**Leetcode 刷题汇总**

1. **题目：**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。

例如, 给定数组 nums = [-1, 0, 1, 2, -1, -4]，

满足要求的三元组集合为：

[

[-1, 0, 1],

[-1, -1, 2]

]

**实现代码（C++）：**

|  |
| --- |
| class Solution {  public:  vector<vector<int>> threeSum(vector<int>& nums) {  sort(nums.begin(), nums.end());  int len = nums.size();  vector<vector<int> > res;  for (int i = 0; i < len - 2; ++i) {  if (nums[i] > 0) break;  if (i > 0 && nums[i] == nums[i - 1]) continue;  int left = i + 1;  int right = len - 1;  while (left < right) {  int sum = nums[i] + nums[left] + nums[right];  if (sum > 0) {  --right;  } else if (sum < 0) {  ++left;  } else {  res.push\_back({nums[i], nums[left], nums[right]});  while (left < right && nums[left] == nums[++left]);  while (left < right && nums[right] == nums[--right]);  }  }  }  return res;  }  }; |

1. **题目：**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:

输入: 123

输出: 321

 示例 2:

输入: -123

输出: -321

示例 3:

输入: 120

输出: 21

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。

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  int reverse(int x) {  long int res=0;  while(x!=0){  int temp=x%10;//获得个位数  x=x/10;  res=res\*10+temp;  }  if(res>INT\_MAX||res<INT\_MIN) return 0;  return res;  }  }; |

1. **题目：**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:

输入: 121

输出: true

示例 2:

输入: -121

输出: false

解释: 从左向右读, 为 -121 。 从右向左读, 为 121- 。因此它不是一个回文数。

示例 3:

输入: 10

输出: false

解释: 从右向左读, 为 01 。因此它不是一个回文数。

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  bool isPalindrome(int x) {  bool isH=false;  if(x<0){  isH=false;  }  else if(x==0){  isH=true;  }else{  if(x%10==0){  isH=false;  }  else{  //将整数分离保存到数组中，然后与其逆序数组作比较  vector<int> originalNum;  int leen=0;  while(x!=0){  int temp=x%10;  x = x/10;  originalNum.push\_back(temp);  leen++;  }  vector<int> arrayx;  for(int i=leen-1;i>-1;i--){  arrayx.push\_back(originalNum[i]);  }  int flag=0;  for(int i=0;i<leen;i++){  if(originalNum[i]==arrayx[i]){  flag++;  }  }  if(flag==leen) isH=true;  }  }  return isH;  }  }; |

1. **题目：**

罗马数字包含以下七种字符: I， V， X， L，C，D 和 M。

字符 数值

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

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：

I 可以放在 V (5) 和 X (10) 的左边，来表示 4 和 9。

X 可以放在 L (50) 和 C (100) 的左边，来表示 40 和 90。

C 可以放在 D (500) 和 M (1000) 的左边，来表示 400 和 900。

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:

输入: "III"

输出: 3

示例 2:

输入: "IV"

输出: 4

示例 3:

输入: "IX"

输出: 9

示例 4:

输入: "LVIII"

输出: 58

解释: L = 50, V= 5, III = 3.

示例 5:

输入: "MCMXCIV"

输出: 1994

解释: M = 1000, CM = 900, XC = 90, IV = 4.

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  int getNum(char a){  int num=0;  switch(a){  case 'I':  num=1;  break;  case 'V':  num=5;  break;  case 'X':  num=10;  break;  case 'L':  num=50;  break;  case 'C':  num=100;  break;  case 'D':  num=500;  break;  case 'M':  num=1000;  break;  default:  break;  }  return num;  }  int romanToInt(string s) {  int resNum=0;  int temNUm=0;  for(int i=0;i<s.length();i++){  if(i+1<s.length()){  int ni=getNum(s[i]),ni1=getNum(s[i+1]);  if(ni>ni1){ //遇到当前值比下一个值大的情况就截断处理  if(i==0){  resNum +=ni;  }  else{  int \_1ni=getNum(s[i-1]);//上一个值  if(\_1ni>ni||\_1ni==ni){  resNum += ni;  }  else{  resNum += temNUm;  temNUm =0;  }  }  }  else if(ni==ni1){  resNum += ni;  }  else{  temNUm = ni1-ni;  }  }  else{ //处理最后一个字符  if(temNUm>0){//说明最后一个字符比上一个字符大  resNum += temNUm;  }  else { //否则将最后一个字符值直接放进数组  int ni=getNum(s[i]);  resNum += ni;  }  }  }  return resNum;  }  }; |

1. **题目：**

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:

输入: ["flower","flow","flight"]

输出: "fl"

示例 2:

输入: ["dog","racecar","car"]

输出: ""

