王争的算法训练营

习题课：二分查找

习题课：二分查找

王争的算法训练营

配套习题：

704. 二分查找（简单） 标准二分查找

374. 猜数字大小（简单）

744. 寻找比目标字母大的最小字母（简单）

35. 搜索插入位置（简单）

34. 在排序数组中查找元素的第一个和最后一个位置 （中等）

面试题 10.05. 稀疏数组搜索（简单）

33. 搜索旋转排序数组（中等）无重复数据 **（已讲）**

153. 寻找旋转排序数组中的最小值（中等） 无重复数据**（已讲）**

852. 山脉数组的峰顶索引（简单）峰值二分**（已讲）**

162. 寻找峰值（中等）峰值二分

367. 有效的完全平方数（简单）二分答案

69. x 的平方根（简单）二分答案**（已讲）**

74. 搜索二维矩阵（中等） 二维转一维，二分查找

以下为选做：

658. 找到 K 个最接近的元素（中等）

875. 爱吃香蕉的珂珂（中等）二分答案

81. 搜索旋转排序数组 II（中等）有重复数据

154. 寻找旋转排序数组中的最小值 II （困难） 有重复数据

二分查找正确的编写姿势：

- 查找区间永远是闭区间[low, high]
- 循环条件永远是：low<=high
- 返回值永远是mid，而不要是low、high
- low、high的更新永远是low=mid+1和high=mid-1
- 对于非确定性查找，使用前后探测法，来确定搜索区间
- 先处理命中情况，再处理在左右半部分查找的情况

习题课：二分查找

王争的算法训练营

704. 二分查找（简单） 标准二分查找

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
```

习题课：二分查找

704. 二分查找（简单） 标准二分查找

```
class Solution {
    public int search(int[] nums, int target) {
        int low = 0;
        int high = nums.length-1;
        while (low <= high) {
            int mid = (low+high)/2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                low = mid+1;
            } else {
                high = mid-1;
            }
        }
        return -1;
    }
}
```

习题课：二分查找

王争的算法训练营

374. 猜数字大小（简单）

猜数字游戏的规则如下：

- 每轮游戏，我都会从 **1** 到 ***n*** 随机选择一个数字。 请你猜选出的是哪个数字。
- 如果你猜错了，我会告诉你，你猜测的数字比我选出的数字是大了还是小了。

你可以通过调用一个预先定义好的接口 `int guess(int num)` 来获取猜测结果，返回值一共有 3 种可能的情况（`-1`，`1` 或 `0`）：

- -1：我选出的数字比你猜的数字小 `pick < num`
- 1：我选出的数字比你猜的数字大 `pick > num`
- 0：我选出的数字和你猜的数字一样。恭喜！你猜对了！ `pick == num`

返回我选出的数字。

示例 1：

```
输入：n = 10, pick = 6
输出：6
```

示例 2：

```
输入：n = 1, pick = 1
输出：1
```

习题课：二分查找

374. 猜数字大小（简单）

```
public class Solution extends GuessGame {
    public int guessNumber(int n) {
        int low = 1;
        int high = n;
        while (low <= high) {
            int mid = low+(high-low)/2;
            int ret = guess(mid);
            if (ret == 0) {
                return mid;
            } else if (ret == -1) {
                high = mid-1;
            } else {
                low = mid+1;
            }
        }
        return -1;
    }
}
```

习题课：二分查找

744. 寻找比目标字母大的最小字母（简单）

给你一个排序后的字符列表 `letters` ，列表中只包含小写英文字母。另给出一个目标字母 `target` ，请你寻找在这一有序列表里比目标字母大的最小字母。

在比较时，字母是依序循环出现的。举个例子：

- 如果目标字母 `target = 'z'` 并且字符列表为 `letters = ['a', 'b']` ，则答案返回 `'a'`

查找第一个大于给定值的元素

示例：

```
输入:
letters = ["c", "f", "j"]
target = "a"
输出: "c"
```

习题课：二分查找

王争的算法训练营

744. 寻找比目标字母大的最小字母（简单）

```
class Solution {
    // 第一个大于target的元素
    public char nextGreatestLetter(char[] letters, char target) {
        int low = 0;
        int high = letters.length-1;
        while (low <= high) {
            int mid = low + (high-low)/2;
            char c = letters[mid];
            if (c > target) {
                if (mid == 0 || letters[mid-1] <= target) {
                    return letters[mid];
                } else {
                    high = mid-1;
                }
            } else {
                low = mid+1;
            }
        }
        return letters[0]; // 这个题目的特殊要求
    }
}
```

习题课：二分查找

王争的算法训练营

35. 搜索插入位置（简单）

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。

查找第一个大于等于给定值的元素位置

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
```

