I. Strstr

## 13. strStr ☆

Description | Notes | >_ Testcase | Judge

For a given source string and a target string, you should output the **first** index(from 0) of target string in source string.

If target does not exist in source, just return `-1`.

Have you met this question in a real interview? Yes

### Clarification

Do I need to implement KMP Algorithm in a real interview?

- Not necessary. When you meet this problem in a real interview, the interviewer may just want to test your basic implementation ability. But make sure you confirm with the interviewer first.

### Example

If source = `"source"` and target = `"target"`, return `-1`.

If source = `"abcdabcdefg"` and target = `"bcd"`, return `1`.

```
class Solution {
    /**
     * Returns a index to the first occurrence of target in source,
     * or -1  if target is not part of source.
     * @param source string to be scanned.
     * @param target string containing the sequence of characters to match.
     */
    public int strStr(String source, String target) {
        // write your code here
        if (source==null || target==null) {
            return -1;
        }
        if (target.length()==0) {
            return 0;
```

```
        }
        for (int i=0;i<source.length()-target.length()+1;i++) {
            for (int j=0;j<target.length();j++) {
                if (source.charAt(i+j)!=target.charAt(j)) {
                    break;
                }
                if (j==target.length()-1) {
                    return i;
                }
            }
        }
        return -1;
    }
}
```

II. Binary Search

## 459. Closest Number in Sorted Array ☆

Description | Notes | >_ Testcase | Judge

Given a target number and an integer array A sorted in ascending order, find the index `i` in A such that A[i] is closest to the given target.

Return -1 if there is no element in the array.

**Notice**

There can be duplicate elements in the array, and we can return any of the indices with same value.

Have you met this question in a real interview? Yes

**Example**

Given `[1, 2, 3]` and target = `2`, return `1`.

Given `[1, 4, 6]` and target = `3`, return `1`.

Given `[1, 4, 6]` and target = `5`, return `1` or `2`.

Given `[1, 3, 3, 4]` and target = `2`, return `0` or `1` or `2`.

```
public class Solution {
    /*
     * @param A: an integer array sorted in ascending order
     * @param target: An integer
     * @return: an integer
     */
    public int closestNumber(int[] A, int target) {
        // write your code here
        if (A == null || A.length == 0) {
            return -1;
```

```
        }
        int index = binarySearch(A, target);
        if (index == 0) {
            return 0;
        }
        if (Math.abs(A[index]-target) < Math.abs(A[index - 1]-target)) {
            return index;
        } else {
            return index - 1;
        }

    }
    private int binarySearch(int[] A, int target) {
        int start = 0;
        int end = A.length - 1;
        int mid;

        while (start + 1 < end) {
            mid = start + (end - start) / 2; // start = 0, end = 4, mid = 2
            // start = 0, end = 5, mid = 2.5 we get 2
            if (A[mid] < target) {
                start = mid;
            } else if (A[mid] > target) {
                end = mid;
            } else {
                return mid;
            }
        }
        if (A[start] >= target) { // target is out of the range of A. smaller than the smallest item.
        return start;

        }
        if (A[end] >= target) {
            return end; // return the number which just larger than or equal the target.

        }
        return A.length - 1; // target is out of the range of A. larger than the largest item.
    }

}
```

# 458. Last Position of Target ☆

Description | Notes | Testcase | Judge

Find the last position of a target number in a sorted array. Return -1 if target does not exist.

Have you met this question in a real interview? Yes

**Example**

Given `[1, 2, 2, 4, 5, 5]`.

For target = `2`, return 2.

For target = `5`, return 5.

For target = `6`, return -1.

