# 题目

给你一个下标从 0 开始的整数数组nums 以及一个目标元素 target 。

目标下标是一个满足nums[i] == target的下标 i 。

将 nums 按 非递减 顺序排序后，返回由 nums 中目标下标组成的列表。如果不存在目标下标，返回一个 空 列表。返回的列表必须按 递增 顺序排列。

示例 1：

输入：nums = [1,2,5,2,3], target = 2

输出：[1,2]

解释：排序后，nums 变为 [1,2,2,3,5] 。

满足 nums[i] == 2 的下标是 1 和 2 。

示例 2：

输入：nums = [1,2,5,2,3], target = 3

输出：[3]

解释：排序后，nums 变为 [1,2,2,3,5] 。

满足 nums[i] == 3 的下标是 3 。

示例 3：

输入：nums = [1,2,5,2,3], target = 5

输出：[4]

解释：排序后，nums 变为 [1,2,2,3,5] 。

满足 nums[i] == 5 的下标是 4 。

示例 4：

输入：nums = [1,2,5,2,3], target = 4

输出：[]

解释：nums 中不含值为 4 的元素。

提示：

1 <= nums.length <= 100

1 <= nums[i], target <= 100

# 分析

## 方法一：排序后遍历

我们首先按要求对 nums数组升序排序，随后从左到右遍历数组中的所有元素，并按顺序记录所有数值等于 target的元素的下标。这样我们可以保证记录的下标数组中的下标（如果存在）必定按照递增顺序排列。

最后，如果符合要求的下标存在，我们返回记录的下标数组作为答案；如果不存在，我们返回空数组即可。

代码：

class Solution {

public:

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

int n = nums.size();

sort(nums.begin(), nums.end());

vector<int> res;

for (int i = 0; i < n; ++i){

if (nums[i] == target){

res.push\_back(i);

}

}

return res;

}

};

复杂度分析

时间复杂度：O(nlogn)，其中n为 nums的长度。排序的时间复杂度为O(nlogn)，遍历记录目标下标数组的时间复杂度为O(n)$。

空间复杂度：O(logn)，即为排序的栈空间开销。