习题课：二分查找

王争的算法训练营

35. 搜索插入位置（简单）

```
class Solution {
    // 查找第一个大于等于target的位置
    public int searchInsert(int[] nums, int target) {
        int low = 0;
        int high = nums.length-1;
        while (low <= high) {
            int mid = low+(high-low)/2;
            if (nums[mid]>=target) {
                if (mid==0 || nums[mid-1]<target) {
                    return mid;
                } else {
                    high = mid-1;
                }
            } else {
                low = mid+1;
            }
        }
        return nums.length;
    }
}
```

习题课：二分查找

王争的算法训练营

34. 在排序数组中查找元素的第一个和最后一个位置 （中等）

给定一个按照升序排列的整数数组 `nums`，和一个目标值 `target`。找出给定目标值在数组中的开始位置和结束位置。

如果数组中不存在目标值 `target`，返回 `[-1, -1]`。

查找第一个值等于给定值的元素
查找最后一个值等于给定值的元素

进阶：

- 你可以设计并实现时间复杂度为 `O(log n)` 的算法解决此问题吗？

示例 1：

```
输入：nums = [5,7,7,8,8,10], target = 8
输出：[3,4]
```

示例 2：

```
输入：nums = [5,7,7,8,8,10], target = 6
输出：[-1,-1]
```

示例 3：

```
输入：nums = [], target = 0
输出：[-1,-1]
```

```java
class Solution {
    public int[] searchRange(int[] nums, int target) {
        int low = 0;
        int high = nums.length-1;
        int left = -1;
        while (low <= high) {
            int mid = (low+high)/2;
            if (nums[mid] == target) {
                if (mid == 0 || nums[mid-1] != target) {
                    left = mid;
                    break;
                } else {
                    high = mid-1;
                }
            } else if (nums[mid] > target) {
                high = mid-1;
            } else {
                low = mid+1;
            }
        }

        low = 0;
        high = nums.length-1;
        int right = -1;
        while (low <= high) {
            int mid = (low+high)/2;
            if (nums[mid] == target) {
                if (mid == nums.length-1 || nums[mid+1] != target) {
                    right = mid;
                    break;
                } else {
                    low = mid+1;
                }
            } else if (nums[mid] > target) {
                high = mid-1;
            } else {
                low = mid+1;
            }
        }

        return new int[]{left, right};
    }
}
```

习题课：二分查找

王争的算法训练营

[面试题 10.05. 稀疏数组搜索](#)（简单）

稀疏数组搜索。有个排好序的字符串数组，其中散布着一些空字符串，编写一种方法，找出给定字符串的位置。

示例1:

```
 输入: words = ["at", "", "", "", "ball", "", "", "car", "", "","dad",
"", ""], s = "ta"
 输出: -1
 说明: 不存在返回-1。
```

示例2:

```
 输入: words = ["at", "", "", "", "ball", "", "", "car", "", "","dad",
"", ""], s = "ball"
 输出: 4
```

提示:

1. words的长度在[1, 1000000]之间

习题课：二分查找

面试题 10.05. 稀疏数组搜索（简单）

```
class Solution {
    public int findString(String[] words, String s) {
        int low = 0;
        int high = words.length-1;
        while (low <= high) {
            int mid = (low+high)/2;
            if (words[mid].equals(s)) {
                return mid;
            } else if (words[mid].equals("")) {
                if (words[low].equals(s)) return low;
                else low++;
            } else if (words[mid].compareTo(s) < 0) {
                low = mid+1;
            } else {
                high = mid-1;
            }
        }
        return -1;
    }
}
```

习题课：二分查找

33. 搜索旋转排序数组（中等）无重复数据 **（已讲）**

整数数组 `nums` 按升序排列，数组中的值 **互不相同** 。

在传递给函数之前，`nums` 在预先未知的某个下标 `k`（`0 <= k < nums.length`）上进行了 **旋转**，使数组变为 `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]`（下标 **从 0 开始** 计数）。例如，`[0,1,2,4,5,6,7]` 在下标 `3` 处经旋转后可能变为 `[4,5,6,7,0,1,2]` 。

