# 题目

给你两个整数数组arr1，arr2和一个整数d，请你返回两个数组之间的距离值。

「距离值」定义为符合此距离要求的元素数目：对于元素arr1[i]，不存在任何元素arr2[j]满足|arr1[i]-arr2[j]| <= d。

示例 1：

输入：arr1 = [4,5,8], arr2 = [10,9,1,8], d = 2

输出：2

解释：

对于 arr1[0]=4 我们有：

|4-10|=6 > d=2

|4-9|=5 > d=2

|4-1|=3 > d=2

|4-8|=4 > d=2

所以 arr1[0]=4 符合距离要求

对于 arr1[1]=5 我们有：

|5-10|=5 > d=2

|5-9|=4 > d=2

|5-1|=4 > d=2

|5-8|=3 > d=2

所以 arr1[1]=5 也符合距离要求

对于 arr1[2]=8 我们有：

|8-10|=2 <= d=2

|8-9|=1 <= d=2

|8-1|=7 > d=2

|8-8|=0 <= d=2

存在距离小于等于 2 的情况，不符合距离要求

故而只有 arr1[0]=4 和 arr1[1]=5 两个符合距离要求，距离值为 2

示例 2：

输入：arr1 = [1,4,2,3], arr2 = [-4,-3,6,10,20,30], d = 3

输出：2

示例 3：

输入：arr1 = [2,1,100,3], arr2 = [-5,-2,10,-3,7], d = 6

输出：1

提示：

1 <= arr1.length, arr2.length <= 500

-10^3 <= arr1[i], arr2[j] <= 10^3

0 <= d <= 100

# 分析

## 方法一：模拟

思路：

按照题意模拟：对于arr1数组中的每一个元素xi，枚举数组arr2中的每一个元素yj，检查是否对于每一个yj都有∣xi−yj∣>d，如果是，就将答案增加1。

代码：

class Solution {

public:

int findTheDistanceValue(vector<int>& arr1, vector<int>& arr2, int d) {

int cnt = 0;

for (auto &num1 : arr1) {

bool match = true; // 初始假设存在符合条件的arr2[j]

for (auto &num2 : arr2) {

if (abs(num1 - num2) <= d) {

match = false; // 找到了符合条件的arr2[j]，所以match设为false

break; // 不需要继续检查arr2的其他元素

}

}

if (match) cnt++; // 如果match仍为true，说明没有找到符合条件的arr2[j]，cnt加1

}

return cnt;

}

};

复杂度分析

假设arr1中元素个数为n，arr2中元素个数为m。

时间复杂度：从代码可以看出这里的渐进时间复杂度是O(n×m)。

空间复杂度：这里没有使用任何的辅助空间，故渐进空间复杂度为O(1)。

## 方法二：二分查找

思路：

代码：

class Solution {

public:

int findTheDistanceValue(vector<int>& arr1, vector<int>& arr2, int d) {

// 先对arr2进行排序，以便可以使用二分查找

sort(arr2.begin(), arr2.end());

int cnt = 0;

for (int num1 : arr1) {

// 使用二分查找在arr2中查找满足条件的元素

int left = 0, right = arr2.size() - 1;

bool found = false;

while (left <= right) {

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

if (abs(num1 - arr2[mid]) <= d) {

found = true;

break;

} else if (num1 < arr2[mid]) {

right = mid - 1;

} else {

left = mid + 1;

}

}

// 如果没有找到满足条件的元素，则增加计数器

if (!found) {

cnt++;

}

}

return cnt;

}

};

复杂度分析

假设arr1中元素个数为n，arr2中元素个数为m。

时间复杂度：给arr2排序的时间代价是O(mlogm)，对于 arr1 中的每个元素都在arr2中二分的时间代价是O(nlogm)，故渐进时间复杂度为O((n+m)logm)。

空间复杂度：这里没有使用任何的辅助空间，故渐进空间复杂度为O(1)。