## Bank Accounts

You are going to receive  payments in the upcoming month. The payments are numbered from  to and  denotes the amount of money in dollars that will be paid in connection with the -th payment. Before receiving the payments, you have two banking options to consider:

1. You can receive all the payments on your current bank account remembering that the bank charges you for each payment  , where  and  are given.
2. You can pay the bank  dollars upfront to open a new special account for which the bank doesn't charge you for any of the upcoming transactions.

Your task is to decide which of the above two options is more profitable to you. If both ways are equally profitable, then you prefer to be charged for each transaction.

**Input Format**

In the first line, there is a single integer  denoting the number of scenarios to handle. After that, descriptions of all these scenarios follow. In the first line of a single scenario, there are  space-separated integers . In the second line, there are  space separated integers .

**Constraints**

**Output Format**

Print exactly  lines. In the -th of them, print the answer to the -th scenario. More specifically, print feeif it's more profitable to use the current account and let the bank charge for each transaction, or print upfront if it's better to pay the bank up front for opening the new special account.

**Sample Input 0**

3

3 20 10 60

100 200 300

3 20 15 120

200 250 300

1 1 10 100

1000

**Sample Output 0**

upfront

fee

fee

**Explanation 0**

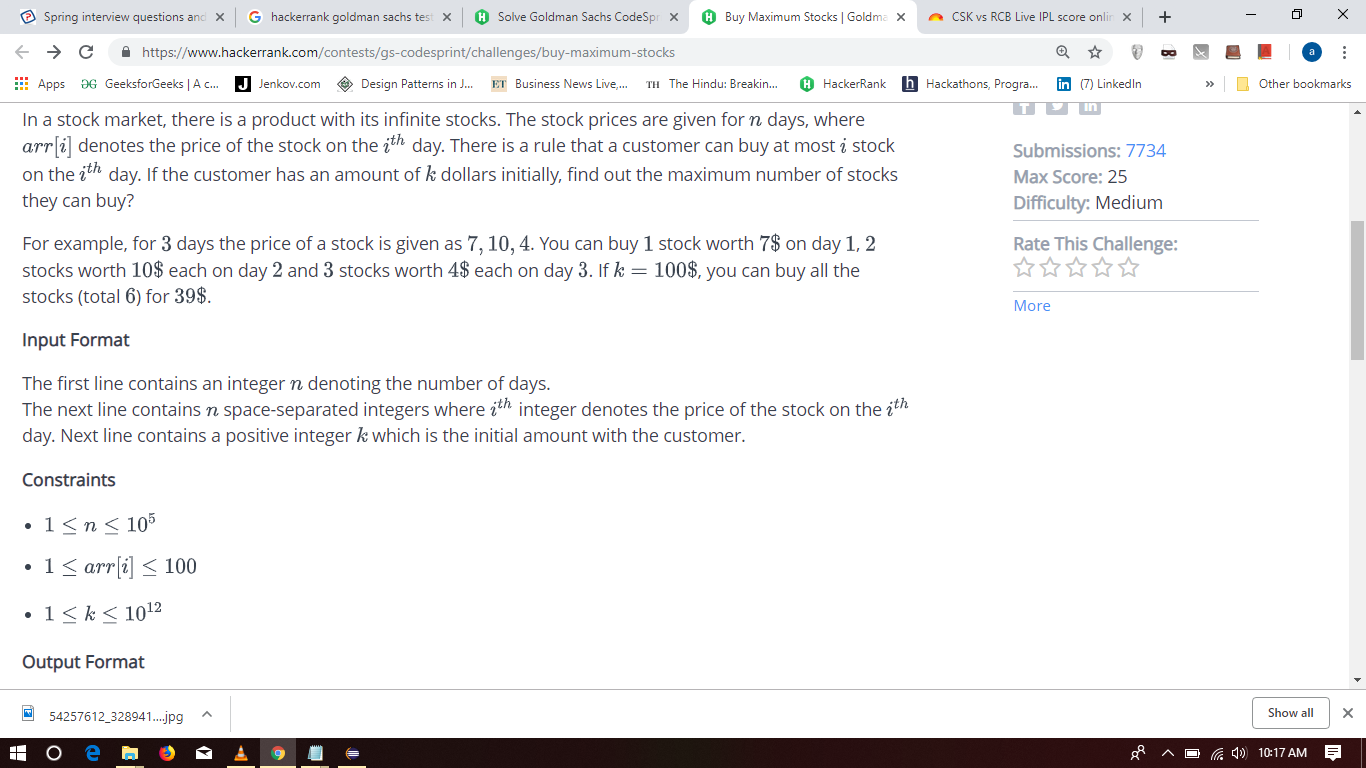
There are  scenarios in the sample.

