

LintCode领扣题解 (/problem) / 目标最后位置 · Last Position of Target

## 目标最后位置 · Last Position of Target

中文

LintCode 版权所有 (/problem/?tags=lintcode-copyright)

二分法 (/problem/?tags=binary-search)

### 描述

给一个升序数组, 找到 target 最后一次出现的位置, 如果没出现过返回 -1

### 样例

样例 1:

输入: nums = [1,2,2,4,5,5], target = 2  
输出: 2

样例 2:

输入: nums = [1,2,2,4,5,5], target = 6  
输出: -1

在线评测地址: <https://www.lintcode.com/problem/last-position-of-target/> (<https://www.lintcode.com/problem/last-position-of-target/>)

收起题目描述 ^

语言类型

ALL (19)

java (9)

cpp (7)

python (3)

上传题解



**serenity**

更新于 10/13/2020, 8:59:02 PM

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
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;
        int end = nums.length - 1;
        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            if (nums[mid] == target) {
                start = mid;
            } else if (nums[mid] < target) {
                start = mid;
            } else {
                end = mid;
            }
        }
        if (nums[end] == target) {
            return end;
        } else if (nums[start] == target) {
            return start;
        } else {
            return -1;
        }
    }
}
```

👍 获赞 1

💬 添加评论

# 你的口袋题库

## 2000+ 算法真题、国内外名企题库免费开放



九章算法APP



DDBear

更新于 8/2/2020, 5:12:10 AM

## 算法：二分

### 算法思路

- 题目要求我们找到 target 最后一次出现的位置, 由于数组是有序数组, 我们可以考虑使用二分法来查找

### 代码思路

1. 设置左边界等于0, 右边界等于 numsLen - 1

2. 对于 `mid` 所指向的数，当 `target < nums[mid]` 时，说明 `target` 在 `mid` 左侧，那么 `right = mid`；否则说明 `target` 在 `mid` 右侧，或者如果 `target == nums[mid]` 的话说明 `mid` 还可能存在 `target` 那么 `left = mid`
3. 不断重复 2 直到 `left + 1 == right` 退出
4. 判断 `nums[right]` 是否等于 `target`，若等于返回 `right`，否则返回 `left`，注意一定要先判 `nums[right]`，因为 `nums[left]` 可能也等于 `target`，但不是最后的位置

## 复杂度分析

$N$  表示 `nums` 数组长度

- 空间复杂度： $O(1)$
- 时间复杂度： $O(\log N)$

java   c++   python

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */

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) {

        if(nums == null || nums.length == 0)

            return -1;

        if(target < nums[0] || target > nums[nums.length-1])

            return -1;

        int left = 0, right = nums.length-1;

        while(left+1 < right){

            int mid = left + (right - left) / 2;

            if(nums[mid] > target)

                right = mid;
```

```
        else

            left = mid;

    }

    if(nums[right] == target)

        return right;

    if(nums[left] == target)

        return left;

    return -1;

}
```

👍 获赞 1      💬 添加评论



令狐冲

更新于 6/9/2020, 7:04:31 AM

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param A an integer array sorted in ascending order
     * @param target an integer
     * @return an integer
     */
    int lastPosition(vector<int>& A, int target) {
        // Write your code here
        int n = A.size();
        if (n == 0)
            return -1;
        if (A[n-1] < target || A[0] > target)
            return -1;

        int l = 0, r = n - 1;
        int end = -1;
        while (l <= r) {
            int mid = (l + r) >> 1;
            if (A[mid] == target)
                end = mid;
            if (A[mid] <= target) {
                l = mid + 1;
            } else
                r = mid - 1;
        }
        return end;
    }
};
```

令狐冲  
更新于 6/9/2020, 7:04:31 AM

👍 获赞 0      💬 10 条评论

**Riley**  
更新于 6/9/2020, 7:03:48 AM

Page 5 of 21

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
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;
        int end = nums.length - 1;
        while (start + 1 < end) {
            if (nums[start] == target && nums[end] == target) {
                break;
            }
            int mid = start + (end - start) / 2;
            if (nums[mid] == target) {
                start = mid;
            } else if (nums[mid] < target) {
                start = mid;
            } else {
                end = mid;
            }
        }
        if (nums[end] == target) {
            return end;
        } else if (nums[start] == target) {
            return start;
        } else {
            return -1;
        }
    }
}
```

👍 获赞 3

💬 添加评论

**D**

更新于 6/9/2020, 7:03:49 AM

为啥大家都用模板写这么多行呀.....

