E - 1512. Number of Good Pairs

Given an array of integers nums, return the number of good pairs.
A pair (i, j) is called good if nums[i] == nums[j] and i < j.

Example 1:

Input: nums = [1,2,3,1,1,3]
Output: 4

Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

Example 2:

Input: nums = [1,1,1,1]
Output: 6

Explanation: Each pair in the array are good.

Example 3:

Input: nums = [1,2,3]
Output: 0

Constraints:
1 <= nums.length <= 100
1 <= nums[i] <= 100

Solutions:

Approach-1 => Brute Force

- Initialize a variable count to 0.
- Use two nested loops to iterate through all possible pairs of indices (i, j) where i < j.
- If nums[i] is equal to nums[j], increment the count by 1.
- After both loops finish, return the count.

Code :

```
class Solution {
    public int numIdenticalPairs(int[] nums) {
        int count=0;
        for(int i=0;i<nums.length;i++){
            for(int j=i+1;j<nums.length;j++){
                if(nums[i]==nums[j])
                    count++;
            }
        }
        return count;
    }
}
```

Approach-1 => HashMap

- Initialize an empty hash map num_count.
  Initialize a variable count to 0.
- Iterate through the array nums from left to right.
- For each element num, check if it exists in the num_count hash map.
- If it exists, increment count by the value associated with num in the hash map, and increment the value by 1.
- If it doesn't exist, add num to the hash map with a value of 1.
- After iterating through the array, return count.

Code :

```
class Solution {
    public int numIdenticalPairs(int[] nums) {
        int count =0;
        Map<Integer,Integer> numsFreq=new HashMap<>();
        for(var num:nums){
            count += numsFreq.getOrDefault(num,0);
            numsFreq.put(num,numsFreq.getOrDefault(num,0)+1);
        }
        return count;
    }
}
```