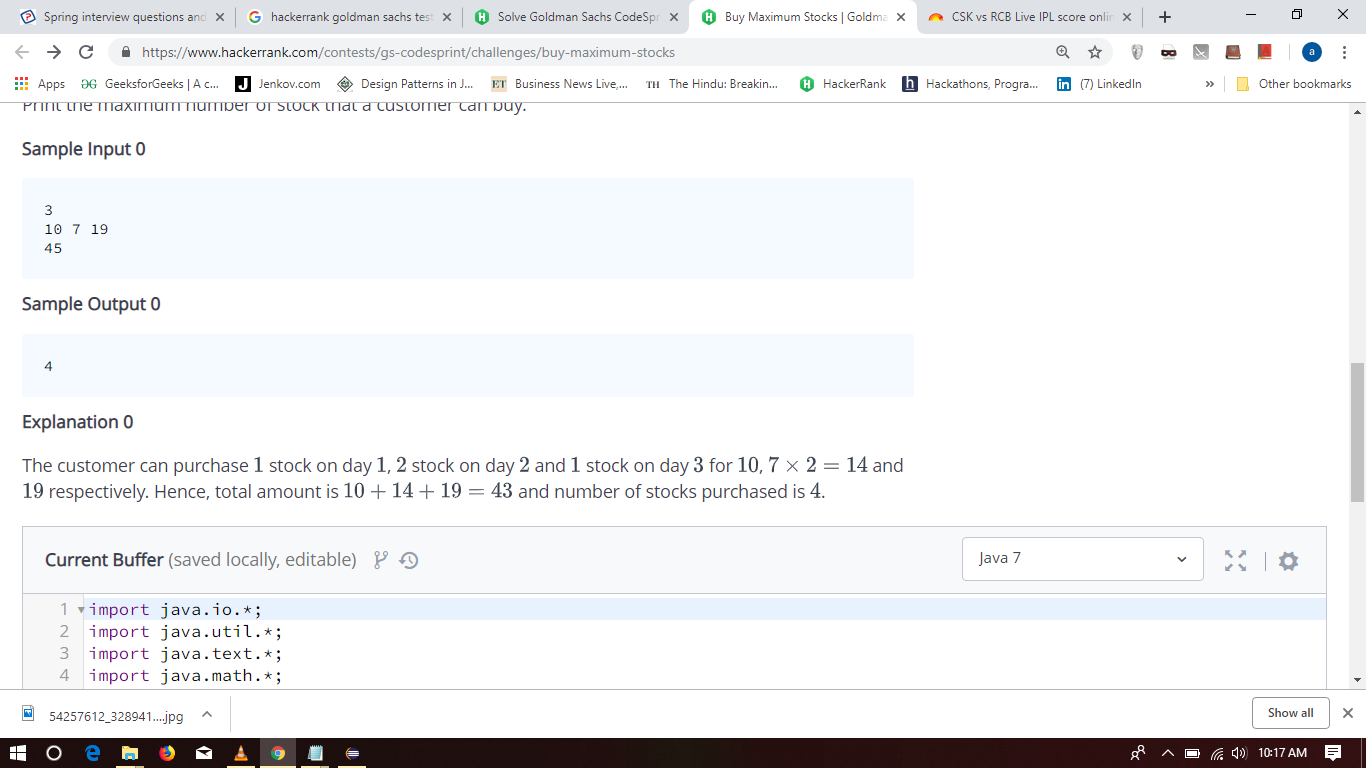
In the first one, there are  payments with amounts: . In the first option, for each transaction with amount , the bank charges . In the second option, you can pay upfront . It turns out that it's better to choose the second option, i.e. pay up front, because in the first option the bank would charge  across all the payments.

In the second scenario, it's better to be charged for each transaction because the bank will charge across all payments, which is strictly less than the cost of paying upfront.