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class Solution {
    public int lastPosition(int[] nums, int target) {
        if( nums == null || nums.length ==0 ) return -1;
        int n = nums.length;
        int l = 0, r = n;
        while(l+1<r){
            int mid = l+(r-l)/2;
            if(nums[mid]<=target)
                l = mid;
            else
                r = mid;
        }
        return nums[l] == target ? l : -1;
    }
}
```

👍 获赞 2

💬 5 条评论

**九章用户GO5M2H**

更新于 6/9/2020, 7:03:55 AM

原来的Version 1 有个小BUG

修正了下

numsmid ()==target的时候, 老师的答案直接返回了mid的值, 没有考虑target在字符串里重复的情况

我改成了 start = mid, 这样哪怕遇到1,1,1,1,1,1,1 (), 因为我们没有改变end,只改变了start, 也不会出问题

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public int lastPosition(int[] nums, int target) {
    int start = 0, end = nums.length-1;

    while (start + 1 < end ){
        int mid = start + ( end - start)/2;
        if(nums[mid] == target){
            start = mid;
        }else if (nums[mid] < target){
            start = mid;
        }else if (nums[mid] > target){
            end = mid;
        }
    }
    if (nums[end]==target){
        return end;
    }
    if (nums[start] == target){
        return start;
    }

    return -1;
}

```

👍 获赞 1

💬 添加评论

**Yummy Corgi**

更新于 6/9/2020, 7:03:52 AM

binary search (first/last)四种解法

1, 九章标准模板,  $start + 1 < end$  2, 使用index记录搜索过程中的符合条件的解 3,  $mid = start + (end - start + 1) / 2$ , 使得mid偏向右边 4, 使用firstIndex这个基本函数 (C++ lower\_bound), 找到第一个index使得  $A[index] \geq target$ 。利用这个基本函数, 可以实现找任意位置, 最左边, 最右边三种功能

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &A, int target) {
        // write your code here
        int n = A.size();
        if (n == 0) {

```



```

        return -1;
    }

    int start = 0, end = n - 1;
    int mid;

    while (start + 1 < end) {
        mid = start + (end - start) / 2;
        if (A[mid] <= target) {
            start = mid;
        } else {
            end = mid;
        }
    }

    // final check
    if (A[end] == target) {
        return end;
    }

    if (A[start] == target) {
        return start;
    }

    return -1;
}
};

```

```

class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &A, int target) {
        // write your code here
        int n = A.size();
        if (n == 0) {
            return -1;
        }

        int start = 0, end = n - 1;
        int mid;
        int index = -1;

        while (start <= end) {
            mid = start + (end - start) / 2;
            if (A[mid] < target) {
                start = mid + 1;
            } else if (A[mid] == target) {
                index = mid;
                start = mid + 1;
            } else {
                end = mid - 1;
            }
        }

        return index;
    }
};

```

```

class Solution {
public:

```

```
/**
 * @param nums: An integer array sorted in ascending order
 * @param target: An integer
 * @return: An integer
 */
int lastPosition(vector<int> &A, int target) {
    // write your code here
    int n = A.size();
    if (n == 0) {
        return -1;
    }

    int start = 0, end = n - 1;
    int mid;

    while (start < end) {
        mid = start + (end - start + 1) / 2;
        if (A[mid] < target) {
            start = mid + 1;
        } else if (A[mid] == target) {
            start = mid;
        } else {
            end = mid - 1;
        }
    }

    return A[start] == target ? start : -1;
}



};

class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &A, int target) {
        // write your code here
        int n = A.size();
        if (n == 0) {
            return -1;
        }

        int index = firstIndex(A, target + 1) - 1;

        return A[index] == target ? index : -1;
    }
private:
    int firstIndex(vector<int> &A, int target) {
        // first index such that A[index] >= target
        // C++, lower_bound
        int n = A.size();
        if (n == 0) {
            return n;
        }
        int start = 0, end = n - 1;
        int mid;
        while (start < end) {
            mid = start + (end - start) / 2;
            if (A[mid] < target) {
                start = mid + 1;
            } else {
                end = mid;
            }
        }
    }
}
```

