Leetcode(c++) 二分查找 简单题

35搜索插入位置

题目：

给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。

你可以假设数组中无重复元素。

示例 1:

输入: [1,3,5,6], 5

输出: 2

示例 2:

输入: [1,3,5,6], 2

输出: 1

示例 3:

输入: [1,3,5,6], 7

输出: 4

示例 4:

输入: [1,3,5,6], 0

输出: 0

思路：二分查找

代码：

1. **class** Solution {
2. **public**:
3. **int** searchInsert(vector<**int**>& nums, **int** target) {
4. **int** left=0;
5. **int** right=nums.size()-1;
6. **int** mid;
7. **if**(target>nums[right]) **return** nums.size();
8. **else** **if**(target<nums[0]) **return** 0;
9. **else** **if**(right==0) **return** 0;
10. **else**{
11. **while**(left<=right){
12. mid=(left+right)/2;
13. **if**(nums[mid]==target) **return** mid;
14. **else** **if**(nums[mid]<target) left=mid+1;
15. **else** **if**(nums[mid]>target) right=mid-1;
16. }
17. **return** left;
18. }
19. }
20. };

69 x的平方根

题目：

实现 int sqrt(int x) 函数。

计算并返回 x 的平方根，其中 x 是非负整数。

由于返回类型是整数，结果只保留整数的部分，小数部分将被舍去。

示例 1:

输入: 4

输出: 2

示例 2:

输入: 8

输出: 2

说明: 8 的平方根是 2.82842...,

由于返回类型是整数，小数部分将被舍去。

思路：二分查找

代码：

1. **class** Solution {
2. **public**:
3. **int** mySqrt(**int** x) {
4. **if**(x<1) **return** x;
5. **int** left=0,right=x;
7. **while**(left<=right){
8. **int** mid=(left+right)/2;
9. **long** **int** m=(**long**)mid\*mid;
10. **if**(m<x) left=mid+1;
11. **else** **if**(m>x) right=mid-1;
12. **else** **if**(m==x) **return** mid;
13. }
14. **return** left-1;
15. }
16. };

167两数之和II-输入有序数组

题目：

给定一个已按照升序排列 的有序数组，找到两个数使得它们相加之和等于目标数。

函数应该返回这两个下标值 index1 和 index2，其中 index1 必须小于 index2。

说明:

返回的下标值（index1 和 index2）不是从零开始的。

你可以假设每个输入只对应唯一的答案，而且你不可以重复使用相同的元素。

示例:

输入: numbers = [2, 7, 11, 15], target = 9

输出: [1,2]

解释: 2 与 7 之和等于目标数 9 。因此 index1 = 1, index2 = 2 。

思路：

代码：

1. **class** Solution {
2. **public**:
3. vector<**int**> twoSum(vector<**int**>& numbers, **int** target) {
4. **int** start=0;
5. **int** end=numbers.size()-1;
6. vector<**int**> ans;
7. **while**(start<end){
8. **if**(numbers[start]+numbers[end]==target){
9. ans.push\_back(start+1);
10. ans.push\_back(end+1);
11. **break**;
12. }**else** **if**(numbers[start]+numbers[end]>target){
13. end--;
14. }**else** **if**(numbers[start]+numbers[end]<target){
15. start++;
16. }
17. }
18. **return** ans;

21. }
22. };

278第一个错误的版本

题目：

你是产品经理，目前正在带领一个团队开发新的产品。不幸的是，你的产品的最新版本没有通过质量检测。由于每个版本都是基于之前的版本开发的，所以错误的版本之后的所有版本都是错的。

假设你有 n 个版本 [1, 2, ..., n]，你想找出导致之后所有版本出错的第一个错误的版本。

你可以通过调用 bool isBadVersion(version) 接口来判断版本号 version 是否在单元测试中出错。实现一个函数来查找第一个错误的版本。你应该尽量减少对调用 API 的次数。

示例:

给定 n = 5，并且 version = 4 是第一个错误的版本。

调用 isBadVersion(3) -> false

调用 isBadVersion(5) -> true

调用 isBadVersion(4) -> true

所以，4 是第一个错误的版本。

思路：

代码：

1. // Forward declaration of isBadVersion API.
2. **bool** isBadVersion(**int** version);
4. **class** Solution {
5. **public**:
6. **int** firstBadVersion(**int** n) {
7. **int** low=1;
8. **int** high=n;
9. **while**(low<=high){
10. **int** mid=low+(high-low)/2;
11. **if**(isBadVersion(mid)){
12. high=mid-1;
13. }
14. **else**{
15. low=mid+1;
16. }
17. }
18. **return** low;
19. }
20. };

349两个数组的交集

350两个数组的交集II

367有效的完全平方数

题目：

给定一个正整数 num，编写一个函数，如果 num 是一个完全平方数，则返回 True，否则返回 False。

说明：不要使用任何内置的库函数，如 sqrt。

示例 1：

输入：16

输出：True

示例 2：

输入：14

输出：False

思路：

代码：

1. **class** Solution {
2. **public**:
3. **bool** isPerfectSquare(**int** num) {
4. **long** left=1,right=num;
5. **while**(left<=right){
6. **long** mid=(left+right)/2;
7. **if** (mid\*mid==num) **return** **true**;
8. **else** **if**(mid\*mid>num) right=mid-1;
9. **else** left=mid+1;
10. }
11. **return** **false**;
12. }
13. };

374猜数字大小

题目：

我们正在玩一个猜数字游戏。 游戏规则如下：

我从 1 到 n 选择一个数字。 你需要猜我选择了哪个数字。

每次你猜错了，我会告诉你这个数字是大了还是小了。

你调用一个预先定义好的接口 guess(int num)，它会返回 3 个可能的结果（-1，1 或 0）：

-1 : 我的数字比较小

1 : 我的数字比较大

0 : 恭喜！你猜对了！

示例 :

输入: n = 10, pick = 6

输出: 6

思路：

代码：

1. // Forward declaration of guess API.
2. // @param num, your guess
3. // @**return** -1 **if** my number **is** lower, 1 **if** my number **is** higher, otherwise **return** 0
4. int guess(int num);
6. **class** Solution {
7. public:
8. int guessNumber(int n) {
9. int left=1,right=n;
10. int center;
11. **while**(left<right){
12. center=left+(right-left)/2;
13. **if**(guess(center)==1)
14. left=center+1;
15. **else** **if**(guess(center)==-1)
16. right=center;
17. **else**
18. **return** center;
19. }
20. **return** n;
22. }
23. };

441排列硬币

题目：

你总共有 n 枚硬币，你需要将它们摆成一个阶梯形状，第 k 行就必须正好有 k 枚硬币。

给定一个数字 n，找出可形成完整阶梯行的总行数。

n 是一个非负整数，并且在32位有符号整型的范围内。

示例 1:

n = 5

硬币可排列成以下几行:

¤

¤ ¤

¤ ¤

因为第三行不完整，所以返回2.

示例 2:

n = 8

硬币可排列成以下几行:

¤

¤ ¤

¤ ¤ ¤

¤ ¤

因为第四行不完整，所以返回3.

思路：

代码：

1. **class** Solution {
2. **public**:
3. **int** arrangeCoins(**int** n) {
4. **int** i=1;
5. **while**(n>0)
6. {
7. n=n-i;
8. i++;
9. }
10. **if**(n<0)
11. **return** i-2;
12. **else**
13. **return** i-1;
14. }
15. };

475供暖器

题目：

冬季已经来临。 你的任务是设计一个有固定加热半径的供暖器向所有房屋供暖。

现在，给出位于一条水平线上的房屋和供暖器的位置，找到可以覆盖所有房屋的最小加热半径。

所以，你的输入将会是房屋和供暖器的位置。你将输出供暖器的最小加热半径。

说明:

给出的房屋和供暖器的数目是非负数且不会超过 25000。

给出的房屋和供暖器的位置均是非负数且不会超过10^9。

只要房屋位于供暖器的半径内(包括在边缘上)，它就可以得到供暖。

所有供暖器都遵循你的半径标准，加热的半径也一样。

示例 1:

输入: [1,2,3],[2]

输出: 1

解释: 仅在位置2上有一个供暖器。如果我们将加热半径设为1，那么所有房屋就都能得到供暖。

示例 2:

输入: [1,2,3,4],[1,4]