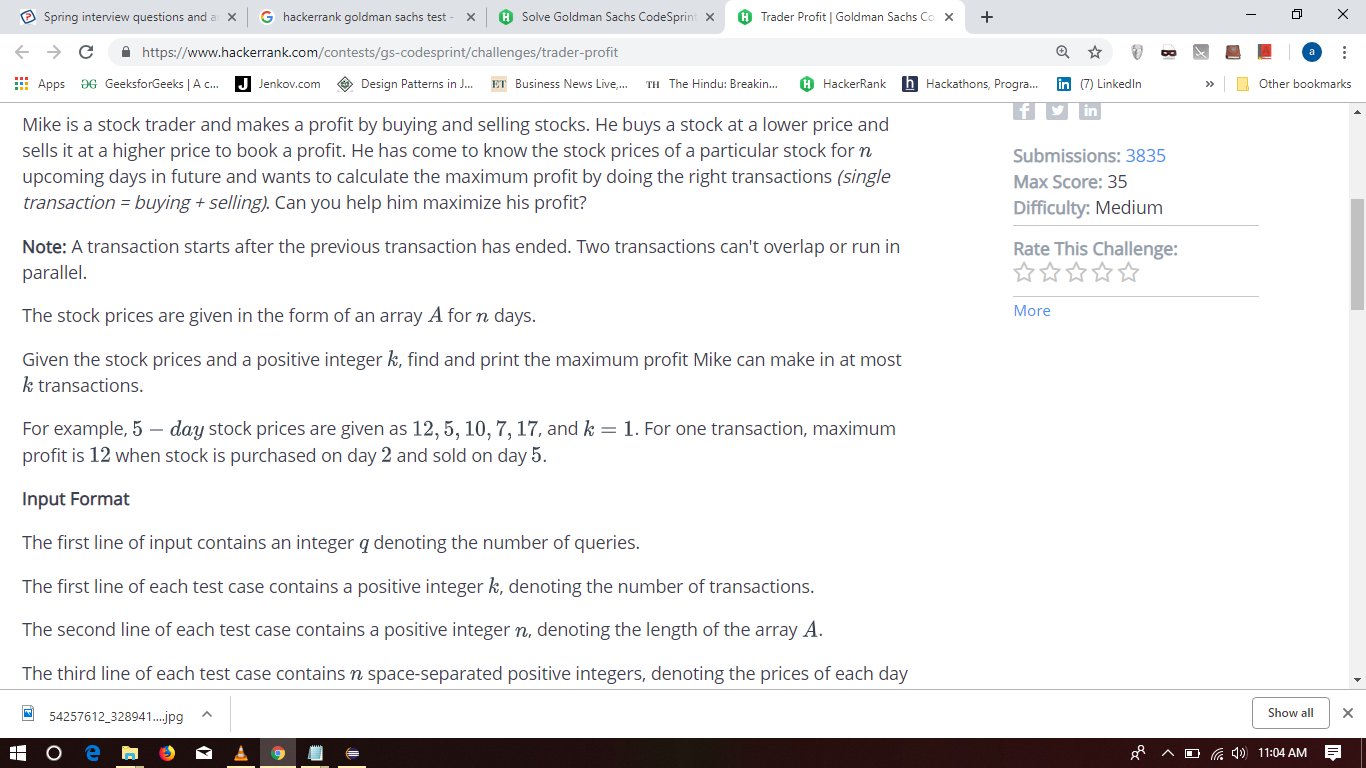
In the third scenario, each banking option costs . In the statement, it's written than in such a case we

