Design and Analysis of Algorithms Assignment 1

Members:
Kaushal Kumar IIT2019030
Sarvesh IIT2019031
Aarushi IIT2019032

B.Tech 4th Semester Department of Information Technology
Indian Institute of Information Technology, Allahabad

Abstract:
In this paper we have prepared a frequency Distribution Table along with location information from a randomly generated matrix of any size and 2-digit positive integers.
We have also discussed the time complexity of the algorithm by both Apriori and Aposteriori analysis.

Keywords: *Map,Frequency Distribution table, Hashing, 2-D vector, Pair*

1) *INTRODUCTION:*

According to the given problem statement we have to generate a 2D matrix, with 2 digit positive integer value and accordingly generate its frequency distribution table along with the position of each value as situated in the generated matrix.
Here, we have used hashing techniques to store the frequencies of the elements in an array and mapped their values to the positions in which they are situated.
To store the key/value pair, you can use a simple array like a data structure where keys (integers) can be used directly as an index to store values. However, in cases where the keys are large and cannot be used directly as an index, you should use a hash function.
The mapping is done using a 2D vector and pair to store x-y indexes.

2) *APPLICATION:*

Basically, Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

For the implementation of hashing concept, we can use hashmap or simply we can use an array to keep track of the values that have occurred during the iteration of the given 2-D Array but in some cases the array won't work, like in case when maximum element is greater than the MAX size of Array(~10^6 - 10^7).

So here we can use the map which has a key,value pair so that the unnecessary wastage of size can be prevented.

The purpose of hashing is to achieve search, insert and delete complexity to O(1).

Some standard question hashing algorithm can be used is:-

- Pair with given sum
- Rabin-Karp Algorithm for string matching

3) ALGORITHM DESIGN:-

Our algorithm to plot the histogram of various experimental values can be done in the following steps:-

Algorithm-1

Firstly we have to take the size of the matrix as input from the user, after the size we can generate the matrix using the rand() function in c++, within the range of 10-99 (2-digit).

1.) Use a frequency array(freq) to store the frequencies of the elements of the matrix.

2.) Store the indexes of elements using a 2D vector of type int and a pair of int values to store x and y indexes.

3.) In the vector store the values correspondingly, and display it through looping over the entire 2D vector.

Algorithm-2

In this algorithm we are making a slight change. We are using Map instead of frequency array which makes our algorithm more efficient with respect to space.

4) PSEUDO CODE:-

```
function Main()
max ←1000000
Get n,m
a[n+1][m+1]
Get random inputs
vector<vector<pair<int,int> > >v(max)
map<int,int>m
for i←1 to n do
for j←1 to m do
m[a[i][j]]←m[a[i][j]]+1
v[a[i][j]].push_back(i,j))
for i←1 to 100 do
if(m[i]) then
for j←1 to v[i].size()do
print v[i][j].first
print v[i][j].second
return 0
```

5) COMPLEXITY ANALYSIS:-

1) **Time complexity**:-

In all the cases**,** time complexity will always be same as **O(n*m)**. As we have to do the 3 computations: first begin the generation of random arrays. Second being the computation of frequency

array and the last one is print the frequency array.

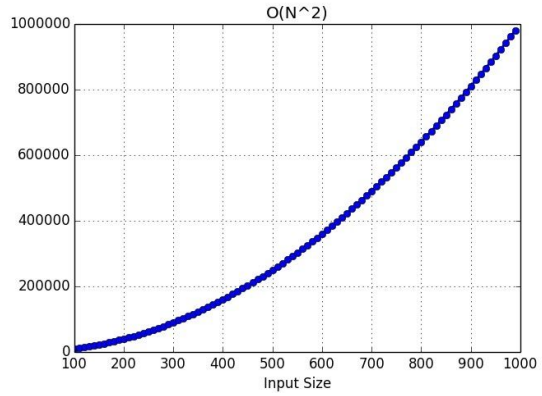
For first and second Computation, time complexity is always **O(n*m)**.
For third computation, time complexity is equals to the **O(no. of distinct elements*respective occurrence)** and ultimately this comes as the **O(n*m)**
So overall time complexity is **O(n*m)**.

2) **Space complexity:**
   The complexity of both the algorithm is almost same for small inputs which is generally O(n*m) but for large inputs algorithm 2 is more efficient than algorithm 1 because it will reduce unnecessary wastage of memory and solve the problems in O(n*m) complexity.

6) *EXPERIMENTAL STUDY:*

Apriori Analysis:
Apriori analysis means, analysis is performed prior to running it on a specific system. This analysis is a stage where a function is defined using some theoretical model. Hence, we determine the time and space complexity of an algorithm by just looking at the algorithm rather than running it on a particular system with a different memory, processor, and compiler.