输出: 1

解释: 在位置1, 4上有两个供暖器。我们需要将加热半径设为1，这样所有房屋就都能得到供暖。

思路：

代码：

704二分查找

题目：

给定一个 n 个元素有序的（升序）整型数组 nums 和一个目标值 target ，写一个函数搜索 nums 中的 target，如果目标值存在返回下标，否则返回 -1。

示例 1:

输入: nums = [-1,0,3,5,9,12], target = 9

输出: 4

解释: 9 出现在 nums 中并且下标为 4

示例 2:

输入: nums = [-1,0,3,5,9,12], target = 2

输出: -1

解释: 2 不存在 nums 中因此返回 -1

思路：

代码：

1. **class** Solution {
2. **public**:
3. **int** search(vector<**int**>& nums, **int** target) {
4. **int** right=nums.size()-1;
5. **int** left=0;
6. **int** mid=0;
7. **while**(left<=right){
8. **if**(nums[left]==target) {
9. **return** left;
10. }
11. **if**(nums[right]==target) {
12. **return** right;
13. }
14. mid=(right+left)/2;
15. **if**(nums[mid]==target){
16. **return** mid;
17. }
18. **else** **if**(nums[mid]>target){
19. left++;
20. right=mid-1;
21. }
22. **else**{
23. right--;
24. left=mid+1;
25. }
26. }
27. **return** -1;
28. }
29. };

744寻找比目标字母大的最小字母

题目：

给定一个只包含小写字母的有序数组letters 和一个目标字母 target，寻找有序数组里面比目标字母大的最小字母。

数组里字母的顺序是循环的。举个例子，如果目标字母target = 'z' 并且有序数组为 letters = ['a', 'b']，则答案返回 'a'。

示例:

输入:

letters = ["c", "f", "j"]

target = "a"

输出: "c"

输入:

letters = ["c", "f", "j"]

target = "c"

输出: "f"

输入:

letters = ["c", "f", "j"]

target = "d"

输出: "f"

输入:

letters = ["c", "f", "j"]

target = "g"

输出: "j"

输入:

letters = ["c", "f", "j"]

target = "j"

输出: "c"

输入:

letters = ["c", "f", "j"]

target = "k"

输出: "c"

注:

letters长度范围在[2, 10000]区间内。

letters 仅由小写字母组成，最少包含两个不同的字母。

目标字母target 是一个小写字母。

思路：利用二分查找， 最后判断low的位置，如果大于数组长度就输出第一个元素，否则直接输出第low个元素

代码：

1. **class** Solution {
2. **public**:
3. **char** nextGreatestLetter(vector<**char**>& letters, **char** target) {
4. **int** low=0;
5. **int** height=letters.size()-1;
6. **int** mid;
7. **while**(low<=height){
8. mid=(low+height)/2;
9. **if**(letters[mid]>target){
10. height=mid-1;
11. }**else**{
12. low=mid+1;
13. }
14. }
15. **return** low>=letters.size()? letters[0]:letters[low];
16. }
17. };

852山脉数组的封顶索引

题目：

我们把符合下列属性的数组 A 称作山脉：

A.length >= 3

存在 0 < i < A.length - 1 使得A[0] < A[1] < ... A[i-1] < A[i] > A[i+1] > ... > A[A.length - 1]

给定一个确定为山脉的数组，返回任何满足 A[0] < A[1] < ... A[i-1] < A[i] > A[i+1] > ... > A[A.length - 1] 的 i 的值。

示例 1：

输入：[0,1,0]

输出：1

示例 2：

输入：[0,2,1,0]

输出：1

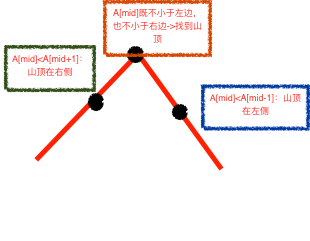
提示：

3 <= A.length <= 10000

0 <= A[i] <= 10^6

A 是如上定义的山脉

思路：



代码：

1. **class** Solution {
2. **public**:
3. **int** peakIndexInMountainArray(vector<**int**>& A) {
4. **int** low=0;
5. **int** high=A.size()-1;
6. **int** mid=0;
7. **while**(low<=high){
8. mid = low + ((high-low)>>2);
9. **if**(A[mid]<A[mid+1])
10. low=mid+1;
11. **else** **if**(A[mid]<A[mid-1])
12. high=mid-1;
13. **else**
14. **return** mid;
15. }
16. }
17. };

