CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

EXPERIMENT – 3

Student Name: Gaganjot Singh **UID: 22BCS14843**
Branch: BE-CSE **Section/Group:** 22BCS_JT_802-B
Semester: 5th **Date of Performance:** 1 August 2024
Subject Name: Design & Analysis of Algorithms **Subject Code:** 22CSH-311

1. **Aim:** Code to find frequency of elements in a given array in O(n) time complexity.

2. **Objective:** To find the frequency of an element in a given array in O(n) time complexity using unordered hash maps.

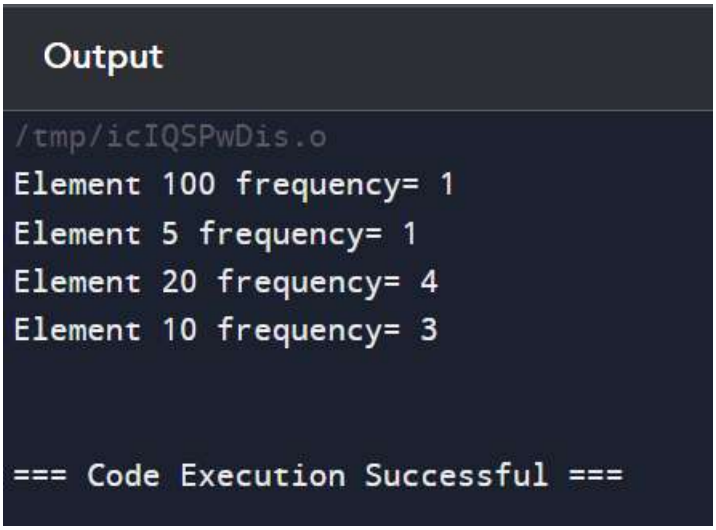
3. **Algorithm**

- Start.
  - Include Headers:
    #include
    <bits/stdc++.h> using
    namespace std;
  - Define countFreq Function:
    Input: Integer array arr and its size n.
    Declare an unordered map mp of type <int, int>.
  - Count
    Frequencies:
    For i = 0 to
    n-1:
    Increment the count of arr[i] in mp: mp[arr[i]]++.
  - Print
    Frequencies:
    For each x in
    mp:
    Print: "Element x.first frequency= x.second".
  - Define main Function:
    Declare and initialize array arr with elements.
    Calculate the size of the array n using: n = sizeof(arr) / sizeof(arr[0]).
    Call countFreq(arr, n).
  - End .

4. Implementation/Code:

```
#include <bits/stdc++.h>
#include<iostream>
using namespace std;
void countFreq(int arr[], int n)
{
    unordered_map<int, int> mp;

    for (int i = 0; i < n; i++)
    {
        mp[arr[i]]++;
    }

    for (auto x : mp)
    {
        cout <<"Element "<< x.first << " frequency= " << x.second << endl;
    }
}

int main()
{
    int arr[] = { 10, 20, 20, 10, 10, 20, 5, 20,100 };

    int n = sizeof(arr) / sizeof(arr[0]);

    countFreq(arr, n);

    return 0;

}
```

5. **Output**



6. **Time Complexity:**
   - The time complexity of this method is O(n) because it involves a single linear pass through the array, while the space complexity is O(n) due to the use of an unordered_map for storage, where n is the number of elements in the array.
7. **Learning Outcomes:**
   - Learned about the unordered hash maps in the C++ for efficient key value storage and retrieval.
   - Learn how to analyze the time complexity of algorithms.
   - Understand the concept of counting the frequency of elements in an array.
   - Understand how to use data structures effectively to optimize performance.