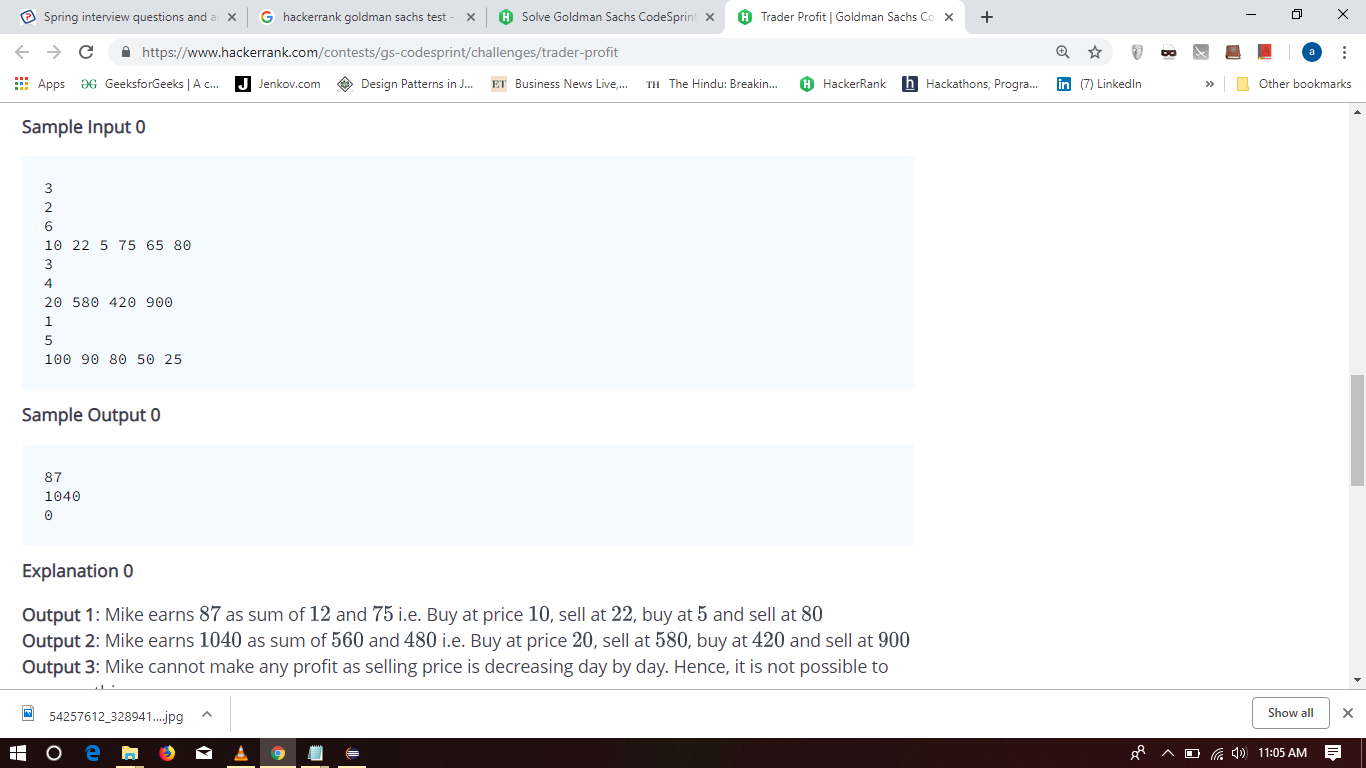
## Buy Maximum Stocks



