Data Structure

Tutorial 5: U19CS012

Question 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:

No of Elements: N

Set of N elements: A[i]

Output:

Sum

Set of elements involved in the sum

Constraints:

$1 \leq N \leq 10^5$

$-10^9 \leq Ai \leq 10^9$

SAMPLE INPUT

```
5
1 2 -4 -2 3
```

SAMPLE OUTPUT

```
6     3
```

Explanation

The chosen subset is {1, 2, 3}.

1.) Algorithm & Dry Run

(I) ALGORITHM for Question 1

Step 1 : Declare (start) global arr of 1e5 size

Declare n → size of array

sum → max sum of (+ve / 0) & initialize 0

ele_cnt → Element count & initialize 0

min → -(1e9 + 1)

// for finding Highest -ve number in arr [Edge case]

Step 2 : FOR i = (0) to (n - 1)

Read arr[i]

IF (arr[i] >= 0)

Add it to sum

Increase element count

ELSE

IF (arr[i] > min)

min = arr[i]

// For Highest -ve number in Array

Step 3 : IF (ele_cnt != 0) // 0 & +ve number exist

print sum , ele_cnt

ELSE

print min, 1 // If all elements are negative

I will chose Highest -ve number for max sum

(II) Dry Run for sample Input

5

1	2	-4	-2	3

	sum (0)	ele cnt (0)	min
1 >= 0	1	1	-(1e9+1)
2 >= 0	3	2	-(1e9+1)
-4 ✗ 0	3	2	-4
-2 ✗ 0	3	2	-2
3 >= 0	6	3	-2

∵ (3) ele_cnt != 0 → Ans is 6 3

2.) Code

```c
#include <stdio.h>

long long int arr[100001];
// At Max = 1e5 Elements
typedef long long int ll;
// Since -1e9<=A[i]<=1e9

int main()
{
    int n; // Number of Elements
    scanf("%d", &n);

    ll sum = 0;       // Max Sum
    int ele_cnt = 0; // Elements in Sub-Set
    ll min = -1000000001;
    // for Finding Highest -ve Number [Edge Case]

    for (int i = 0; i < n; i++)
    {
        scanf("%lli", &arr[i]);
        if (arr[i] >= 0)
        {
            ele_cnt++;
            sum += arr[i];
        }
        else
        {
            if (arr[i] > min)
                min = arr[i];
            // To Handle Edge Case When All Numbers are Negative
        }
    }
    // If All Elements are Negative, then ele_cnt = 0
    if (ele_cnt != 0)
        printf("%lli %d", sum, ele_cnt);
    else
        printf("%lli %d", min, 1);
    return 0;
}
```

3.) Test Cases

A.) Sample Test Pass [Positive + Negative]

```
5
1 2 -4 -2 3
6 3
```

B.) Postive + Zero + Negative

```
10
1 2 3 4 5 -1 -2 -3 -4 0
15 6
```

C.) Zero + Negative

```
5
0 0 0 0 -1
0 4
```

D.) All Negative Numbers

```
6
-10293 -122 -192 -65 -2 -9929313
-2 1
```

Question 2:

Given: Two array of integers **A, B, in Decreasing Order**. Create a Mad Array M, where

Madness of two numbers = **M (A[i], B[j]) = j - i**, if j >=i and B[j] >= A[i],

= 0 otherwise.

Final Madness Mad **(A, B) = max (M (A[i], B[j])) for 0≤ i, j< n-1.**

Input Format:
Input size of an array (Both array are of same size)

Input both array (values in Descending order)

Output format:
Print the madness of the two arrays.

Constraints:
1 <= Test Cases <= 50
1 <= N <= 10^5
1 <= Ai, Bi <= 10^{12}

SAMPLE INPUT

1
9
7 7 3 3 3 2 2 2 1
8 8 7 7 5 5 4 3 2

Sample Output

5

Sample Input

1
6
6 5 4 4 4 4
2 2 2 2 2 2

SAMPLE OUTPUT

0

Explanation

In the first case, we can see that 3 in the second array is the number which is equal to the 3 in the first array, and the difference between their positions is 5. So, the answer is 5.

