**Experiment Title 1.3**

**Student Name: Sahul Kumar Parida UID: 20BCS4919**

**Branch: CSE Section/Group: WM-904/B**

**Semester: 5th Date of Performance: 21/08/2022**

**Subject Name: Design and Analysis Algorithm Lab**

**Subject Code: 20CSP-312**

**1. Aim/Overview of the practical:**

In O(n) time complexity, find the frequency of elements in a given array.

**2. Task to be done/ Which logistics used:**

* Using Brute Force Approach.
* Using Counting sort Algorithm.
* Using Best Algorithm HashMaps.

**3. Algorithm/Flowchart (For programming based labs):**

1. Brute Force Algorithm: -

* Get an input from user to get the size.
* Create an array of elements and insert the elements as entered by the user.
* Create a for loop using variable i to iterate over these elements from 0 to size.
* Create a variable count and initialize it to zero in this for loop.
* Inside for loop create another loop using variable j from i to size.
* If the element at ith position is equal to the element at jth position increment the count and change the jth element to -1.
* Exiting the inner loop, element is not equal to -1, show the element and count.

1. Efficient Approach Using Counting sort Algorithm: -

* Get an input from user to get the size.
* Create an array of elements named user and insert the elements as entered by the user.
* Create a variable to store the maximum value in this array and find the maximum while taking the input only.
* Now create another array named count with size equal to the maximum element in the user array.
* Iterate over this array from 0 to n.
* Now we need to store the count of each element on the index of the element.

e.g.: - if user[0]=3 than we will change count array to count[3]++ or count[user[0]]++;

* After exiting this loop we would iterate on the count array from 0 to the maximum element found in user array.
* If the count array value is not equal to zero we shall print the element and its count.

1. Best Approach Using HashMap

* Get an input from user to get the size.
* Create an array of elements and insert the elements as entered by the user.
* Create an unordered map named as count.
* Create a for loop using variable i to iterate over these elements from 0 to size.
* Iterate over the array size and set the map’s key to the value of element in user’s array and increment the count of that place.
* Use another loop to show the desired result.

**4. Steps for experiment/practical/Code:**

Brute Force Approach

#include<bits/stdc++.h>

using namespace std;

int main()

{

cout<< "SAHUL KUMAR PARIDA"<<endl;

cout<< "20BCS4919"<<endl;

int n;

cout<<"Enter the size of array: ";

cin>>n;

vector<int> arr(n);

for(int i=0;i<n;i++)

{

int temp;

cin>>temp;

arr[i]=temp;

}

for(int i=0;i<n;i++)

{

int cnt=1;

for(int j=i+1;j<n;j++)

{

if(arr[i]==arr[j])

{

arr[j]=-1;

cnt++;

}

}

if(arr[i]!=-1)

cout<<arr[i] << " OCCURS "<<cnt <<" times"<<endl;

}

}

Counting Sort Algorithm Approach

#include<bits/stdc++.h>

using namespace std;

int main()

{

cout<< "SAHUL KUMAR PARIDA"<<endl;

cout<< "20BCS4919"<<endl;

int n;

cout<<"Enter the size of array: ";

cin>>n;

vector<int> arr(n);

int maxi=-1;

for(int i=0;i<n;i++)

{

int temp;

cin>>temp;

arr[i]=temp;

maxi=max(arr[i],temp);

}

vector<int> count(maxi);

for(int i=0;i<n;i++)

{

count[arr[i]]++;

}

for(int i=0;i<=maxi;i++)

{

if(count[i]!=0)

{

cout<<"element "<<i <<" occurs "<<count[i]<<" times"<<endl;

}

}

}

HashMap Approach

#include<bits/stdc++.h>

using namespace std;

int main()

{

cout<< "SAHUL KUMAR PARIDA"<<endl;

cout<< "20BCS4919"<<endl;

int n;

cout<<"Enter the size of array: ";

cin>>n;

vector<int> arr(n);

for(int i=0;i<n;i++)

{

int temp;

cin>>temp;

arr[i]=temp;

}

unordered\_map<int,int> map;

for(int i=0;i<n;i++)

{

map[arr[i]]++;

}

for(auto i:map)

{

cout<< i.first <<" number occurs "<<i.second <<" times"<<endl;

}

}

**5. Observations/Discussions/ Complexity Analysis:**

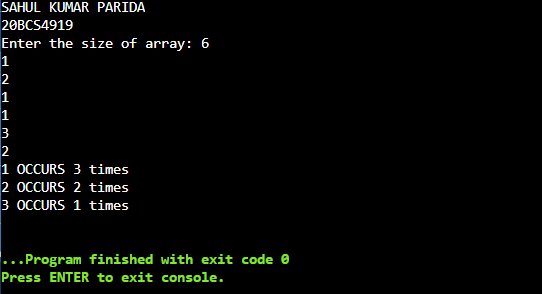
Brute Force has a time complexity of O(N^2) where N is the array. It has a space works of O(N) used to initialize the array.

The Counting Sort Algorithm has a time complexity of O(max(N, max element)) since we are iterating through both N and max element also It has the same Space Complexity.

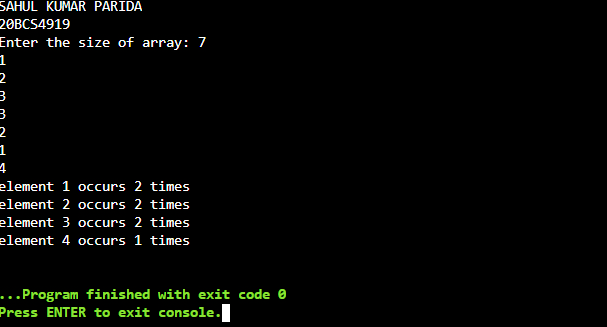
The HashMap approach has the Best time and space complexity which is O(N) as insertion in unordered HashMap takes place in O(1) time though if we have used an ordered map the insertion time would have been increased to O(log (n)).

**6. Result/Output/Writing Summary:**

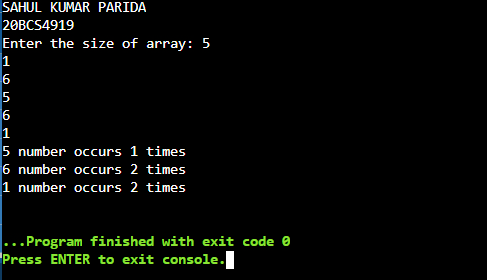
Brute Force Approach



Counting Sort Algorithm Approach



HashMap Approach



**Learning outcomes (What I have learnt):**

1. Understanding the question to find frequencies of number.

2. Use of Counting Algorithm

3. Using different approaches for same problem and identifying the optimized approach.

4. Use of HashMap and understanding space and time complexities.