中等题

29两数相除

题目：

给定两个整数，被除数 dividend 和除数 divisor。将两数相除，要求不使用乘法、除法和 mod 运算符。

返回被除数 dividend 除以除数 divisor 得到的商。

示例 1:

输入: dividend = 10, divisor = 3

输出: 3

示例 2:

输入: dividend = 7, divisor = -3

输出: -2

说明:

被除数和除数均为 32 位有符号整数。

除数不为 0。

假设我们的环境只能存储 32 位有符号整数，其数值范围是 [−231, 231 − 1]。本题中，如果除法结果溢出，则返回 231 − 1。

思路：

可以利用位操作(Bit Operator)，即二进制相关的形式来实现。操作符 "<<" 和 ">>" 分别为将数整体左移和右移，例如a<<1表示将a向左移动一位，即变为原来的二倍。

思路:思考了一下，当被除数大于等于除数时(否则的话就为0了)，我们设置两个变量t和p，并分别初始化为除数和1(最小的情况)，当被除数大于等于t的二倍时，将t和p同时扩大二倍(左移)，并将返回值加上p，除数减去t。拿十进制举例:29除以8，8扩大二倍，16小于29，再扩大二倍，超过29，于是29减去之前的16，返回值加上2。第二次循环时因为此时的13小于8的二倍，故加上1，整个循环结束，最终结果为2+1=3，符合要求。此外还要注意判断结果正负号时亦或的作用。

代码：

1. **class** Solution {
2. **public**:
3. **int** divide(**int** dividend, **int** divisor) {
4. **if** (divisor == 0 || (dividend == INT\_MIN && divisor == -1)) **return** INT\_MAX;
5. **long** **long** m=abs((**long** **long**)dividend),n=abs((**long** **long**)divisor),res=0;
6. **int** sign=((dividend<0)^(divisor<0))? -1:1;
7. **if** (n==1) **return** sign==1?m:-m;
8. **while**(m>=n){
9. **long** **long** t=n,p=1;
10. **while**(m>=(t<<1)){
11. p<<=1;
12. t<<=1;
13. }
14. res+=p;
15. m-=t;
16. }
17. **return** sign>0? res:-res;

20. }
21. };

33搜索旋转排序数组

题目：

假设按照升序排序的数组在预先未知的某个点上进行了旋转。

( 例如，数组 [0,1,2,4,5,6,7] 可能变为 [4,5,6,7,0,1,2] )。

搜索一个给定的目标值，如果数组中存在这个目标值，则返回它的索引，否则返回 -1 。

你可以假设数组中不存在重复的元素。

你的算法时间复杂度必须是 O(log n) 级别。

示例 1:

输入: nums = [4,5,6,7,0,1,2], target = 0

输出: 4

示例 2:

输入: nums = [4,5,6,7,0,1,2], target = 3

输出: -1

思路：每次把数组分成两半，无论中点取在什么地方，必然有一般是已经排序的，或者中点左边的序列，或者中点右边的序列，甚至两边的序列。如此一来，判断数组是在哪一边就可以了，基本程序写好，每次相当于在递归的计算

代码：

1. **class** Solution {
2. **public**:
3. **int** search(vector<**int**>& nums, **int** target) {
4. **if**(nums.empty())
5. {
6. **return** -1;
7. }
8. **int** left=0, right=nums.size()-1,mid=0;
9. **while**(left<=right)
10. {
11. mid=(left+right)/2;
12. **if**(nums[mid]==target)
13. {
14. **return** mid;
15. }
16. **if**(nums[mid]>=nums[left])//nums[mid]在前半段
17. {
18. **if**(target<nums[mid] && target>=nums[left])//满足此条件，说明targte可能在mid之前,即，target是处于排过序的那一段
19. {
20. right=mid-1;
21. }
22. **else**
23. {
24. left=mid+1;
25. }
26. }
27. **else**//nums[mid]在后半段
28. {
29. **if**(target>nums[mid] && target<=nums[right])
30. {
31. left=mid+1;
32. }
33. **else**
34. {
35. right=right-1;
36. }
37. }
38. }
39. **return** -1;
41. }
43. };