2364. Count Number of Bad Pairs

You are given a **0-indexed** integer array `nums`. A pair of indices `(i, j)` is a **bad pair** if `i < j` and `j - i != nums[j] - nums[i]`.

Return *the total number of **bad pairs** in* `nums`.

Example 1:

```
Input: nums = [4,1,3,3]
Output: 5
Explanation: The pair (0, 1) is a bad pair since 1 - 0 != 1 - 4.
The pair (0, 2) is a bad pair since 2 - 0 != 3 - 4, 2 != -1.
The pair (0, 3) is a bad pair since 3 - 0 != 3 - 4, 3 != -1.
The pair (1, 2) is a bad pair since 2 - 1 != 3 - 1, 1 != 2.
The pair (2, 3) is a bad pair since 3 - 2 != 3 - 3, 1 != 0.
There are a total of 5 bad pairs, so we return 5.
```

Example 2:

```
Input: nums = [1,2,3,4,5]
Output: 0
Explanation: There are no bad pairs.
```

Constraints:

- `1 <= nums.length <=` 10^5
- `1 <= nums[i] <=` 10^9

2364. Count Number of Bad Pairs

```
/*
   Hash map+Math+counting
   Time complexity: O(2n)
   Space complexity: O(n)
*/
typedef long long ll;
class Solution {
   public:
      ll countBadPairs(std::vector<int>& nums){
         ll n=nums.size();

         //  Create a hash map to store the frequency of good pairs
         // good pair is nums[j]-j==nums[i]-i => nums[j]-j==nums[i]-i
         // Initiate the hash map with -1, because the difference of nums[j]-j=nums[i]-i
         // such that i==j is not counted.
         struct defaut_value{int val=-1;};
         std::unordered_map<int,defaut_value> freq;
         for(int i=0;i<n;++i) freq[nums[i]-i].val++;

         // Determine the number of good pairs
         ll good_pairs=0;
         // For each nums[j]
         for(int j=0;j<n;++j){
            // If the frequency of nums[j]-j>0, add it to the answer and reduce it by 1
            // because that difference nums[j]-j will not be counted any more, if we encounter
            // the same difference one more time
            good_pairs+=freq[nums[j]-j].val>0?freq[nums[j]-j].val--:0;
         }

         // Count the total number of pairs (good and bad)
         ll total_pairs=n*(n-1)/2;

         // Number of bad pairs = total number of pairs-number of good pairs
         return total_pairs-good_pairs;
      }
};
```

2364. Count Number of Bad Pairs

```
/*
   single pass: Hash map+Math+counting
   Time complexity: O(n)
   Space complexity: O(n)
*/
typedef long long ll;
class Solution {
   public:
      ll countBadPairs(std::vector<int>& nums){
         ll n=nums.size();

         //  Create a hash map to store the frequency of good pairs
         // good pair is nums[j]-j==nums[i]-i => nums[j]-j==nums[i]-i
         std::unordered_map<int,ll> freq;

         ll ans=0;
         // For each nums[i]
         for(int i=0;i<n;++i){
            // Determine the number of good pairs that it can make it
            ll good_pairs=freq[nums[i]-i];

            // Each element at position i, can make a total of i pairs
            // so, to find the number of bad pairs we subtract the number of
            // good pairs from the total number of pairs that nums[i] can make
            ans+=i-good_pairs;

            // nums[i]-i is a good pair by iteself
            // number of good pairs including nums[i] is total pairs-bad pairs
            // => i-(i-good pairs) + 1 = good pairs+1
            // i: total number of pairs made by nums[i]
            // i-good pairs; #bad pairs before nums[i]
            // +1: nums[i]-i is a good pair
            freq[nums[i]-i]=good_pairs+1;
         }

         return ans;
      }
};
```