**Tags**

```
public class Solution {
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return an integer
     */
    public int lastPosition(int[] nums, int target) {
        // Write your code here
        if (nums == null || nums.length == 0) {
            return -1;
        }
        int start = 0, end = nums.length - 1;
        int mid = 0;
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (nums[mid] == target) {
                /* find a index where item equals target,
                set to start, nums[mid] >= target,
                if nums[mid] > target, end = mid -1;
                find the last index where item == target
                */
                start = mid;
```

```
            } else if (nums[mid] < target) {
                start = mid;
            } else {
                end = mid;
            }
        }
        // end - start = 1

        if (nums[end] == target) {
            return end;
        }
        if (nums[start] == target) {
            return start;
        }
        return -1;
    }
}
```

## 28. Search a 2D Matrix ☆

Description | Notes | >_ Testcase | Judge

Write an efficient algorithm that searches for a value in an *m* x *n* matrix.

This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Have you met this question in a real interview? Yes

**Example**

Consider the following matrix:

```
[
    [1, 3, 5, 7],
    [10, 11, 16, 20],
    [23, 30, 34, 50]
]
```

Given `target = 3`, return `true`.

```
public class Solution {
    /*
     * @param matrix: matrix, a list of lists of integers
     * @param target: An integer
     * @return: a boolean, indicate whether matrix contains target
     */
    public boolean searchMatrix(int[][] matrix, int target) {
        // write your code here
```

```
if (matrix == null || matrix.length == 0) {
    return false;
}
if (matrix[0] == null || matrix[0].length == 0) {
    return false;
}
int row = matrix.length;
int column = matrix[0].length;
int start = 0, end = row - 1;
int mid;
// find the last index where matrix[index][0] <= target;
while (start + 1 < end) {
    mid = start + (end - start) / 2;
    if (matrix[mid][0] == target) {
        return true;
    } else if (matrix[mid][0] < target) {
        start = mid;
    } else {
        end = mid;
    }

}
if (matrix[end][0] <= target) {
    row = end;
} else if (matrix[start][0] <= target) {
    row = start;
} else {
    return false; // all integers are larger than target
}

// find the integer matrix[row][index] = target
start = 0;
end = matrix[0].length - 1;
mid = 0;
while (start + 1 < end) {
    mid = start + (end - start) / 2;
    if (matrix[row][mid] == target) {
        return true;
    } else if (matrix[row][mid] < target) {
        start = mid + 1;
    } else {
        end = mid - 1;
    }
}
if (matrix[row][end] == target) {
    return true;
} else if (matrix[row][start] == target) {
    return true;
} else {
```

```
            return false;
        }

    }
    /*



        // find the column index, the number equal to target
        start = 0;
        end = column - 1;
        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            if (matrix[row][mid] == target) {
                return true;
            } else if (matrix[row][mid] < target) {
                start = mid;
            } else {
                end = mid;
            }
        }
        if (matrix[row][start] == target) {
            return true;
        } else if (matrix[row][end] == target) {
            return true;
        }
        return false;
    */
}
```

# 585. Maximum Number in Mountain Sequence ☆

Description | Notes | >_ Testcase | Judge

Given a mountain sequence of `n` integers which increase firstly and then decrease, find the mountain top.

Have you met this question in a real interview? Yes

**Example**

Given `nums` = `[1, 2, 4, 8, 6, 3]` return `8`
Given `nums` = `[10, 9, 8, 7]`, return `10`

**Tags**

**Related Problems**

```
public class Solution {
    /*
     * @param nums: a mountain sequence which increase firstly and then decrease
     * @return: then mountain top
     */
    public int mountainSequence(int[] nums) {
        // write your code here
        if (nums == null || nums.length == 0) {
            return 0;
        }
        int start = 0;
        int end = nums.length - 1;
        int mid = 0;
        while (start + 1 < end) {
```

```
            mid = start + (end - start) / 2;
            if (nums[mid] > nums[mid + 1]) {
                end = mid;
            } else {
                start = mid;
            }
        }
        return Math.max(nums[start], nums[end]);
    }
}
```

# 447. Search in a Big Sorted Array ☆

Description | Notes | Testcase | Judge

Given a big sorted array with positive integers sorted by ascending order. The array is so big so that you can not get the length of the whole array directly, and you can only access the kth number by `ArrayReader.get(k)` (or ArrayReader->get(k) for C++). Find the first index of a target number. Your algorithm should be in O(log k), where k is the first index of the target number.