给你 **旋转后** 的数组 `nums` 和一个整数 `target` ，如果 `nums` 中存在这个目标值 `target` ，则返回它的下标，否则返回 `-1` 。

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
```

习题课：二分查找

王争的算法训练营

33. 搜索旋转排序数组（中等）无重复数据

```
class Solution {
    public int search(int[] nums, int target) {
        int left = 0;
        int right = nums.length-1;
        while (left <= right) {
            int mid = (left+right)/2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[left] <= nums[mid]) {//left side sorted
                if (target >= nums[left] && target < nums[mid]) {
                    right = mid-1;
                } else {
                    left = mid+1;
                }
            } else {
                if (target > nums[mid] && target <= nums[right]) {
                    left = mid+1;
                } else {
                    right = mid-1;
                }
            }
        }
        return -1;
    }
}
```

习题课：二分查找

王争的算法训练营

153. 寻找旋转排序数组中的最小值（中等） 无重复数据 **（已讲）**

已知一个长度为 `n` 的数组，预先按照升序排列，经由 `1` 到 `n` 次 **旋转** 后，得到输入数组。例如，原数组 `nums = [0,1,2,4,5,6,7]` 在变化后可能得到：

- 若旋转 `4` 次，则可以得到 `[4,5,6,7,0,1,2]`
- 若旋转 `7` 次，则可以得到 `[0,1,2,4,5,6,7]`

注意，数组 `[a[0], a[1], a[2], ..., a[n-1]]` **旋转一次** 的结果为数组 `[a[n-1], a[0], a[1], a[2], ..., a[n-2]]` 。

给你一个元素值 **互不相同** 的数组 `nums` ，它原来是一个升序排列的数组，并按上述情形进行了多次旋转。请你找出并返回数组中的 **最小元素** 。

示例 1：

输入： nums = [3,4,5,1,2]
输出： 1
解释： 原数组为 [1,2,3,4,5] ，旋转 3 次得到输入数组。

习题课：二分查找

王争的算法训练营

153. 寻找旋转排序数组中的最小值（中等） 无重复数据

```
class Solution {
    public int findMin(int[] nums) {
        int low = 0;
        int high = nums.length-1;
        while (low <= high) {
            int mid = (low+high)/2;
            // 特殊处理low==high的情况
            if (low == high) {
                return nums[mid];
            }
            //先处理命中情况
            if ((mid != 0 && nums[mid]<nums[mid-1])
              ||(mid == 0 && nums[mid]<nums[high])) {
                return nums[mid];
            } else if (nums[mid]>nums[high]) { // 右循环有序
                low = mid+1;
            } else { // 右侧非循环有序
                high = mid-1;
            }
        }
        return -1;//永远到不了这里
    }
}
```

习题课：二分查找

王争的算法训练营

852. 山脉数组的峰顶索引（简单）峰值二分（已讲）

符合下列属性的数组 `arr` 称为 **山脉数组**：

- `arr.length >= 3`
- 存在 `i`（`0 < i < arr.length - 1`）使得：
  - `arr[0] < arr[1] < ... arr[i-1] < arr[i]`
  - `arr[i] > arr[i+1] > ... > arr[arr.length - 1]`

给你由整数组成的山脉数组 `arr`，返回任何满足 `arr[0] < arr[1] < ... arr[i - 1] < arr[i] > arr[i + 1] > ... > arr[arr.length - 1]` 的下标 `i`。

示例 1：

```
输入：arr = [0,1,0]
输出：1
```

示例 2：

```
输入：arr = [0,2,1,0]
输出：1
```

习题课：二分查找

852. 山脉数组的峰顶索引（简单）峰值二分

```
class Solution {
    public int peakIndexInMountainArray(int[] arr) {
        int n = arr.length;
        int low = 0;
        int high = n-1;
        while (low <= high) {
            int mid = (low+high)/2;
            if (mid == 0) {
                low = mid+1;
            } else if (mid == n-1) {
                high = mid-1;
            } else if (arr[mid] > arr[mid-1] && arr[mid] > arr[mid+1]) {
                return mid;
            } else if (arr[mid] > arr[mid-1]) {
                low = mid+1;
            } else {
                high = mid-1;
            }
        }
        return -1;
    }
}
```

习题课：二分查找

王争的算法训练营

162. 寻找峰值（中等）峰值二分

峰值元素是指其值大于左右相邻值的元素。

给你一个输入数组 `nums`，找到峰值元素并返回其索引。数组可能包含多个峰值，在这种情况下，返回 **任何一个峰值** 所在位置即可。

你可以假设 `nums[-1] = nums[n] = -∞` 。

示例 1：