So, as we discussed under the heading complexity analysis we arrived at the conclusion that time complexity is equals to O(n*m) and space complexity is also equals to the O(n*m).



Aposteriori Analysis:

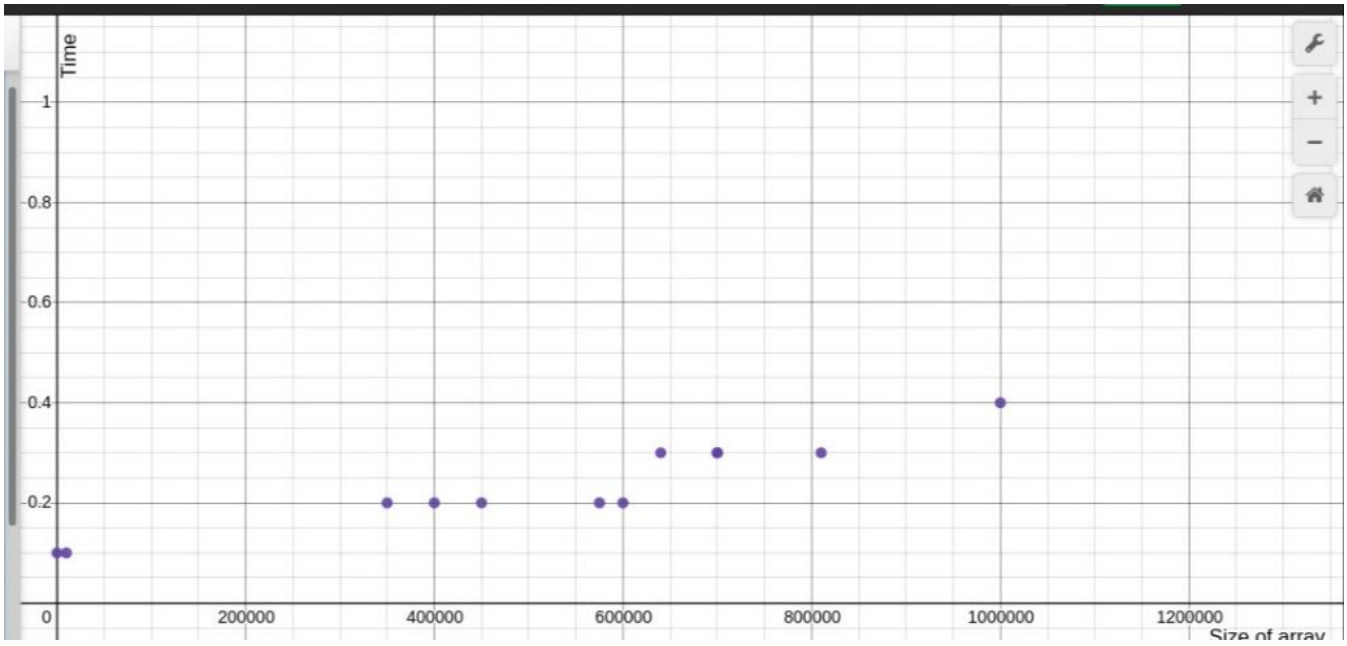
Apostiari analysis of an algorithm means we perform analysis of an algorithm only after running it on a system. It directly depends on the system and changes from system to system.

So for the a aposteriori analysis of the algorithm, we have run our code on the compiler and get values of the time by specifying the value of n and m and different value of the time had occur.

S.No	Sizeof array(n*m)	Time Taken
1	1(1,1)	0.10000
2	10,000 (100,100)	0.10000
3	3,50,000 (350,1000)	0.200000
4	4,00,000 (400,1000)	0.200000
5	5,75,000 (575,1000)	0.200000

6	6,40,000 (640,1000)	0.30000001
7	7,00,000 (700, 1000)	0.30000001
8	8,10,000 (810,1000)	0.30000001
9	10,00,000 (1000,1000)	0.40000000
10	4,50,000 (450,1000)	0.20000000
11	6,00,000 (600,1000)	0.20000000
12	7,00,000 (700,1000)	0.30000001

Hence the graphical representation is given as below:-



7) CONCLUSION:

Therefore, here we learnt about hashing techniques using maps and another using a frequency array.
And we conclude that in order to make the code space efficient we must use the map method approach.

8) REFERENCES:

[1]:-https://www.geeksforgeeks.org/find-frequency-number-array/
[2]:-https://www.geeksforgeeks.org/vector-of-vectors-in-c-stl-with-examples/