1.) Algorithm & Dry Run

Algorithm for Question 2

Start

Step 1 : Declare two global arr 'a' & 'b' of size (1e5)

Step 2 : Declare test (int)
Read test
WHILE (test--) // Loop runs for each TEST
Declare n
Declare madness & initialize to 0
Read n, a[0], a[1], ... a[n-1]
b[0], b[1], .. b[n-1]

```
//LOGIC
FOR i = 0 to n-1
    FOR j = i to n-1

        IF ( b[j] >= a[i] )
            madness = max (madness, j-1)
        ELSE
            break the loop
            //since a & b are non decreasing sequence

print madness
```

(II)

Dry Run for Sample test

1
9

7	7	3	3	3	2	2	2	1
8	8	7	7	5	5	4	3	2
0	1	2	3	4	5	6	7	8

i	j	madness	j break at
0	(0 to 8) but ~~j > 3~~		(j=4)
1	(1 to 8)	max(3,2)=3	(j=4)
2	(2 to 8)	max(7-2, 3) = 5	(j=8)
3	(3 to 8)	max(7-3, 5) = 5	(j=8)
4	(4 to 8)	max(7-4, 5) = 5	(j=8)
5	(5 to 8)	max(8-5, 5) = 5	"
6	(6 to 8)	max(8-6, 5) = 5	"
7	(7 to 8)	max(8-7, 5) = 5	"
8		max ⇒ (5)	

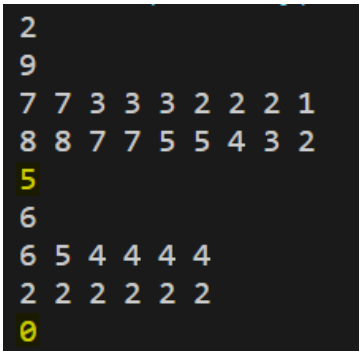
Final Answer ⇒ (5)

2.) Code

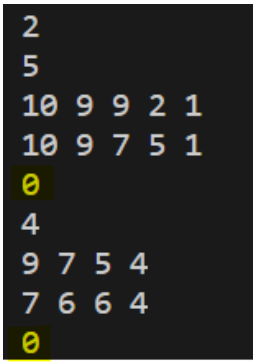
```c
#include <stdio.h>

long long int a[100001], b[100001];
// At Max = 1e5 Elements
#define max(a, b) (a < b ? b : a)

int main()
{
    int test; // Number of Test Cases
    scanf("%d", &test);
    while (test--)
    {
        int n; // Size of Both Array [A & B]
        int madness = 0;

        scanf("%d", &n);

        for (int i = 0; i < n; i++)
        {
            scanf("%lli", &a[i]);
        }
        for (int i = 0; i < n; i++)
        {
            scanf("%lli", &b[i]);
        }

        for (int i = 0; i < n; i++)
        {
            for (int j = i; j < n; j++)
            {
                if (b[j] >= a[i])
                {
                    madness = max(j - i, madness);
                }
                else
                {
                    break;
                    // Since it is Non-Decreasing Sequence
                    // we wont find any b[j] >= a[i] further
                }
            }
        }
        printf("%d\n", madness);
    }
    return 0;
}
```

3.) Test Cases

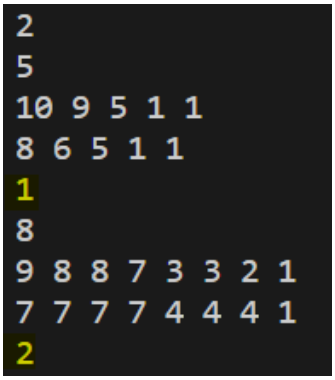
A.) Sample Test



B.) Another Test [There is no j that follows the Condition]



C.) Another Test [i=2 & j=2 in Test 1 and i=4 & j=6 in Test 2]



Submitted By:
Roll Number: **U19CS012** (*D-12*)
Name: *Bhagya Rana*