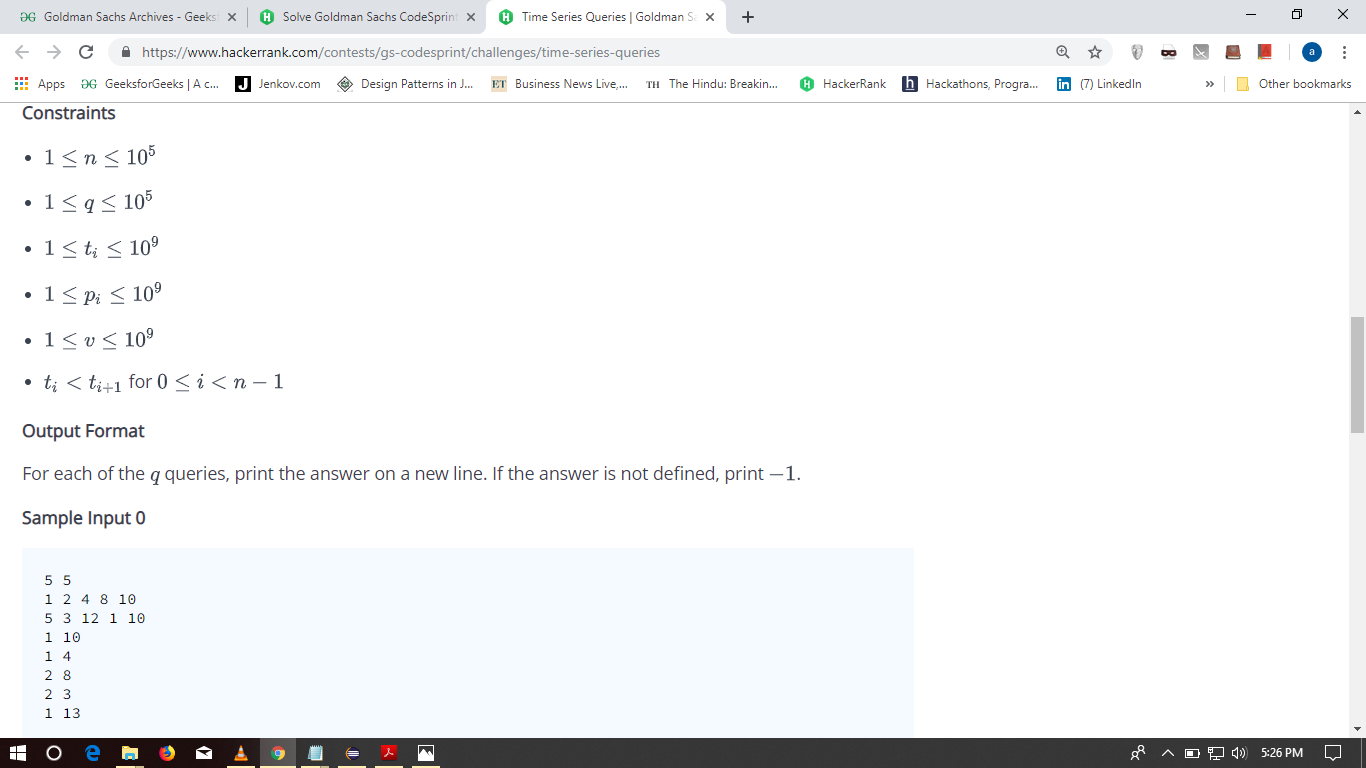
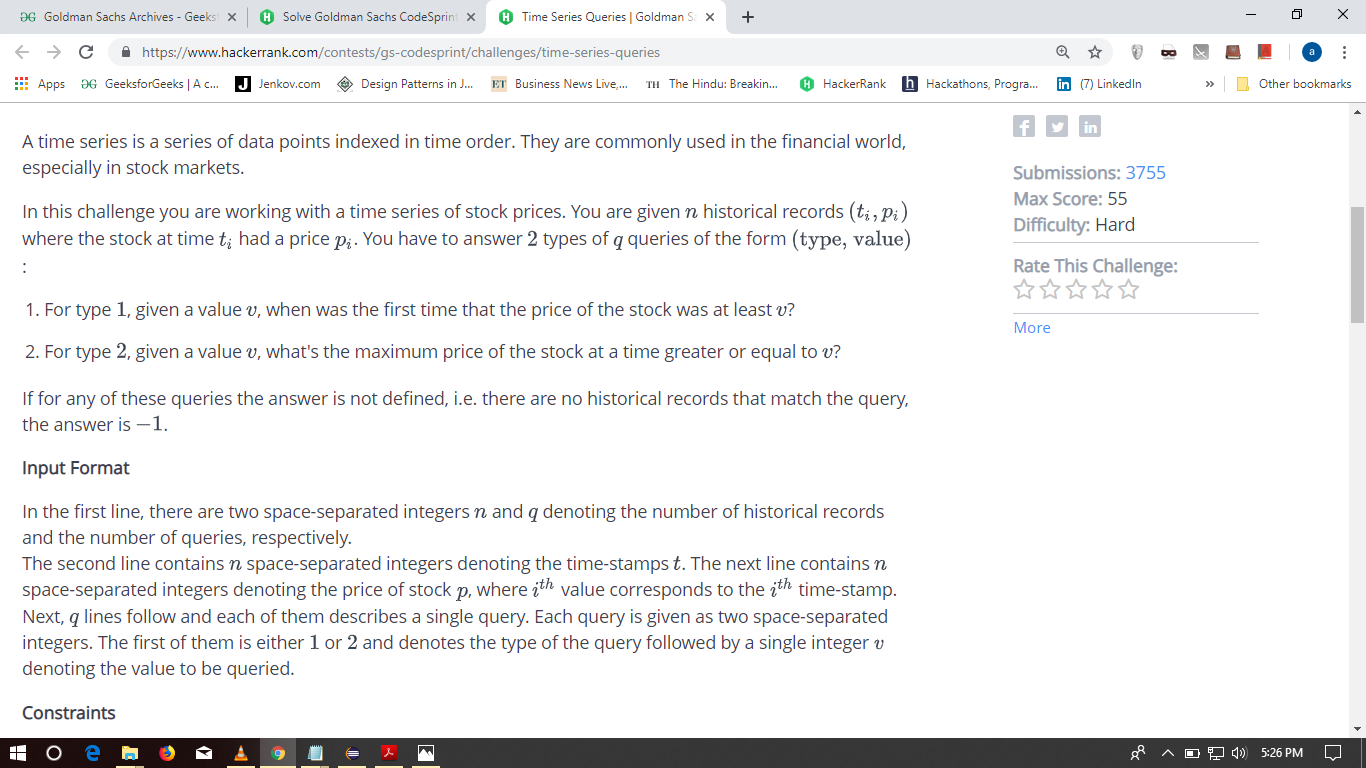


