L1483 A-B 数对 ✫✫

题目描述

给出一串正整数数列以及一个正整数 C，要求计算出所有满足 $A - B = C$ 的数对 (A, B) 的个数。

注意：不同位置的数字如果数值相同，算作不同的数对。

输入输出格式

输入：第一行包含两个正整数 N 和 C（$1 \le N \le 2 \times 10^5, 1 \le C < 2^{30}$）。第二行包含 N 个正整数，表示数列中的元素（数值 $< 2^{30}$）。

输出：输出一个整数，表示满足条件的数对个数。

输入示例	输出示例
4 1 1 1 2 3	3

样例解释

输入的数列为 $\{1, 1, 2, 3\}$，目标差值 $C = 1$。我们需要寻找满足 $A - B = 1$ 的数对。满足条件的数对共有以下 3 对：

序号	数对 (A, B)	详细说明
1	$(2, 1)$	数组中**第3个**元素和**第1个**元素
2	$(2, 1)$	数组中**第3个**元素和**第2个**元素
3	$(3, 2)$	数组中**第4个**元素和**第3个**元素

*注：虽然第1和第2组数对的数值看起来一样，但因为 B 分别对应了数组中**不同下标位置**的元素，所以算作两个不同的答案。

算法分析

1. **问题分析**
   题目要求统计满足 $A - B = C$ 的数对。这个等式可以变换为 $B = A - C$。这意味着，对于数列中的每一个数，如果我们把它当作 A，那么我们只需要去查找数列中有多少个数值等于 $A - C$ 的数即可。
2. **算法选择**
   （1） **解法一 - 暴力枚举 (TLE)**：使用双重循环枚举所有的 A 和 B，判断差值是否为 C。复杂度为 $O(N^2)$。由于 N 最大可达 2×10^5，计算量高达 4×10^{10}，显然会超时。

   （2） **解法二 - Map 统计**： 利用 `std::map` 记录每个数字出现的次数。然后遍历每个 A，使用map获取 B = A　C 在数组中的数量。时间复杂度 $O(N \log N)$。

   （3） **解法三 - 排序 + 二分查找**： 先将数组排序，然后对于每个 A，使用二分查找快速定位 $B = A - C$ 在数组中的数量。时间复杂度同样为 $O(N \log N)$，但不需要 Map 的额外空间开销，常数更小。

3. **实现思路**

- **思路一：Map 统计**

（1） 遍历数组，用 map 统计每个数出现的频率。

```
mp[a[i]]++;
```

（2） 再次遍历，累加 ‘mp[a[i] - c]‘ 即为答案。

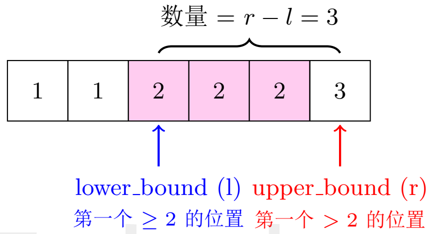
```
cnt += mp[a[i] - c];
```

- **思路二：排序 + 二分查找**

（1） **排序：**首先对数组进行升序排序，这是二分查找的前提。

```
std::sort(a.begin(), a.end()); // 先对数组进行排序
```

（2） **二分查找（图解）：**假设排序后的数组为 {1, 1, 2, 2, 2, 3}，我们要查找目标值 **2** 的数量。



– lower_bound：指向第一个 **大于等于** 目标值的位置（开始位置）。
– upper_bound：指向第一个 **大于** 目标值的位置（结束位置的下一个）。
– **目标数量** = upper_bound 的下标 - lower_bound 的下标。

```
auto l = std::lower_bound(a.begin(), a.end(), a[i] - c);
auto r = std::upper_bound(a.begin(), a.end(), a[i] - c);
cnt += r - l;
```


拓展思考

1. **优化 - 双指针法** ($O(N)$)
在解法三中，我们先排序再二分，总复杂度是 $O(N \log N)$。

- 思考一下：既然数组已经有序了，当我们枚举的 A 增大时，对应的 B $(= A - C)$ 肯定也是单调递增的。
- 是否可以使用两个指针（或三个指针，考虑到重复元素），只需遍历一遍数组就能统计出答案？这样可以将排序后的搜索过程降低到 $O(N)$。

2. **经典变式 - 两数之和** ($A + B = C$) 如果题目改为计算 $A + B = C$ 的数对个数,该做哪些修改