Unique Number of Occurrences LeetCode

Given an array of integers arr, return true if the number of occurrences of each value in the array is unique or false otherwise.

Input: arr = [1,2,2,1,1,3]

Output: true

Input: arr = [1,2]

Output: false

Input: arr = [-3,0,1,-3,1,1,1,-3,10,0]

Output: true

Approach 1: Using Hashmap and Set to track the count of each Array elements.

This approach takes a reference to a vector of integers as input and returns a boolean value.

It initializes an unordered_map count to store the count of occurrences for each unique value in the array.

It also initializes an unordered_set unique to keep track of the unique occurrence counts.

It iterates over each element i in the input array and increments its count in the count map.

After counting the occurrences, it iterates over the key-value pairs in the count map.

For each occurrence count i.second, it attempts to insert it into the unique set using unique.insert(i.second).

The .second accesses the value (occurrence count) in the key-value pair.

The .insert(i.second) function returns a pair consisting of an iterator and a boolean indicating if the insertion was successful.

If the insertion was not successful (i.e., the occurrence count already exists in the unique set), it means there is a duplicate occurrence count, so the function returns false.

If the loop completes without encountering any duplicate occurrence counts, the function returns true.

Time Complexity:

The time complexity of this code is O(n), where n is the size of the input array. The loop that counts the occurrences and inserts them into the map takes O(n) time. The loop that checks for unique occurrence counts also takes O(n) time in the worst case, as it iterates over all the elements in the map.

Space Complexity:

The space complexity is O(n) as well. The count map can store at most n distinct keys, and the unique set can also store at most n unique occurrence counts. Hence, the overall space complexity is linear in the size of the input array.