ACM SUMMER CHALLENGE

Editorial – Sorting

Victor In Georgopol

Problem prepared by: JIGNESH

Fun Fact : this problem was prepared before starting of contest but still problem was waiting for its turn till sorting

This problem is basic implementation of “Insertion sort”. You have to count number of swaps in Insertion sort.

If count of swaps is greater than **K**, then print “NO” else print “YES”.

Goku and discount

Problem prepared by: SHIVANGI

Fun Fact : all testers got WA in their first attempt

First idea that seems to correct is :-

sort the array and if sum of previous element ≤ current element, then count++.

But, Consider example **1, 4, 8, 5, 2**

If we select following order **1, 2, 4, 5, 8** then not disappointed customers = 3

and if we select **1, 2, 4, 8, 5** then we have not disappointed customers=4.

SOLUTION : sort the array and maintain sum of previously not disappointed customers say **SUM**.

If **SUM** ≤ A[i] , then customer A[i] is not disappointed. So **SUM += A[i];**

else customer A[i] is disappointed. So **SUM** += **A[i]**

Dr. Programmer

Problem prepared by: JITENDRA

Fun Fact : some testers saw the author's solution before testing their own. Because they also became frustrated with TLE. In the challenge, many students claimed that problem is not well prepared instead of reducing the complexity of their own solution.

Problem statement can be reduced to –

Find number of ways such that **X[i] + Y[j] ≤ Z**

All three types of query reduce our possible choices for **X[i]** or **Y[i]** or **Both.**

Hint 1: Sort the array

Hint 2: Consider query of **type A.** We have to find indices **LL1, RR1** such that

for all **index ≥ LL1 ,** X[index] ≥ L1

and for all **index ≤ RR1 ,** X[index] ≤ R1.

You can easily find **LL1, RR1** simply by binary search in time O(log N).

Also you can find **LL1, RR1** by a single loop in O(N) which is also accepted.

Similar approach can be done for query of ***type B*** and t***ype C***

Hint 3: Run a loop for all X-coordinates.

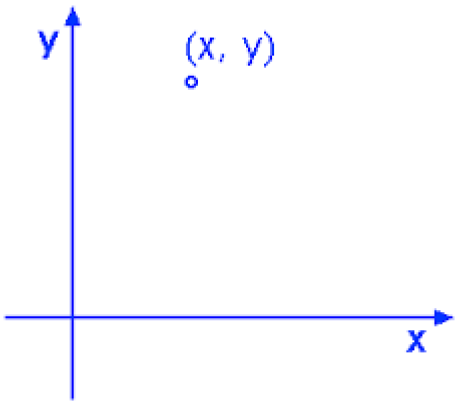
For all X[i] , all Y-coordinates ≤ Z – X[i] is valid.

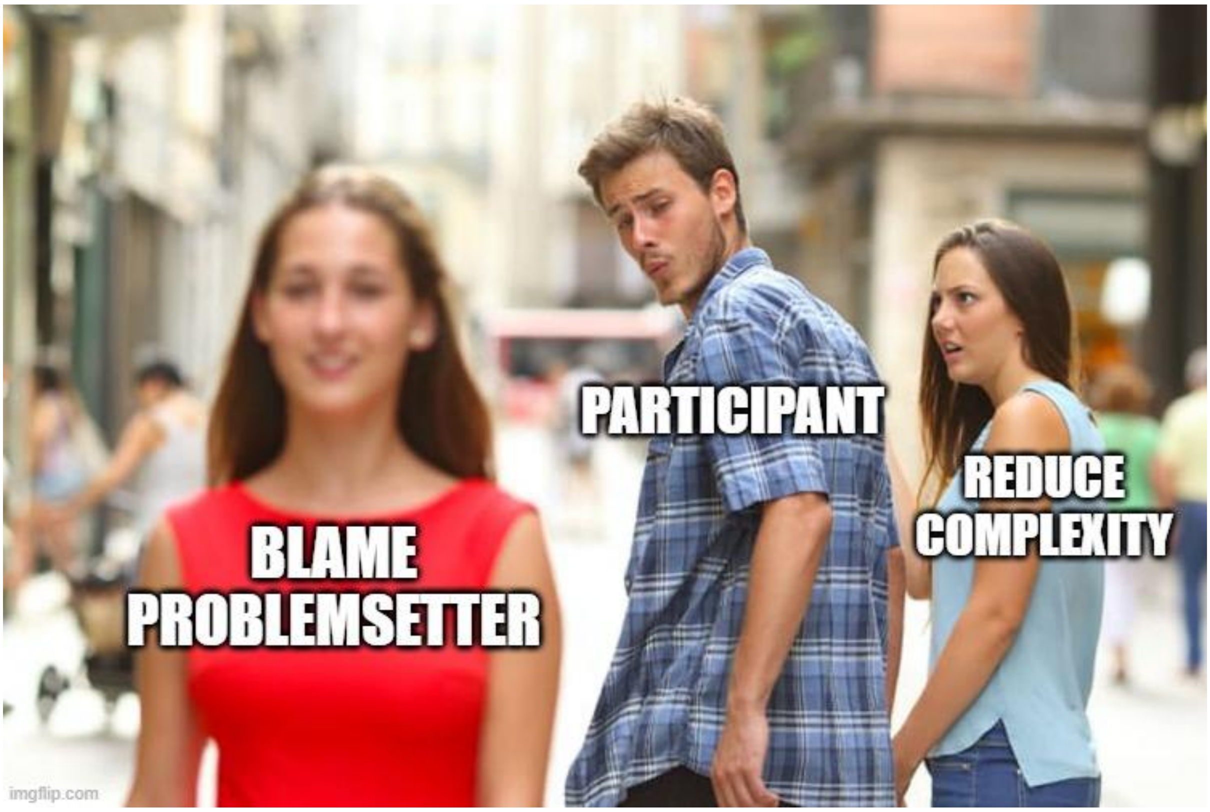
You can binary search for such valid Y-coordinates.

complexity of this method – O(N * Log N) , Which is not sufficient to pass all testcases.

Hint 4: instead of Hint3 use "**two pointer**" to reduce complexity to O(N).

Just maintain two variable **index, right.** Initially "index=1 & right=m" and iterate **index** in X-coordinates from left to right and **right** in Y-coordinates from right to left.(Assuming both array is sorted).


y
(x, y)
x


BLAME PROBLEMSETTER
PARTICIPANT
REDUCE COMPLEXITY
imgflip.com