```
        return A[start] >= target ? start : n;  
    }  
};
```

 获赞 1 1 条评论**Yummy Corgi**

更新于 10/5/2020, 12:18:24 PM

全网最简单的二分法模板，不接受反驳。一个firstGreaterEqual函数（等价于C++ lower\_bound）解决所有问题：第一个，最后一个，任意一个target位置。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &nums, int target) {
        // write your code here
        int n = nums.size();
        if (n == 0) {
            return -1;
        }

        // first index
        int firstIndex = firstGreaterEqual(nums, target);
        if (firstIndex == n || nums[firstIndex] != target) {
            firstIndex = -1;
        }

        // last index
        // This trick is very useful, makes one BS works for all cases
        int lastIndex = firstGreaterEqual(nums, target + 1) - 1;
        if (nums[lastIndex] != target) {
            lastIndex = -1;
        }

        return lastIndex;
    }
private:
    // This helper function can be used to find first/last index of a target
    // C++ lower_bound, find first index i st A[i] >= target
    // think of A[n] = INF
    int firstGreaterEqual(vector<int> &nums, int target) {
        // return n if not find
        int n = nums.size();
        int start = 0, end = n - 1;
        int mid;
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (nums[mid] < target) {
                start = mid;
            } else {
                end = mid;
            }
        }

        // final check, check start frist
        if (nums[start] >= target) {
            return start;
        }
        if (nums[end] >= target) {
            return end;
        }
        return n;
    }
};

```

 获赞 0     添加评论**九章用户16GH7O**

更新于 6/9/2020, 7:04:24 AM

使用 recursion 解决此问题。recursion 终结情况1: start > end, 此时没有找到匹配数, return -1 recursion 终结情况2: start == end 且 nums[start] == target, 此条件必要, 不然会跳不出 num[mid] == target 的循环。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
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;
        }

        return positionHelper(nums, target, 0, nums.length - 1);
    }

    private int positionHelper(int[] nums, int target, int start, int end) {
        // ending situation 1
        if (start > end) {
            return -1;
        }
        // ending situation 2
        if (start == end && nums[start] == target) {
            return start;
        }
        // must use "+1" here, to make the mid closer to the right side, or it won't reach last position
        // e.g. [5,5]
        int mid = start + (end - start + 1) / 2;

        if (nums[mid] == target) {
            // continue recursion to find last one
            return positionHelper(nums, target, mid, end);
        }
        if (nums[mid] < target) {
            return positionHelper(nums, target, mid + 1, end);
        }
        return positionHelper(nums, target, start, mid - 1);
    }
}
```

 获赞 0     添加评论**九章用户16GH7O**

更新于 6/9/2020, 7:04:24 AM

使用 recursion 解决此问题。recursion 终结情况1: start > end, 此时没有找到匹配数, return -1 recursion 终结情况2: start == end 且 nums[start] == target, 此条件必要, 不然会跳不出 num[mid] == target 的循环。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
```

```
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;
        }

        return positionHelper(nums, target, 0, nums.length - 1);
    }

    private int positionHelper(int[] nums, int target, int start, int end) {
        // ending situation 1
        if (start > end) {
            return -1;
        }
        // ending situation 2
        if (start == end && nums[start] == target) {
            return start;
        }
        // must use "+1" here, to make the mid closer to the right side, or it won't reach last position
        // e.g. [5,5]
        int mid = start + (end - start + 1) / 2;

        if (nums[mid] == target) {
            // continue recursion to find last one
            return positionHelper(nums, target, mid, end);
        }
        if (nums[mid] < target) {
            return positionHelper(nums, target, mid + 1, end);
        }
        return positionHelper(nums, target, start, mid - 1);
    }
}
```

👍 获赞 0

💬 添加评论



九章用户2WYHZK

更新于 6/9/2020, 7:04:23 AM

分享我的二分法模版。while循环结束后, 两个指针相邻, 但是right指针在左, left指针在右 (left = right + 1)。也可以理解为, right指针在OXX分界线左边一个位置, 而left指针在分界线右边的第一个位置。



invitation/sh



/\*\*  
\* 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。  
\* - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。  
\* - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版  
\* - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD  
\* - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课  
\* - 更多详情请见官方网站: [http://www.jiuzhang.com/?utm\\_source=code](http://www.jiuzhang.com/?utm_source=code)  
\*/  
**public class** Solution {  
  
    **public** int lastPosition(int[] nums, int target) {  
        **if** (nums == null || nums.length == 0)  
            **return** -1;  
  
        int left = 0, right = nums.length - 1;  
        **while** (left <= right)  
        {  
            int mid = left + (right - left) / 2;  
            **if** (target == nums[mid])  
                left = mid + 1;  
            **else if** (target < nums[mid])  
                right = mid - 1;  
            **else**  
                left = mid + 1;  
        }  
  
        **if** (right >= 0 && nums[right] == target) // right = last position  
            **return** right;  
        **return** -1;  
    }  
}

