# 题目

给你一个整数数组nums和两个整数k和t。请你判断是否存在 两个不同下标i和j，使得abs(nums[i] - nums[j]) <= t，同时又满足abs(i - j) <= k 。

如果存在则返回true，不存在返回false。

示例 1：

输入：nums = [1,2,3,1], k = 3, t = 0

输出：true

示例 2：

输入：nums = [1,0,1,1], k = 1, t = 2

输出：true

示例 3：

输入：nums = [1,5,9,1,5,9], k = 2, t = 3

输出：false

提示：

0 <= nums.length <= 2 \* 104

-231 <= nums[i] <= 231 - 1

0 <= k <= 104

0 <= t <= 231 - 1

注意：本题与主站 220 题相同： https://leetcode-cn.com/problems/contains-duplicate-iii/

# 分析

## 方法一：滑动窗口+二分

class Solution {

public:

bool containsNearbyAlmostDuplicate(vector<int>& nums, int k, int t) {

set<long long> window; // 使用 long long 类型，避免溢出

for (int i = 0; i < nums.size(); ++i) {

// 在窗口中查找大于等于 nums[i] 的最小元素

auto it = window.lower\_bound(static\_cast<long long>(nums[i]) - t);

// 如果找到并且差值满足要求，则返回 true

if (it != window.end() && \*it - nums[i] <= t) {

return true;

}

// 将当前元素加入窗口

window.insert(nums[i]);

// 如果窗口大小超过 k，则移除窗口最左侧的元素

if (i >= k) {

window.erase(nums[i - k]);

}

}

// 遍历完成仍未找到满足条件的情况，返回 false

return false;

}

};

## 方法二：桶排序

思路：

我们也可以使用利用桶排序的思想解决本题。我们按照元素的大小进行分桶，维护一个滑动窗口内的元素对应的元素。

对于元素 x，其影响的区间为 [x−t,x+t]。于是我们可以设定桶的大小为t+1。如果两个元素同属一个桶，那么这两个元素必然符合条件。如果两个元素属于相邻桶，那么我们需要校验这两个元素是否差值不超过t。如果两个元素既不属于同一个桶，也不属于相邻桶，那么这两个元素必然不符合条件。

具体地，我们遍历该序列，假设当前遍历到元素x，那么我们首先检查 x所属于的桶是否已经存在元素，如果存在，那么我们就找到了一对符合条件的元素，否则我们继续检查两个相邻的桶内是否存在符合条件的元素。

实现方面，我们将 int范围内的每一个整数 x表示为 x=(t+1)×a+b(0≤b≤t)的形式，这样 x即归属于编号为 a的桶。因为一个桶内至多只会有一个元素，所以我们使用哈希表实现即可。

代码：

class Solution {

public:

int getID(int x, long w) {

return x < 0 ? (x + 1ll) / w - 1 : x / w;

}

bool containsNearbyAlmostDuplicate(vector<int>& nums, int k, int t) {

unordered\_map<int, int> mp;

int n = nums.size();

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

long x = nums[i];

int id = getID(x, t + 1ll);

if (mp.count(id)) {

return true;

}

if (mp.count(id - 1) && abs(x - mp[id - 1]) <= t) {

return true;

}

if (mp.count(id + 1) && abs(x - mp[id + 1]) <= t) {

return true;

}

mp[id] = x;

if (i >= k) {

mp.erase(getID(nums[i - k], t + 1ll));

}

}

return false;

}

};

**复杂度分析：**

时间复杂度：O(n)，其中n是给定数组的长度。每个元素至多被插入哈希表和从哈希表中删除一次，每次操作的时间复杂度均为O(1)。

空间复杂度：O(min(n,k))，其中n是给定数组的长度。哈希表中至多包含min(n,k+1)个元素。