Return -1, if the number doesn't exist in the array.

**Notice**

If you accessed an inaccessible index (outside of the array), ArrayReader.get will return `2,147,483,647`.

Have you met this question in a real interview? Yes

**Example**

Given `[1, 3, 6, 9, 21, ...]`, and target = `3`, return `1`.

Given `[1, 3, 6, 9, 21, ...]`, and target = `4`, return `-1`.

```
public class Solution {
    /*
     * @param reader: An instance of ArrayReader.
     * @param target: An integer
     * @return: An integer which is the first index of target.
     */
    public int searchBigSortedArray(ArrayReader reader, int target) {
        // write your code here
        //get index that reader.get(i) >= target
        int index = 1;
        // exponential backoff
        while(reader.get(index - 1) < target) {
            index = index * 2;
        }
        // binary search the target between 0 and index
```

```
        int start = 0, end = index - 1;
        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            if (reader.get(mid) < target) {
                start = mid + 1;
            } else { // reader.get(mid) >= target
            // we need to find the first index
                end = mid;
            }
        }
        if (reader.get(start) == target) {
            return start;
        }
        if (reader.get(end) == target) {
            return end;
        }

        return -1;

    }
}
```

## 159. Find Minimum in Rotated Sorted Array ☆

Description | Notes | >_ Testcase | Judge

Suppose a sorted array is rotated at some pivot unknown to you beforehand.

(i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).

Find the minimum element.

**Notice**

You may assume no duplicate exists in the array.

Have you met this question in a real interview? Yes

**Example**

Given `[4, 5, 6, 7, 0, 1, 2]` return `0`

**Tags**

```
public class Solution {
    /*
     * @param nums: a rotated sorted array
     * @return: the minimum number in the array
     */
    public int findMin(int[] nums) {
        // write your code here
        if (nums == null || nums.length == 0) {
            return -1;
        }
        int start = 0, end = nums.length - 1;
        int target = nums[nums.length - 1];
        int mid = 0;
        // min is the first element <= target
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (nums[mid] <= target) { // 4567012 select the front part
                end = mid;
```

```
            } else { // nums[mid] > target back part
                start = mid;
            }
        }
        if (nums[start] <= target) {
            return nums[start];
        } else {
            return nums[end];
        }
    }
}
/*
4567012
start value end value mid value target
0     4    6    2  0     4   2
loop 1
0     4    6    2  3     7   2
3     7    6    2  3     7   2
loop 2
3     7    6    2  4     0   2
3     7    4    0  4     0   2

jump out of loop
nums[start]=7
nums[end] = 0

*/
```

## 75. Find Peak Element ☆

Description | Notes | Testcase | Judge

There is an integer array which has the following features:

- The numbers in adjacent positions are different.
- A[0] < A[1] && A[A.length - 2] > A[A.length - 1].

We define a position P is a peak if:

```
A[P] > A[P-1] && A[P] > A[P+1]
```

Find a peak element in this array. Return the index of the peak.

**Notice**

- It's guaranteed the array has at least one peak.
- The array may contain multiple peeks, find any of them.
- The array has at least 3 numbers in it.