```
输入：nums = [1,2,3,1]
输出：2
解释：3 是峰值元素，你的函数应该返回其索引 2。
```

示例 2：

```
输入：nums = [1,2,1,3,5,6,4]
输出：1 或 5
解释：你的函数可以返回索引 1，其峰值元素为 2；
     或者返回索引 5， 其峰值元素为 6。
```

提示：

- `1 <= nums.length <= 1000`
- `-2^31 <= nums[i] <= 2^31 - 1`
- 对于所有有效的 `i` 都有 `nums[i] != nums[i + 1]`

```java
class Solution {
    public int findPeakElement(int[] nums) {
        int n = nums.length;
        int low = 0;
        int high = n-1;
        while (low <= high) {
            int mid = (low+high)/2;
            if (low == high) { //不添加这一行，测试用例[1]报错
                return mid;
            }
            if (mid == 0) {
                if (nums[mid]>nums[mid+1]) {
                    return mid;
                } else {
                    low = mid+1;
                }
            } else if (mid == n-1) {
                if (nums[mid]>nums[mid-1]) {
                    return mid;
                } else {
                    high = mid-1;
                }
            } else if (nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1]) {
                return mid;
            } else if (nums[mid]<nums[mid+1]) {
                low = mid+1;
            } else {
                high = mid-1;
            }
        }
        return -1;
    }
}
```

测试用例：

1）区间长度为0

2）区间长度为1

习题课：二分查找

69. x 的平方根（简单）二分答案 **（已讲）**

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。

习题课：二分查找

王争的算法训练营

```
class Solution {
    public int mySqrt(int x) {
        if (x == 0) return 0;
        // 从[1, x]中查找最后一个平方小于等于x的数
        int low = 1;
        int high = x/2+1;
        while (low <= high) {
            int mid = low + (high-low)/2;
            long mid2 = (long)mid*mid;
            if (mid2==x) {
                return mid;
            } else if (mid2<x) {
                long mid22 = ((long)mid+1)*(mid+1);
                if (mid22 <= x) {
                    low = mid+1;
                } else {
                    return mid;
                }
            } else {
                high = mid-1;
            }
        }
        return -1;
    }
}
```

习题课：二分查找

王争的算法训练营

367. 有效的完全平方数（简单）二分答案

367. 有效的完全平方数

难度 简单　211　收藏　分享　切换为英文　接收动态　反馈

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` 。

进阶：不要 使用任何内置的库函数，如 `sqrt` 。

示例 1：

```
输入：num = 16
输出：true
```

示例 2：

```
输入：num = 14
输出：false
```

提示：

- `1 <= num <= 2^31 - 1`

习题课：二分查找

367. 有效的完全平方数（简单）二分答案

```
class Solution {
    public boolean isPerfectSquare(int num) {
        // [1, num]之间选择平方等于num的数
        int low = 1;
        int high = num;
        while (low <= high) {
            int mid = low + (high-low)/2;
            // 1. mid^2==num  return true
            // 2. mid^2>num  high=mid-1
            // 3. mid^2<num  low=mid+1
            long mid2 = (long)mid*mid;
            if (mid2 == num) {
                return true;
            } else if (mid2 > num) {
                high = mid-1;
            } else {
                low = mid+1;
            }
        }
        return false;
    }
}
```

习题课：二分查找

王争的算法训练营

74. 搜索二维矩阵（中等） 二维转一维，二分查找

编写一个高效的算法来判断 `m x n` 矩阵中，是否存在一个目标值。该矩阵具有如下特性：

- 每行中的整数从左到右按升序排列。
- 每行的第一个整数大于前一行的最后一个整数。

示例 1：

1	3	5	7
10	11	16	20
23	30	34	60

```
输入：matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
输出：true
```

习题课：二分查找

74. 搜索二维矩阵（中等） 二维转一维，二分查找

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;
        int low = 0, high = m * n - 1;
        int mid, midValue;
        while (low <= high) {
            mid = (low + high) / 2;
            midValue  = matrix[mid/n][mid%n];
            if (target == midValue) {
                return true;
            } else if (target < midValue) {
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        return false;
    }
}
```

关注微信公众号“**小争哥**”，

后台回复“**PDF**”获取独家算法资料