趟扫描实现吗?

Related Topics

链表

双指针

1.19.2 题目代码

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
```

(continues on next page)

(continued from previous page)

```
    }
};
```

1.19.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

1.20 [0020] 有效的括号

- https://leetcode-cn.com/problems/valid-parentheses

1.20.1 题目描述

给定一个只包括 '(',')','{','}','[',']' 的字符串，判断字符串是否有效。

有效字符串需满足：

```
<li> 左括号必须用相同类型的右括号闭合。</li>
<li> 左括号必须以正确的顺序闭合。</li>
```

注意空字符串可被认为是有效字符串。

示例 1:

示例 2:

示例 3:

示例 4:

示例 5:

Related Topics

栈

字符串

1.20.2 题目代码

```
class Solution {
public:
    bool isValid(string s) {

    }
};
```

1.20.3 题目解析

方法一

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

方法二

分析

思路

注意

知识点

复杂度

参考

答案

```
//
```

CHAPTER 2

专题

2.1 专题-BFS 算法

2.1.1 引言

2.1.2 例子

2.2 专题-DFS 算法

2.2.1 引言

2.2.2 例子

2.3 专题-DP 算法

2.3.1 引言

2.3.2 例子

2.4 专题-STL 算法

2.4.1 引言

2.4.2 例子

2.5 专题-二分算法

2.5.1 引言

2.5.2 例子

2.6 专题-二叉树

2.6.1 引言

2.6.2 例子

2.7 专题-回溯算法

2.7.1 引言

2.7.2 例子

2.8 专题-图论

2.8.1 引言

2.8.2 例子

2.9 专题-堆

2.9.1 引言

2.9.2 例子

2.10 专题-字符串

2.10.1 引言

2.10.2 例子

2.11 专题-排序算法

2.11.1 引言

2.11.2 例子

2.12 专题-搜索算法

2.12.1 引言

2.12.2 例子

2.13 专题-数学

2.13.1 引言

2.13.2 例子

2.14 专题-数组

2.14.1 引言

2.14.2 例子

2.15 专题-数论

2.15.1 引言

2.15.2 例子

2.16 专题-枚举算法

2.16.1 引言

2.16.2 例子

2.17 专题-查找算法

2.17.1 引言

2.17.2 例子

2.18 专题-栈

2.18.1 引言

2.18.2 例子

2.19 专题-模拟算法

2.19.1 引言

2.19.2 例子

2.20 专题-贪心算法

2.20.1 引言

2.20.2 例子

2.21 专题-递归算法

2.21.1 引言

2.21.2 例子

2.22 专题-链表

2.22.1 引言

2.22.2 例子

2.23 专题-队列

2.23.1 引言

2.23.2 例子

2.24 专题-随机化

2.24.1 引言

2.24.2 例子