Experiment 4 (Searching And Sorting)

Student Name: SANSKAR AGRAWAL **UID: 20BCS5914**
Branch: BE CSE **Section/Group: 20BCSMM_806B**
Semester: 5th **Date of Performance: 18.10.2022**
Subject Name: Competitive Coding **Subject Code: 20CSP_314**

1. Aim/Overview of the practical:

a. Fraudulent Activity Notifications.
b. Full Counting Sort.

2. Task to be done/ Which logistics used:

a. Hacker Land National Bank has a simple policy for warning clients about possible fraudulent account activity. If the amount spent by a client on a particular day is greater than or equal to 2x the client's median spending for a trailing number of days, they send the client a notification about potential fraud. The bank doesn't send the client any notifications until they have at least that trailing number of prior days' transaction data. Given the number of trailing days d and a client's total daily expenditures for a period of n days, determine the number of times the client will receive a notification over all n days.

b. Use the counting sort to order a list of strings associated with integers. If two strings are associated with the same integer, they must be printed in their original order, i.e. your sorting algorithm should be stable. There is one other twist: strings in the first half of the array are to be replaced with the character - (dash, ascii 45 decimal). Insertion Sort and the simple version of Quicksort are stable, but the faster in-place version of Quicksort is not since it scrambles around elements while sorting.

3. Algorithm/Flowchart

a. Fraudent Activity Notifications:

i. Create a class Solution.
ii. Enter the details of transactions.
iii. Check for the fraudent activity.
iv. Add new transactions.
v. Calculate the total number of transactions.
vi. Print with the suitable message.

b. Full Counting Sort:

i. Create a class Solution.
ii. Sort the list using quickable sort.
iii. Print the suitable message.

4. Steps for experiment/practical/Code:

a. Fraudent Activity Notifications:

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int d = input.nextInt();
        int notifications = 0;
        Queue<Integer> queue = new LinkedList<>();
        int[] pastActivity = new int[201];
        for(int i = 0; i < d; i++)
        {
            int transaction = input.nextInt();
            queue.offer(transaction);
            pastActivity[transaction] = pastActivity[transaction]+1;
        }

        for(int i = 0; i < n-d; i++)
        {
            int newTransaction = input.nextInt();
            if(newTransaction >= (2* median(pastActivity, d)))
                notifications++;
            int oldestTransaction = queue.poll();
            pastActivity[oldestTransaction] = pastActivity[oldestTransaction]-1;
            queue.offer(newTransaction);
            pastActivity[newTransaction] = pastActivity[newTransaction]+1;
        }

        System.out.println(notifications);
    }
```

```
static double median(int[] array, int elements)
{
    int index = 0;

    if(elements % 2 == 0)
    {
        int counter = (elements / 2);

        while(counter > 0)
        {
            counter -= array[index];
            index++;
        }
        index--;
        if(counter <= -1)
            return index;
        else
        {
            int firstIndex = index;
            int secondIndex = index+1;
            while(array[secondIndex] == 0)
            {
                secondIndex++;
            }
            return (double) (firstIndex + secondIndex) / 2.0;
        }
    }
    else
    {
        int counter = (elements / 2);

        while(counter >= 0)
        {
            counter -= array[index]; index++;
        }
```

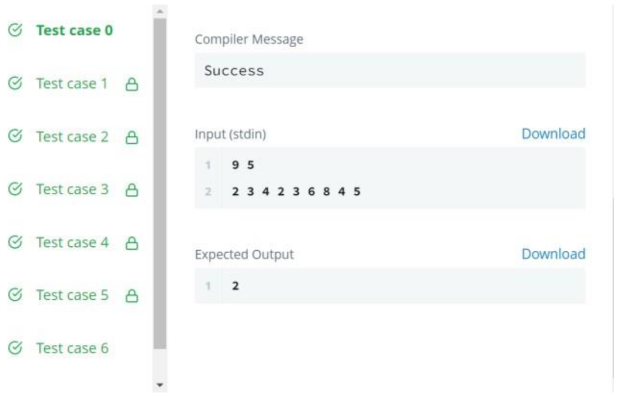
```
            return (double) index-1;
        }
    }


    static void printArray(int[] array)
    {
        System.out.println("Array");
        for(int i = 0; i < array.length; i++)
        {
            if(array[i] > 0)
                System.out.println(i+" : "+array[i]);
        }
    }
}
```

b. Full Counting Sort:

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int entries = scanner.nextInt();
        StringBuilder[] freqs = new StringBuilder[100];
        for (int i = 0; i < entries; i++) {
            int idx = scanner.nextInt();
            if (i < entries / 2) {
                freqs[idx] = freqs[idx] == null ? new StringBuilder("-")
                                                : freqs[idx].append(" -");
                scanner.next();
            } else {
                freqs[idx] = freqs[idx] == null ? new StringBuilder(scanner.next())
: freqs[idx].append(" ").append(scanner.next());
                }
            }
            for (int i = 0; i < freqs.length; i++) {
                if (freqs[i] != null) {
                    System.out.print(freqs[i].toString());
                    System.out.print(" ");
                }
            }
            System.out.println();
        }


}
```

5. Result/Output/Writing Summary:

a. Fraudent Activity Notifications:


Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

```
Success
```

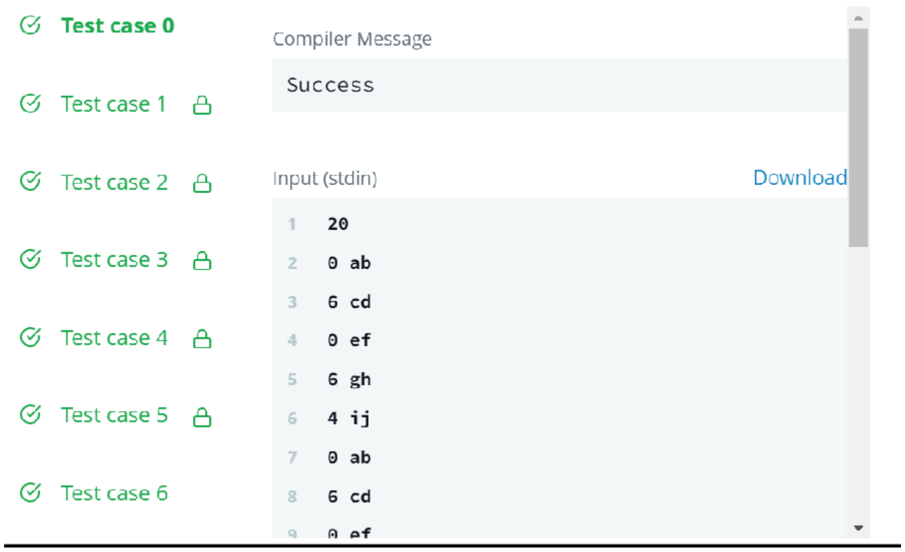
Input (stdin) Download

```
9 5
2 3 4 2 3 6 8 4 5
```

Expected Output Download

```
2
```

b. Full Counting Sort:



Compiler Message

```
Success
```

Input (stdin)

```
20
0 ab
6 cd
0 ef
6 gh
4 ij
0 ab
6 cd
0 ef
```

Learning outcomes (What I have learnt):

a. Learnt about Vectors.
b. Learnt about searching and sorting techniques.
c. Got an overview of the type of questions on hacker-rank.
d. Get to know about crucial test cases.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			