Leetcode(python) 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(object):
2. **def** searchInsert(self, nums, target):
3. """
4. :type nums: List[int]
5. :type target: int
6. :rtype: int
7. """
8. i=0
9. length=len(nums)
10. **while** i<length:
11. mid=i+(length-i)/2
12. **if** nums[mid]>target:
13. length=mid
14. **elif** nums[mid]<target:
15. i=mid+1
16. **else**:
17. **return** mid
18. **return** i

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(object):
2. **def** mySqrt(self, x):
3. """
4. :type x: int
5. :rtype: int
6. """
7. left=0
8. right=x
9. **while** left<=right:
10. mid=(left+right)/2
11. v=mid\*mid
12. **if** v<x:
13. left=mid+1
14. **elif** v>x:
15. right=mid-1
16. **else**:
17. **return** mid
18. **return** right

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(object):
2. **def** twoSum(self, numbers, target):
3. """
4. :type numbers: List[int]
5. :type target: int
6. :rtype: List[int]
7. """
8. left=0
9. right=len(numbers)-1
10. **while** left<right:
11. **if** numbers[left]+numbers[right]==target:
12. **return** [left+1,right+1]
13. **elif** numbers[left]+numbers[right]>target:
14. right-=1
15. **else**:
16. left+=1
17. **return** []

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. # The isBadVersion API is already defined for you.
2. # @param version, an integer
3. # @return a bool
4. # def isBadVersion(version):
6. **class** Solution(object):
7. **def** firstBadVersion(self, n):
8. """
9. :type n: int
10. :rtype: int
11. """
12. left=0;right=n
13. **while**(True):
14. mid=(left+right)//2
15. **if** isBadVersion(mid)==False **and** isBadVersion(mid+1)==True:
16. **return** mid+1
17. **elif** isBadVersion(mid)==False **and** isBadVersion(mid+1)==False:
18. left=mid
19. **elif** isBadVersion(mid)==True **and** isBadVersion(mid+1)==True:
20. right=mid

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(object):
2. **def** isPerfectSquare(self, num):
3. """
4. :type num: int
5. :rtype: bool
6. """
7. left=1
8. right=num
9. **while** left<=right:
10. mid=(left+right)//2
11. t=mid\*\*2
12. **if** t==num: **return** True
13. **elif** t>num: right=mid-1
14. **else**: left=mid+1
16. **return** False

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. # The guess API is already defined for you.
2. # @param num, your guess
3. # @return -1 if my number is lower, 1 if my number is higher, otherwise return 0
4. # def guess(num):
6. **class** Solution(object):
7. **def** guessNumber(self, n):
8. """
9. :type n: int
10. :rtype: int
11. """
12. l = 1
13. h = n
14. **while** l <= h :
15. mid = (l+h)//2
16. **if** **not** guess(mid):
17. **return** mid
18. **elif** guess(mid) == -1:
19. h = mid - 1
20. **else**:
21. l = mid + 1

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(object):
2. **def** arrangeCoins(self, n):
3. """
4. :type n: int
5. :rtype: int
6. """
7. left=0
8. right=n
9. **while** left<=right:
10. mid=(left+right)//2
11. **if** 0 <= n - (mid+1)\*mid/2 < mid + 1:
12. **return** mid
13. **elif** n - (mid+1)\*mid/2 >= mid + 1:
14. low = mid + 1
15. **elif** n - (mid+1)\*mid/2 < 0:
16. high=mid-1

475供暖器

题目：

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(object):
2. **def** search(self, nums, target):
3. """
4. :type nums: List[int]
5. :type target: int
6. :rtype: int
7. """
8. left=0
9. right=len(nums)-1
10. **while** left<=right:
11. mid=(left+right)//2
12. **if** nums[mid]==target:
13. **return** mid
14. **elif** nums[mid]>target:
15. right=mid-1
16. **else**:
17. left=mid+1
18. **return** -1

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：

代码：

1. **class** Solution(object):
2. **def** nextGreatestLetter(self, letters, target):
3. """
4. :type letters: List[str]
5. :type target: str
6. :rtype: str
7. """
8. l=0
9. r=len(letters)-1
10. **while** l<=r:
11. m=l+(r-l)/2
12. **if** letters[m]<=target:
13. l=m+1
14. **else**:
15. r=m-1
16. **if** l<len(letters):
17. **return** letters[l]
18. **else**:
19. **return** letters[0]

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(object):
2. **def** peakIndexInMountainArray(self, A):
3. """
4. :type A: List[int]
5. :rtype: int
6. """
7. l=0
8. r=len(A)-1
9. **while** l<=r:
10. mid=l+(r-l)/2
11. **if** A[mid]>A[mid-1] **and** A[mid]>A[mid+1]:
12. **return** mid
13. **elif** A[mid]<A[mid+1]:
14. l=mid+1
15. **elif** A[mid]<A[mid-1]:
16. r=mid-1