

## **CSE 4304: Data Structures**

### **Lab: 02**

**Topic:** Problems related to growth of function, arrays and searching.

**Instructions:**

- Use appropriate comments in your code. This will help you to easily recall the solution in future.
- If you are stuck, please ask for help. **Please !**

### **Task 1**

You are given an array of integers **A**, you need to find the maximum sum that can be obtained by picking some non-empty subset of the array. If there are many such non-empty subsets, choose the one with the maximum number of elements. Print the maximum sum and the number of elements in the chosen subset.

#### **Input:**

The first line contains an integer **N**, denoting the number of elements of the array. Next line contains space-separated integers, denoting the elements of the array.

#### **Output:**

Print 2 space-separated integers, the maximum sum that can be obtained by choosing some subset and the maximum number of elements among all such subsets which have the same maximum sum.

#### **Constraints:**

$$1 \leq N \leq 10^5$$

$$-10^9 \leq A_i \leq 10^9$$

Sample Input	Sample Output
5 1 2 -4 -2 3	6 3
10 5 6 7 10 -10 -5 11 -77 0 -2	39 6
9 -20 -3 -7 -2 -5 -3 -22 -22 -10	-2 1
11 -20 0 -3 -7 -2 -5 -3 0 -22 -22 -10	0 2

[Note: The maximum time-complexity for your program can be **O(n)**.]

## Task 2

Given a series of  $N$  positive integers  $a_1, a_2, a_3, \dots, a_n$  find the minimum and maximum values that can be calculated by summing exactly  $N-1$  of  $N$  integers. The print the respective minimum and maximum values as a single line of two space-separated integers.

### Input:

First line take input value of  $N$ .

Second line take input  $N$  space separated integer values.

### Output:

Two space separated values (one maximum sum and one minimum sum).

Sample Input	Sample Output
5 1 2 5 4 3	10 14
10 1 7 4 0 9 4 8 8 2 4	38 47
100 41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26 71 38 69 12 67 99 35 94 3 11 22 33 73 64 41 11 53 68 47 44 62 57 37 59 23 41 29 78 16 35 90 42 88 6 40 42 64 48 46 5 90 29 70 50 6 1 93 48 29 23 84 54 56 40 66 76 31 8 44 39 26 23 37 38 18 82 29 41  (if you can't copy this test-case, use the txt file)	4549 4648

[Note: The maximum time-complexity for your program can be  $O(n)$ .]

### Task 3

Tom wants to decorate his house by flower pots. He plans to buy exactly  $N$  ones. He can only buy them from Jerry's shop. There are only two kinds of flower pots available in that shop. The shop is very strange. If you buy  $X$  flower pots of kind 1 then you must pay  $A \times X^2$  and  $B \times Y^2$  if you buy  $Y$  flower pots of kind 2. Please help Tom to buy exactly  $N$  flower pots that minimizes money he pays.

#### Input:

The first line contains an integer  $T$  denoting the number of test cases.

Each test case is described in a single line containing three space-separated integers  $N, A, B$ .

#### Output:

For each test case, print a single line containing the answer.(Minimum money that he needs to pay, value of  $X$  and  $Y$ )

Sample Input	Sample Output
2 5 1 2 10 2 4	17 3 2 134 7 3
6 81 1 19 1 1 24 2 1 99 56 1 2 1 100 17 96 93 9	6233 77 4 1 1 0 4 2 0 2091 37 19 17 0 1 75647 8 88

[Note: The maximum time-complexity for your program can be  $O(n)$ .].

#### Task 4

Alexander wants to fight for Coding Club. In each round there are  $N$  soldiers with various powers. There will be  $Q$  rounds to fight and in each round his power will be varied. With power  $M$ , he can kill all the soldiers whose power is less than or equal to  $M$  ( $\leq M$ ). After each round, All the soldiers who are dead in previous round will be reborn. So in each round there will be  $N$  soldiers to fight. As he is weak in mathematics, help him to count the number of soldiers that he can kill in each round.

#### Input:

First line of input corresponds to the number of soldiers( $N$ ).

Second line contains the powers of each soldier.

The next line is the number of rounds to be played( $Q$ ).

The further lines corresponds to Alexander's power in each round.

Note: The input sequence will be sorted.

#### Output

The number of soldiers that can be defeated by him and the sum of the power of the defeated soldiers.

Sample Input:	Sample Output:
7 1 2 3 4 5 6 7 3 3 10 2	3 6 7 28 2 3
10 5 12 13 17 25 35 41 42 43 55 8 2 39 13 22 73 29 0 35	0 0 6 107 3 30 4 47 10 288 5 72 0 0 6 107

[Note: The maximum time-complexity to find the number of soldiers that can be beaten is  $O(\log(n))$ .]

### **Task 5:**

Implement **Insertion sort** and **Bubble sort** algorithm following the pseudocodes shown below. Then test the performance of these two sorting algorithms for different sizes of inputs. Although both of them have the complexity of  $O(n^2)$ , but due to the constant factors, the time will vary as the input size grows. This is due to the different comparisons done by the two algorithms.

#### **Steps to follow:**

1. Define the size of the input-array of your program. Test the program for different input sizes..
2. Generate a random set of numbers using **srand()** and **rand()** function in **C/C++**. Ensure that you have used the same set of numbers for both algorithms.
3. Implement both of the algorithms.
4. Check the correctness of your program by giving a small input set first.
5. Calculate the number of comparisons required by both algorithms.
6. Calculate the running/execution time of both algorithms. You can use **clock()** function to get the current **clock-tick** in your machine. To convert the clock-ticks into seconds you have to use **CLOCKS\_PER\_SEC** (which is a constant).

$$\text{Seconds} = \text{total clock-ticks} / \text{CLOCKS\_PER\_SEC}$$

7. A sample execution of the program is shown below:

```
Give size of Input: 100000

Generating random input set . . . Done.

Applying Insertion sort . . .
Total comparisons: 2500230376
Total clock ticks: 6068 (6.068000 seconds)

Applying Bubble sort . . .
Total comparisons: 4999950000
Total clock ticks: 24822 (24.882000 seconds)
```

8. Fill up the Evaluation sheet for different size of input.

#### **Note:**

- Use appropriate data type. (preferably long long int)
- Declare the array as Global.
- Check the resource folder if you are stuck !

### INSERTION-SORT( $A$ )

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3      // Insert  $A[j]$  into the sorted sequence  $A[1 \dots j - 1]$ .
4       $i = j - 1$ 
5      while  $i > 0$  and  $A[i] > key$ 
6           $A[i + 1] = A[i]$ 
7           $i = i - 1$ 
8       $A[i + 1] = key$ 
```

Figure 1: Insertion sort Algorithm

### BUBBLESORT( $A$ )

```
1  for  $i = 1$  to  $A.length - 1$ 
2      for  $j = A.length$  downto  $i + 1$ 
3          if  $A[j] < A[j - 1]$ 
4              exchange  $A[j]$  with  $A[j - 1]$ 
```

Figure 2: Bubblesort Algorithm

### Result Analysis

Size of Input-set (n)	Insertion Sort		Bubble Sort	
	Comparisons	Time(sec)	Comparisons	Time(seconds)
100	2451	0	4950	0
1000	244591	0.004	499500	0.008
5000	6305784	0.056	12497500	0.148
10000	25146692	0.2	49995000	0.664
50000	626340562	5.253	1249975000	21.555
100000				
500000				
1000000				