Have you met this question in a real interview? Yes

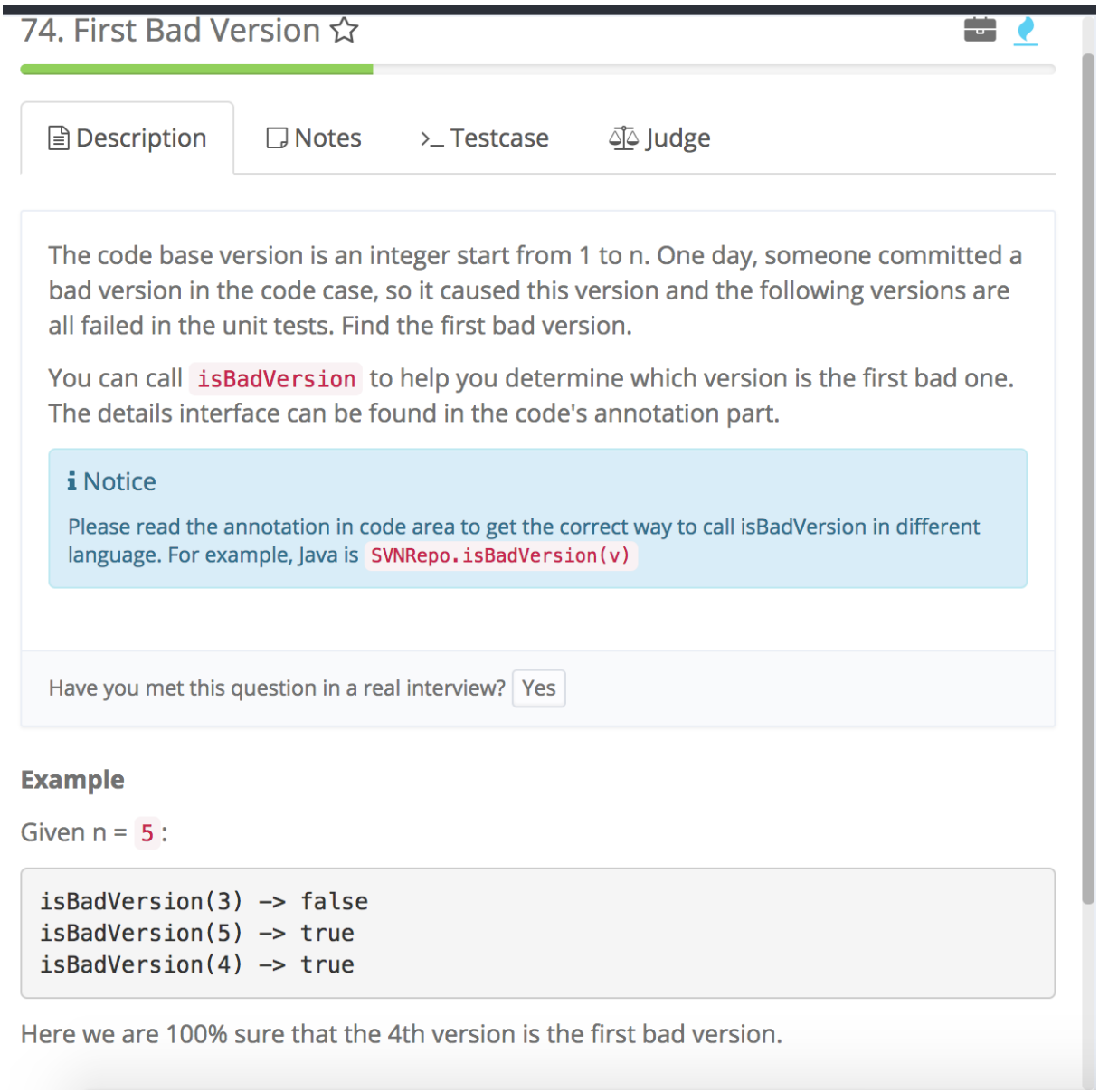
**Example**

Given `[1, 2, 1, 3, 4, 5, 7, 6]`

Return index `1` (which is number 2) or `6` (which is number 7)

```
public class Solution {
    /*
     * @param A: An integers array.
     * @return: return any of peek positions.
     */
    public int findPeak(int[] A) {
        // write your code here
        // 找到保证有答案的范围
        int start = 1;
        int end = A.length - 2;
```

```
        int mid = 1;
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (A[mid] < A[mid + 1]) { //rise
                start = mid;
            } else if (A[mid] > A[mid + 1]) { // go down
                end = mid;
            } else {
                end = mid;
            }
        }
        if (A[start] < A[end]) {
            return end;
        } else {
            return start;
        }
    }
}
```



# 74. First Bad Version ☆

Description | Notes | >_ Testcase | Judge

The code base version is an integer start from 1 to n. One day, someone committed a bad version in the code case, so it caused this version and the following versions are all failed in the unit tests. Find the first bad version.

