Counting frequencies of array elements

Given an array which may contain duplicates, print all elements and their frequencies.

Examples:

```
Input :  arr[] = {10, 20, 20, 10, 10, 20, 5, 20}
Output : 10 3
         20 4
         5  1

Input : arr[] = {10, 20, 20}
Output : 10 1
         20 2
```

A **simple solution** is to run two loops. For every item count number of times, it occurs. To avoid duplicate printing, keep track of processed items.

```
// CPP program to count frequencies of array items
#include <bits/stdc++.h>
using namespace std;

void countFreq(int arr[], int n)
{
    // Mark all array elements as not visited
    vector<bool> visited(n, false);

    // Traverse through array elements and
    // count frequencies
    for (int i = 0; i < n; i++) {

        // Skip this element if already processed
        if (visited[i] == true)
```

```
            continue;

        // Count frequency
        int count = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                visited[j] = true;
                count++;
            }
        }
        cout << arr[i] << " " << count << endl;
    }
}

int main()
{
    int arr[] = { 10, 20, 20, 10, 10, 20, 5, 20 };
    int n = sizeof(arr) / sizeof(arr[0]);
    countFreq(arr, n);
    return 0;
}
```

Output

```
10 3
20 4
5 1
```

Time Complexity : O(n2)

Auxiliary Space : O(n)

An **efficient solution** is to use hashing.

```
// CPP program to count frequencies of array items
#include <bits/stdc++.h>
using namespace std;

void countFreq(int arr[], int n)
{
```

```
    unordered_map<int, int> mp;

    // Traverse through array elements and
    // count frequencies
    for (int i = 0; i < n; i++)
        mp[arr[i]]++;

    // Traverse through map and print frequencies
    for (auto x : mp)
        cout << x.first << " " << x.second << endl;
}

int main()
{
    int arr[] = { 10, 20, 20, 10, 10, 20, 5, 20 };
    int n = sizeof(arr) / sizeof(arr[0]);
    countFreq(arr, n);
    return 0;
}
```

Output

```
5 1
10 3
20 4
```

Time Complexity : O(n)

Auxiliary Space : O(n)

In above efficient solution, how to print elements in same order as they appear in input?

```
// CPP program to count frequencies of array items
#include <bits/stdc++.h>
using namespace std;

void countFreq(int arr[], int n)
{
    unordered_map<int, int> mp;
```

```
    // Traverse through array elements and
    // count frequencies
    for (int i = 0; i < n; i++)
        mp[arr[i]]++;

    // To print elements according to first
    // occurrence, traverse array one more time
    // print frequencies of elements and mark
    // frequencies as -1 so that same element
    // is not printed multiple times.
    for (int i = 0; i < n; i++) {
      if (mp[arr[i]] != -1)
      {
          cout << arr[i] << " " << mp[arr[i]] << endl;
          mp[arr[i]] = -1;
      }
    }
}

int main()
{
    int arr[] = { 10, 20, 20, 10, 10, 20, 5, 20 };
    int n = sizeof(arr) / sizeof(arr[0]);
    countFreq(arr, n);
    return 0;
}
```

Output

```
10 3
20 4
5 1
```

Time Complexity : O(n)

Auxiliary Space : O(n)

This problem can be solved in Java using HashMap. Below is the program.

```
// C++ program to count frequencies of
// integers in array using Hashmap
#include <bits/stdc++.h>
using namespace std;

void frequencyNumber(int arr[],int size)
{

  // Creating a HashMap containing integer
  // as a key and occurrences as a value
  unordered_map<int,int>freqMap;

  for (int i=0;i<size;i++) {
    freqMap[arr[i]]++;
  }

  // Printing the freqMap
  for (auto it : freqMap) {
    cout<<it.first<<" "<<it.second<<endl;
  }
}

int main()
{
  int arr[] = {10, 20, 20, 10, 10, 20, 5, 20};
  int size = sizeof(arr)/sizeof(arr[0]);
  frequencyNumber(arr,size);
}
```

Output

```
5 1
10 3
20 4
```