DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

NAAC GRADE A+ Accredited University

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE/IT-5th Sem)

Design and Analysis of Algorithms

Subject Code: 23CSH-301/ITH-301

Submitted to:

Faculty name: MD. Shaqlain

Submitted by:

Name: Reetesh Sahu
UID: 23BCS10018

Section: Krg-1 A

Group: A

INDEX

Ex. No	List of Experiments	Date	Conduct (MM: 12)	Viva (MM: 10)	Worksheet (MM: 8)	Total (MM:30)	Remarks/Signature
1.1	Analyze if stack Isempty, Isfull and if elements are present then return top element in stacks using templates and also perform push and pop operation in stack.	21/07/25					
1.2	Develop a program for implementation of power function and determine that complexity should be O (log n).	01/08/25					
1.3	Evaluate the complexity of the developed program to find frequency of elements in a given array.	08/08/25					
1.4	i. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and end of Singly Linked List. ii. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.						

2.1	Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.						
2.2	Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.						
2.3	Develop a program and analyze complexity to implement 0-1 Knapsack using Dynamic Programming.						
3.1	Develop a program and analyze complexity to find shortest paths in a graph with positive edge weights using Dijkstra's algorithm.						
3.2	Develop a program and analyze complexity to find all occurrences of a pattern P in a given string S.						
3.3	Lab Based Mini Project.						

Experiment No: 1.3

Student Name: Reetesh Sahu
UID: 23BCS10018
Branch: CSE
Section/Group: Krg-1A
Semester: 5th
Date of Performance: 08/08/25
Subject Name: Design analysis and algorithm
Subject Code: 23CSH-301

Aim:

Code to find frequency of elements in a given array in O(N) time complexity.

Procedure:

Step 1: Input

- Read an array arr of size n.

Step 2: Setup

- Create a hash map (dictionary) freq to maintain mapping: **element → count**.

Step 3: Traverse Array

- Loop through each index i from 0 to n-1:
  - If arr[i] is not present in freq, insert it with value 1.
  - Otherwise, increase its value by 1.

Step 4: Result

- After the full traversal, freq will store every unique element with the number of times it appears.

Step 5: Output

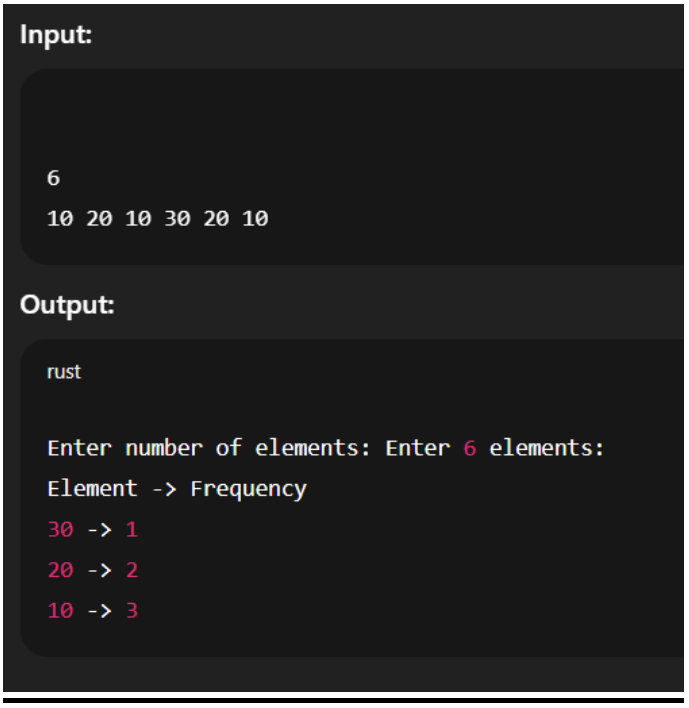
- For each (key, value) in freq, display in the format:
- element → frequency

Code:

```
#include <iostream>
#include <vector>
#include <unordered_map>
using namespace std;

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    vector<int> arr(n);
```

```
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    unordered_map<int, int> countMap;

    // Count frequencies
    for (int val : arr) {
        countMap[val]++;
    }

    cout << "\nElement -> Frequency\n";
    for (auto &entry : countMap) {
        cout << entry.first << " -> " << entry.second << endl;
    }

    return 0;
}
```

Output:



◇ Time Complexity Analysis

- Input phase: Reading n elements into the array takes O(n) time.
- Frequency calculation: Each update in unordered_map works in O(1) average time, so counting for all elements costs O(n).
- Output phase: Printing each element–frequency pair depends on the number of unique items. In the worst case, there are n unique values, so printing also takes O(n).

👉 Hence, the overall time complexity = O(n) (on average, assuming hash operations are constant time).

◇ Learning Outcomes

1. Gained practical understanding of arrays + hash maps (unordered_map).
2. Learned how to store and update key–value pairs quickly.
3. Understood that hashing allows frequency counting to be done in linear time O(n) on average.