# 题目

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。

你的算法时间复杂度必须是 O(log n)级别。

如果数组中不存在目标值，返回 [-1, -1]。

**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]

# 分析

## 方法一：二分查找

**思路：**

**代码：**

#include <iostream>

#include <vector>

using namespace std;

class Solution {

private:

int findFitstPosition(vector<int> &nums, int target) {

int size = nums.size();

int left = 0;

int right = size - 1;

while (left < right) {

int mid = (left + right) >> 1;

if (nums[mid] < target) {

left = mid + 1;

} else {

right = mid;

}

}

if (nums[left] != target) {

return -1;

}

return left;

}

int findLastPosition(vector<int> &nums, int target) {

int size = nums.size();

int left = 0;

int right = size - 1;

while (left < right) {

int mid = (left + right + 1) >> 1;

if (nums[mid] > target) {

right = mid - 1;

} else {

left = mid;

}

}

if (nums[left] != target) {

return -1;

}

return left;

}

public:

vector<int> searchRange(vector<int> &nums, int target) {

int size = nums.size();

if (size == 0) {

return {-1, -1};

}

int fitstPosition = findFitstPosition(nums, target);

if (fitstPosition == -1) {

return {-1, -1};

}

int lastPosition = findLastPosition(nums, target);

return {fitstPosition, lastPosition};

}

};

**复杂度：**

时间复杂度：O(logN)，这里N是数组的长度，两个子问题都是二分查找，因此时间复杂度为对数级别。

空间复杂度：O(1)，只使用了常数个数的辅助变量、指针。

## 方法二：二分查找

**思路：**

**代码：**

class Solution {

public:

vector<int> searchRange(vector<int>& nums, int target) {

int left=0;

int right=nums.size()-1;

int mid;

while(left<=right){

mid=(left+right)/2;

if(nums[mid]>target){

right=mid-1;

}

else if(nums[mid]<target){

left=mid+1;

}

else if(nums[mid]==target){

if(nums[right]>target)

right--;

else if(nums[left]<target)

left++;

if(nums[right]==target&&nums[left]==target)

return vector<int>({left,right});

}

}

return vector<int>({-1,-1});

}

};