👍 获赞 0

💬 添加评论



小李子

更新于 6/9/2020, 7:04:20 AM

因为是排序数组, 所以可以在找到target之后用while循环找最后位置

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
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.length == 0) {
            return -1;
        }

        int start = 0;
        int end = nums.length - 1;
        int result = 0;
        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            result = mid;

            if (target > nums[mid]) {
                start = mid;
            } else if (target < nums[mid]) {
                end = mid;
            } else {
                while (mid < nums.length && nums[mid] == target) {
                    mid++;
                }

                result = mid - 1;
                return result;
            }
        }
        return -1;
    }
}

```

👍 获赞 0

💬 添加评论



**JianxiaGao**

更新于 6/9/2020, 7:04:18 AM

This solution is based on optimization in Programming pearl, column 9, where the loop invariant is set as `numsstart () <= target` and `numsend () > target && start < end`. Compared with given solution, this only have two compare in the loop.



```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &nums, int target){
        int n = nums.size();
        if (n == 0) return -1;
        if (nums[n-1] < target || nums[0] > target) return -1;
        int start = -1;
        int end = nums.size();
        while(start+1 != end){
            auto mid = (start+end)/2;
            if(nums[mid] > target)
                end = mid;
            else
                start = mid;
        }
        if(start < 0 || nums[start] != target) return -1;
        return start;
    }
};
```

👍 获赞 0

💬 添加评论

**JianxiaGao**

更新于 6/9/2020, 7:04:18 AM

I used similar algorithm in programming pearl, column 9.3. Assume numssize () > target, always move end if numsmid () > target

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &nums, int target){
        if(nums.size() == 0) return -1;
        int start = 0;
        int end = nums.size();
        while(start+1 != end){
            auto mid = (start+end)/2;
            if(nums[mid] > target) end = mid;
            else start = mid;
        }
        if(start >= nums.size() || nums[start] != target) return -1;
        return start;
    }
};
```

👍 获赞 0

💬 添加评论



社会我喵哥

更新于 6/9/2020, 7:04:10 AM

最最最典型的二分练习题, 和 14. First Position of Target 的区别在于 1. 一个前者先测试right, 后者先测试left 2. 中间判断的条件一个是>, 一个是>=

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param nums: An integer array sorted in ascending order
     * @param target: An integer
     * @return: An integer
     */
    int lastPosition(vector<int> &nums, int target) {
        // write your code here
        if (nums.size() == 0) return -1;
        int left = 0;
        int right = nums.size() - 1;
        while (left + 1 < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] > target) {
                right = mid;
            } else {
                left = mid;
            }
        }
        if (nums[right] == target) return right;
        if (nums[left] == target) return left;
        return -1;
    }
};
```

👍 获赞 0

💬 添加评论

**DavidT**

更新于 6/9/2020, 7:04:06 AM

二分搜索, 唯一-特殊的地方是要尽量找大的。O (logN) 思路是逐渐缩小范围到两个 (之内)

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param nums: An integer array sorted in ascending order
    @param target: An integer
    @return: An integer
    """
    def lastPosition(self, nums, target):
        if not nums:
            return -1

        start = 0
        end = len(nums) - 1

        while start + 1 < end:
            mid = start + (end - start) // 2
            if target < nums[mid]: #接下来在左边找
                end = mid
            else: #在右边找
                start = mid

        if target == nums[end]:
            return end
        elif target == nums[start]:
            return start
        else:
            return -1

```

👍 获赞 0

💬 添加评论

## 进阶课程

视频+互动

直播+互动

直播+互动

互动课

### 九章算法班 2021 版

8周时间精通 57 个核心高频考点, 9 招击破 FLAG、BATJ 算法面试。22...

### 系统架构设计 System Design 2021 版

成为百万架构师必上。30 课时带你快速掌握18大系统架构设计知识点与面...

### 九章算法面试高频题冲刺班

每期更新 15% 题目, 考前押题, 一举拿下FLAG & BATJ Offer

### 面向对象设计 OOD

应届生及亚马逊面试必考, IT求职必备基础

首页 (/?skip\_redirect=true)

联系我们 (mailto:info@jiuzhang.com)

加入我们 (/joinus)

Copyright © 2013-2020 九章算法 浙ICP备19045946号-1 (http://www.miibeian.gov.cn/)

商务合作: fukesu@jiuzhang.com (mailto:fukesu@jiuzhang.com)

(http://weibo.com/ninechapter)

知 (https://www.zhihu.com/people/crackinterview/)

(/)

https://www.jiuzhang.com/problem/last-position-of-target/

Page 21 of 21