解释: 输入不存在公共前缀。

说明:所有输入只包含小写字母 a-z 。

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  string longestCommonPrefix(vector<string>& strs) {  string resStr="";  if(strs.empty()) return resStr;  if(strs.size()==1){  resStr += strs[0];  return resStr;  }  int charCount=0;  int arrsize=strs.at(0).size();  for(int i=1;i<strs.size();i++)//求得组中最短字符串的长度  {  if(arrsize>strs.at(i).size())  arrsize=strs.at(i).size();  }  for(int k=0,zm=0,jn=0;k<strs.size()-1;k++)  {  while(zm<arrsize&&strs[k][zm]==strs[k+1][zm])  {  jn++;  zm++;  if(zm>=arrsize)  break;  }  if(k==0)  charCount=jn;  else if(jn<charCount)  charCount=jn;  zm=0;  jn=0;  }  for(int j=0;j<charCount;j++)  resStr += strs[0][j];  return resStr;  }  }; |

1. **题目：**

给定一个由整数组成的非空数组所表示的非负整数，在该数的基础上加一。

最高位数字存放在数组的首位， 数组中每个元素只存储单个数字。

你可以假设除了整数 0 之外，这个整数不会以零开头。

示例 1:

输入: [1,2,3]

输出: [1,2,4]

解释: 输入数组表示数字 123。

示例 2:

输入: [4,3,2,1]

输出: [4,3,2,2]

解释: 输入数组表示数字 4321。

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  vector<int> plusOne(vector<int>& digits) {  vector<int> result=digits;  int arraylen = digits.size();  if(arraylen==1){//数组长度为1的情况  if(digits.at(0)<9){  result.at(0) =digits.at(0)+1;  }  else{  result.at(0)=1;  result.push\_back(0);  }  }  else{  int last = digits.at(arraylen-1);  int flag=0,i=1;  while(last==9){//处理末位为9的情况  flag=1;  result.at(arraylen-i)=0;  i++;  if(i>arraylen){//当首位也为9时，如下处理：  result.at(0)=1;  for(int j=1;j<arraylen;j++){  result.at(j)=0;  }  result.push\_back(0);  break;  }  result.at(arraylen-i)=digits.at(arraylen-i)+1;  last=digits.at(arraylen-i);  }  if(flag==0){//处理末位不为9的情况：末位直接加1即可  result.at(arraylen-1)=last+1;  }  }  return result;  }  }; |

1. **题目：**

给定一个字符串，找到它的第一个不重复的字符，并返回它的索引。如果不存在，则返回 -1。

案例:

s = "leetcode"

返回 0.

s = "loveleetcode",

返回 2.

注意事项：您可以假定该字符串只包含小写字母。

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  bool isin(char gpcode,vector<char> vec){  vector<char>::iterator iter;  iter = find(vec.begin(), vec.end(), gpcode);  if(iter != vec.end())  return true;  else  return false;  }  int firstUniqChar(string s) {  int strlen=s.length();  int resindex=0;  vector<char> vec;  if(strlen==0) resindex=-1;//当字符串为空时  else if(strlen==1) resindex=0;//当字符串长度为1时  else {  for(int i=resindex+1;i<strlen;i++){  if(vec.size()>0&&isin(s[resindex],vec)){  resindex++;  i=resindex;  }  else if(s[resindex]==s[i]) {  if(vec.empty()) vec.push\_back(s[resindex]);  else{  if(!isin(s[resindex],vec)) vec.push\_back(s[resindex]);  }  resindex++;  i=resindex;  }  else  continue;  }  if(isin(s[resindex],vec))  if(resindex+1==strlen) resindex=-1;//字符串全是相同字符  }  return resindex;  }  }; |

1. **题目：**

给定一个字符串，验证它是否是回文串，只考虑字母和数字字符，可以忽略字母的大小写。

说明：本题中，我们将空字符串定义为有效的回文串。

示例 1:

输入: "A man, a plan, a canal: Panama"

输出: true

示例 2:

输入: "race a car"

输出: false

**实现代码（C++）:**

|  |
| --- |
| class Solution {  public:  bool isPalindrome(string s) {  if (s == "") return true;  if (s.length() == 1) return true;  bool result = false;  //去除字符串中的标点符号和空格  string charstr = "";  for (int i = 0; i < s.size(); i++)  {  if (!ispunct(s[i])&& s[i]!=' ')  charstr += tolower(s[i]);//全部换成小写  }  //开始判断  int strlen = charstr.size(), flagcount = 0;  int l = 0, r = strlen - 1;  while (l < r) {  if (charstr[l] == charstr[r]) {  l++;  r--;  flagcount++;  }  else {  result = false;  break;  }  }  if (flagcount == strlen / 2) result = true;  return result;  }  }; |

1. **题目：**