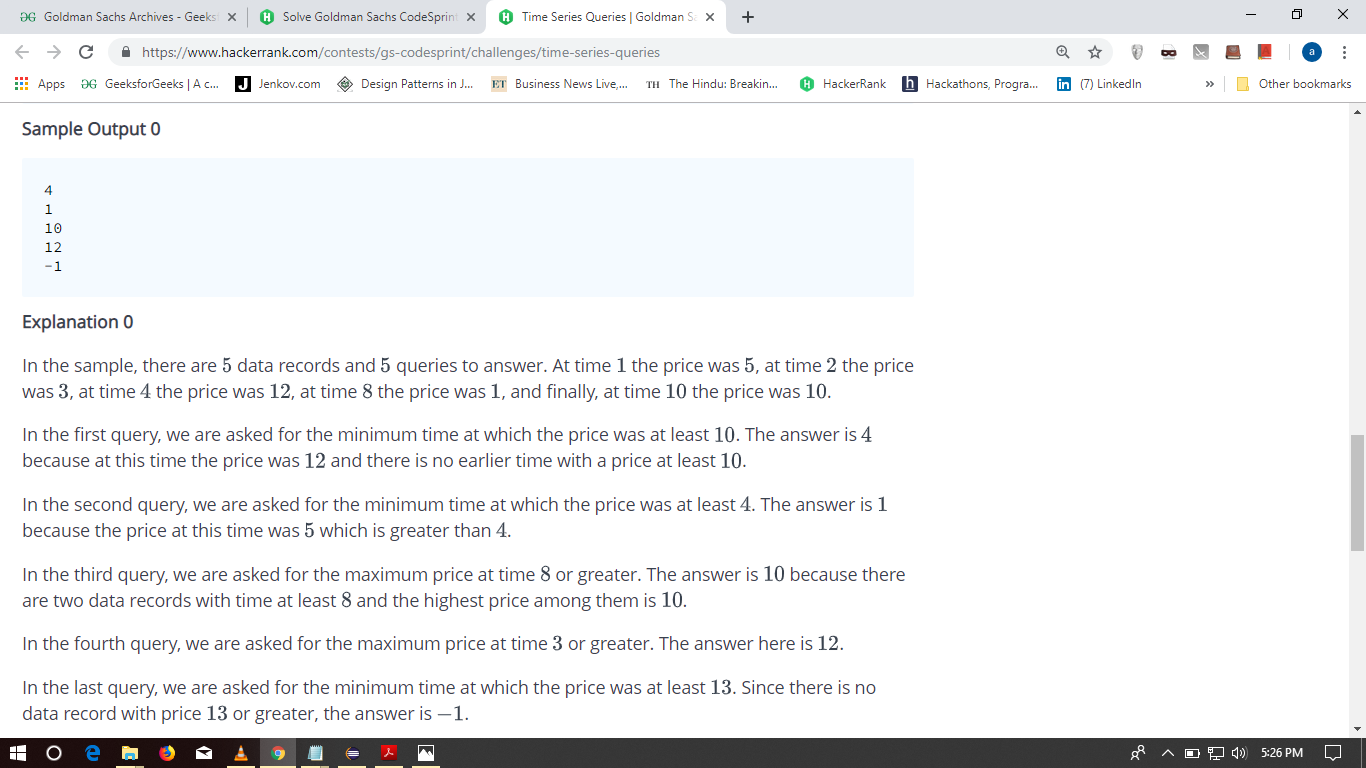
## Trader Profit

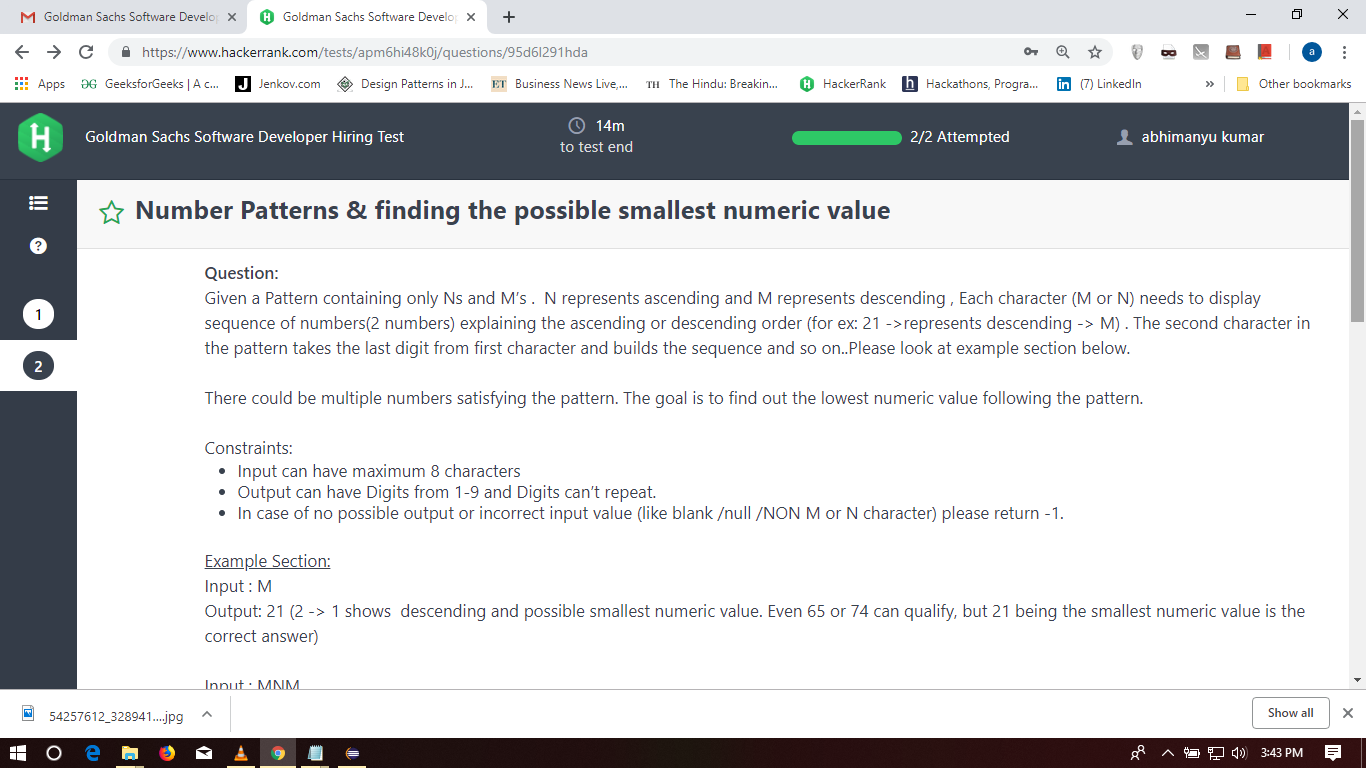


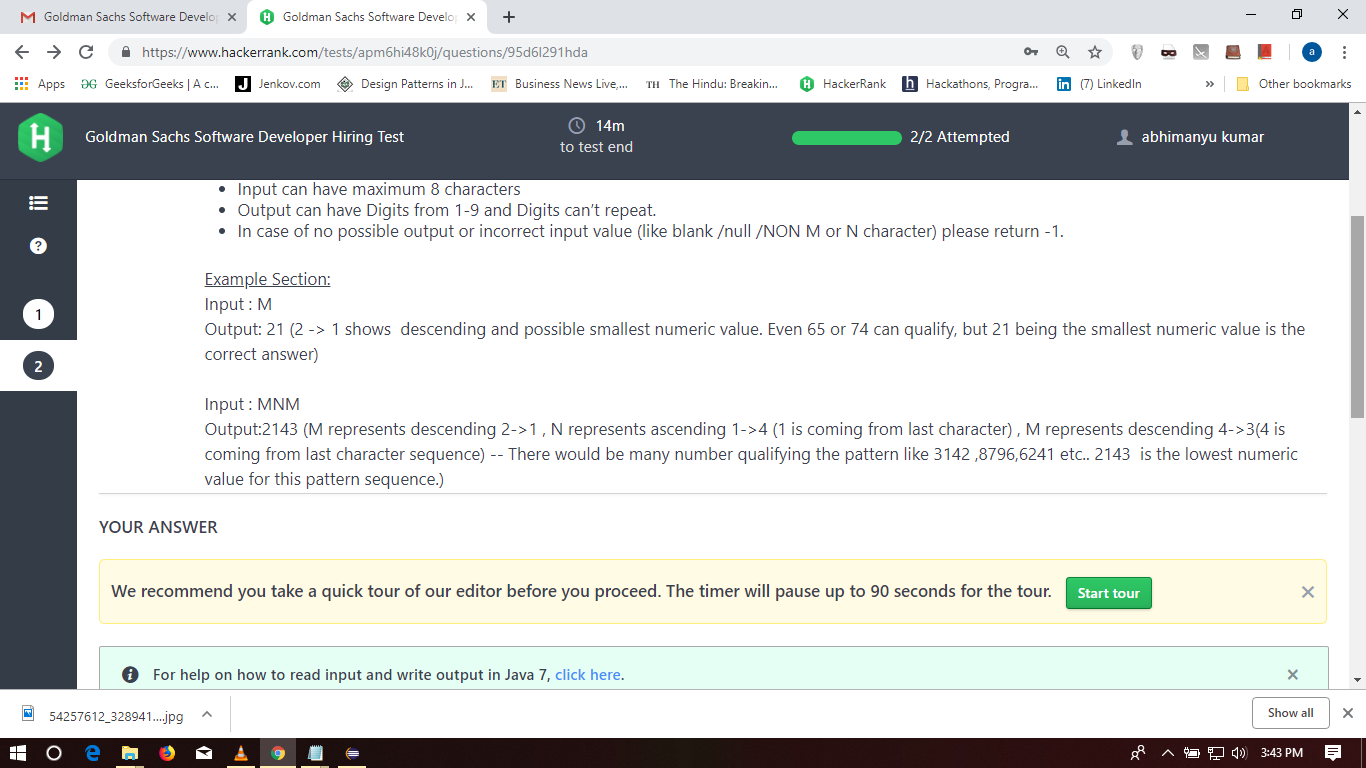


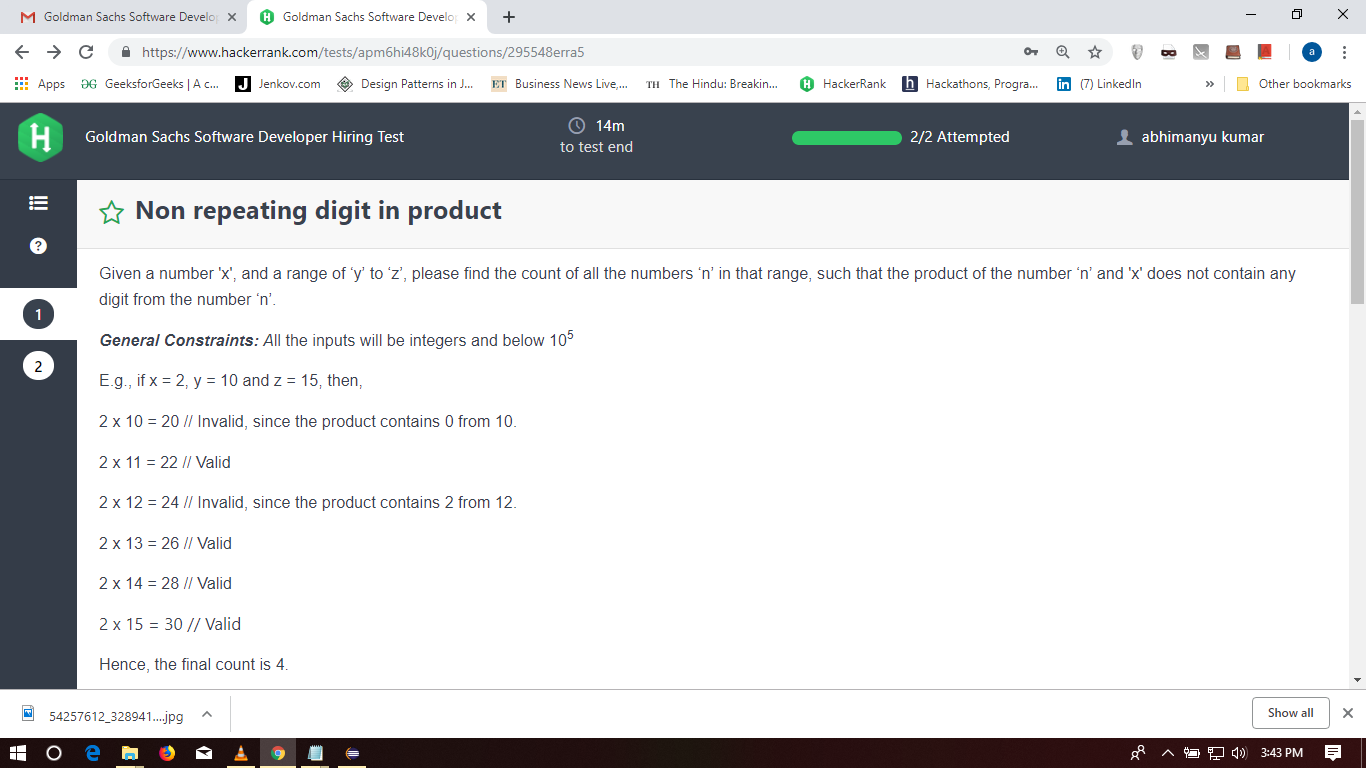
## Time Series Queries











Live coding

/\*

Instructions to candidate.

1) Your task is ultimately to implement a function that takes in an array of non-negative numbers and an integer.

You want to return the \*LENGTH\* of the shortest subarray whose sum is at least the integer,

and -1 if no such sum exists.

2) Run this code in the REPL to observe its behaviour. The

execution entry point is main().

3) Consider adding some additional tests in doTestsPass().

4) Implement subArrayExceedsSum() correctly.

5) If time permits, some possible follow-ups.

[3, 4, 2, 6] [ 3 , 6]

\*/

public class Solution

{

public static int subArrayExceedsSum(int arr[], int target )

{

return 0;

}

/\*\*

\* int doTestsPass()

\* Returns 1 if all tests pass. Otherwise returns 0.

\*/

public static void doTestsPass()

{

boolean result = true;

int[] arr = { 1, 2, 3, 4 };

result = result && subArrayExceedsSum( arr, 6 ) == 2;

result = result && subArrayExceedsSum( arr, 12 ) == -1;

if( result )

{

System.out.println("All tests pass\n");

}

else

{

System.out.println("There are test failures\n");

}

};

/\*\*

\* Execution entry point.

\*/

public static void main(String[] args)

{

doTestsPass();

}

};

1. Find average of two sorted array