You can call `isBadVersion` to help you determine which version is the first bad one. The details interface can be found in the code's annotation part.

**Notice**

Please read the annotation in code area to get the correct way to call isBadVersion in different language. For example, Java is `SVNRepo.isBadVersion(v)`

Have you met this question in a real interview? Yes

**Example**

Given n = `5`:

```
isBadVersion(3) -> false
isBadVersion(5) -> true
isBadVersion(4) -> true
```

Here we are 100% sure that the 4th version is the first bad version.

```
/**
 * public class SVNRepo {
 *     public static boolean isBadVersion(int k);
 * }
 * you can use SVNRepo.isBadVersion(k) to judge whether
 * the kth code version is bad or not.
*/
```

```
public class Solution {
    /*
     * @param n: An integer
     * @return: An integer which is the first bad version.
     */
    public int findFirstBadVersion(int n) {
        // write your code here
        // first posion of target
        int start = 1, end = n;
        int mid = 1;
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (SVNRepo.isBadVersion(mid)) {
                end = mid;
            } else {
                start = mid;
            }
        }
        if (SVNRepo.isBadVersion(start)) {
            return start;
        }
        return end;
    }
}
```

## 62. Search in Rotated Sorted Array ☆

Description | Notes | Testcase | Judge

Suppose a sorted array is rotated at some pivot unknown to you beforehand.

(i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).

You are given a target value to search. If found in the array return its index, otherwise return -1.

You may assume no duplicate exists in the array.

Have you met this question in a real interview? Yes

**Example**

For `[4, 5, 1, 2, 3]` and `target=1`, return `2`.

For `[4, 5, 1, 2, 3]` and `target=0`, return `-1`.

```
public class Solution {
    /*
     * @param A: an integer rotated sorted array
     * @param target: an integer to be searched
     * @return: an integer
     */
    public int search(int[] A, int target) {
        // write your code here
        if (A == null || A.length == 0) {
            return -1;
        }

        int start = 0;
        int end = A.length - 1;
        int mid = 0;

        while(start + 1 < end) {
            mid = start + (end - start) / 2;
            if (A[start] < A[mid]) { // start and mid on the same line
                if (A[start] <= target && target <= A[mid]) {
                    end = mid; // target in the front part
```

```
            } else {
                start = mid;
            }
        } else { // start and mid on the different line
            if (A[mid] <=target && target <= A[end]) {
                start = mid;
            } else {
                end = mid;
            }

        }
    }
    if (A[start] == target) {
        return start;
    }
    if (A[end] == target) {
        return end;
    }
    return -1;
  }
}
/*
  public int search(int[] A, int target) {
    if (A == null || A.length == 0) {
        return -1;
    }

    int start = 0;
    int end = A.length - 1;
    int mid;

    while (start + 1 < end) {
        mid = start + (end - start) / 2;
        if (A[mid] == target) {
            return mid;
        }
        if (A[start] < A[mid]) {
            // situation 1, red line
            if (A[start] <= target && target <= A[mid]) {
                end = mid;
            } else {
                start = mid;
            }
        } else {
            // situation 2, green line
            if (A[mid] <= target && target <= A[end]) {
                start = mid;
            } else {
                end = mid;
```

## 597. Subtree with Maximum Average ☆

Description | Notes | Testcase | Judge

Given a binary tree, find the subtree with maximum average. Return the root of the subtree.

**Notice**

LintCode will print the subtree which root is your return node.
It's guaranteed that there is only one subtree with maximum average.

Have you met this question in a real interview? Yes

**Example**

Given a binary tree:

```
     1
   /   \
 -5     11
 / \   /  \
1   2 4    -2
```

return the node `11`.

```
            }
        }
    } // while

    if (A[start] == target) {
        return start;
    }
    if (A[end] == target) {
        return end;
    }
    return -1;
  }
*/
```