# 题目

给你一个下标从 0 开始、长度为 n 的整数数组 nums ，以及整数 indexDifference 和整数 valueDifference 。

你的任务是从范围 [0, n - 1] 内找出 2 个满足下述所有条件的下标 i 和 j ：

abs(i - j) >= indexDifference 且

abs(nums[i] - nums[j]) >= valueDifference

返回整数数组 answer。如果存在满足题目要求的两个下标，则 answer = [i, j] ；否则，answer = [-1, -1] 。如果存在多组可供选择的下标对，只需要返回其中任意一组即可。

注意：i 和 j 可能 相等 。

示例 1：

输入：nums = [5,1,4,1], indexDifference = 2, valueDifference = 4

输出：[0,3]

解释：在示例中，可以选择 i = 0 和 j = 3 。

abs(0 - 3) >= 2 且 abs(nums[0] - nums[3]) >= 4 。

因此，[0,3] 是一个符合题目要求的答案。

[3,0] 也是符合题目要求的答案。

示例 2：

输入：nums = [2,1], indexDifference = 0, valueDifference = 0

输出：[0,0]

解释：

在示例中，可以选择 i = 0 和 j = 0 。

abs(0 - 0) >= 0 且 abs(nums[0] - nums[0]) >= 0 。

因此，[0,0] 是一个符合题目要求的答案。

[0,1]、[1,0] 和 [1,1] 也是符合题目要求的答案。

示例 3：

输入：nums = [1,2,3], indexDifference = 2, valueDifference = 4

输出：[-1,-1]

解释：在示例中，可以证明无法找出 2 个满足所有条件的下标。

因此，返回 [-1,-1] 。

提示：

1 <= n == nums.length <= 100

0 <= nums[i] <= 50

0 <= indexDifference <= 100

0 <= valueDifference <= 50

# 分析

## 方法一：暴力法

思路：

代码：

class Solution {

public:

vector<int> findIndices(vector<int>& nums, int indexDifference, int valueDifference)

{

int i=0;

int j=0;

int len=nums.size();

for(i=0;i<len;i++)

{

for(j=0;j<len;j++)

{

if(abs(i-j)>=indexDifference && abs(nums[i]-nums[j])>=valueDifference)

{

vector <int>a;

a.push\_back(i);

a.push\_back(j);

return a;

}

}

}

return {-1,-1};

}

};

## 方法二：双指针

思路：

不妨设i≤j−indexDifference。

类似 121. 买卖股票的最佳时机，我们可以在枚举j的同时，维护nums[i]的最大值mx和最小值mn。

那么只要满足下面两个条件中的一个，就可以返回答案了。

mx−nums[j]≥valueDifference

nums[j]−mn≥valueDifference

代码实现时，可以维护最大值的下标 maxIdx和最小值的下标 minIdx。

代码：

class Solution {

public:

vector<int> findIndices(vector<int> &nums, int indexDifference, int valueDifference) {

int max\_idx = 0, min\_idx = 0;

for (int j = indexDifference; j < nums.size(); j++) {

int i = j - indexDifference;

if (nums[i] > nums[max\_idx]) {

max\_idx = i;

} else if (nums[i] < nums[min\_idx]) {

min\_idx = i;

}

if (nums[max\_idx] - nums[j] >= valueDifference) {

return {max\_idx, j};

}

if (nums[j] - nums[min\_idx] >= valueDifference) {

return {min\_idx, j};

}

}

return {-1, -1};

}

};

复杂度分析

时间复杂度：O(n)，其中n为nums的长度。

空间复杂度：O(1)。