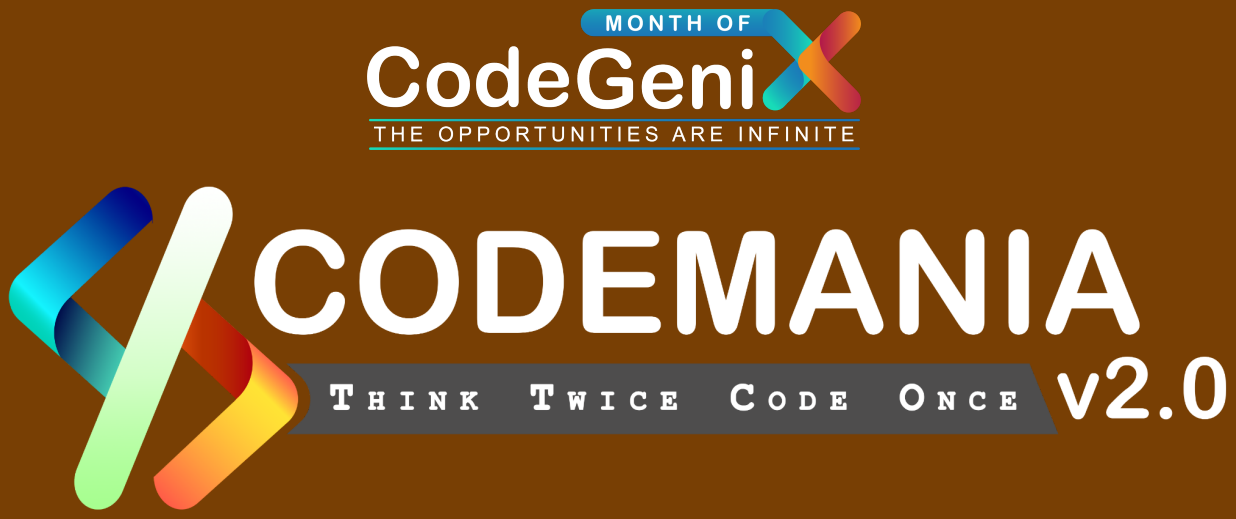


Sample Problems

1. Count Common Factor

This problem is asked in one of the HackerEarth contest.
Problem Statement: Little Robert likes mathematics. Today his teacher has given him two integers and asked to find out how many integers can divide both the numbers. Would you like to help him in completing his school assignment?
Input Formatting: There are two integers, a and b as input to the program.
Output Formatting: Print the number of common factors of a and b. Both the input values should be in a range of 1 to 10^12.
Example:
Input: 10 15

Explanation: The common factors of 10 and 15 are 1 and 5. So the answer will be 2.

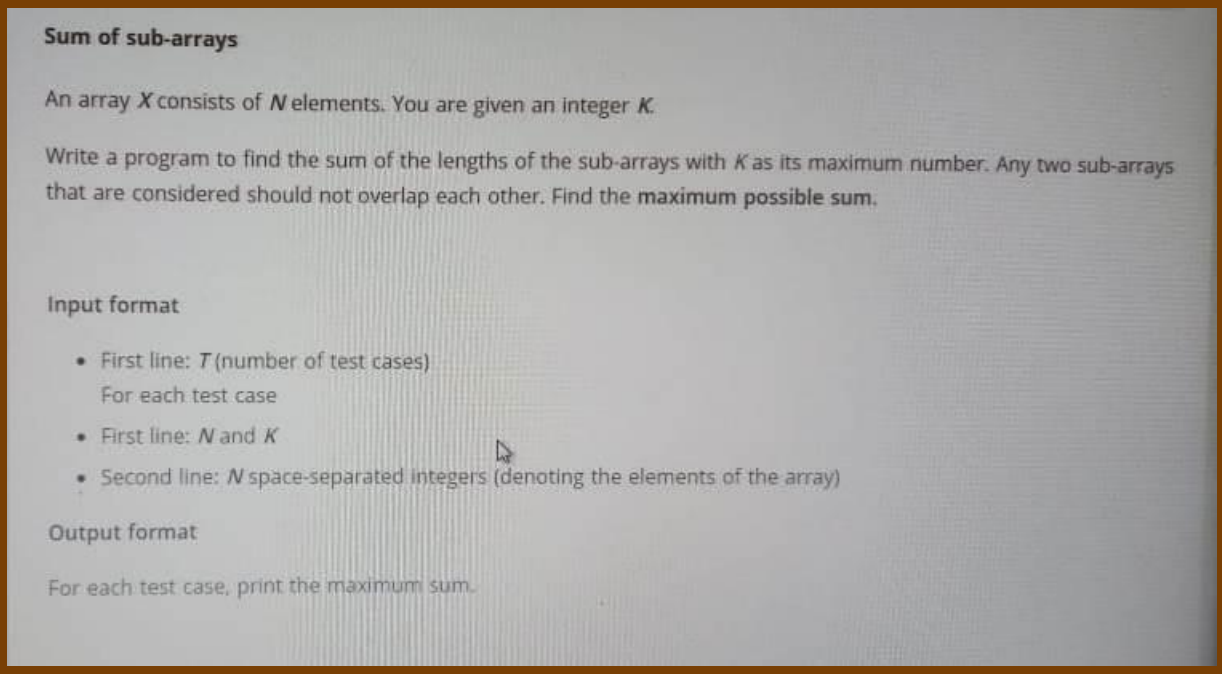
Python Code:

```
1   data  = input()
2   li = data.split()
3
4   a = int(li[0])
5   b = int(li[1])
6
7   def gcd(a, b):
8       if (a == 0):
9           return b;
10      return gcd(b%a, a);
11
12  if (a>0 and a<(10**12+1) and b>=1 and
13  b<(10**12+1)):
14      count = 1
15      for i in range(2, gcd(a, b)+1):
16          if a%i==0 and b%i==0:
17              count = count+1
18      print(count)
```

Explanation:

- **We are reading two integers as a single input and then splitting it using the split() method.**
- **The function gcd() is to find the greatest common divisor.**
- **To know how to find the GCD of two numbers, you can go through the function gcd() which uses recursion technique. It is self-explanatory.**

2). Sum of sub Arrays



Sum of sub-arrays

An array *X* consists of *N* elements. You are given an integer *K*.

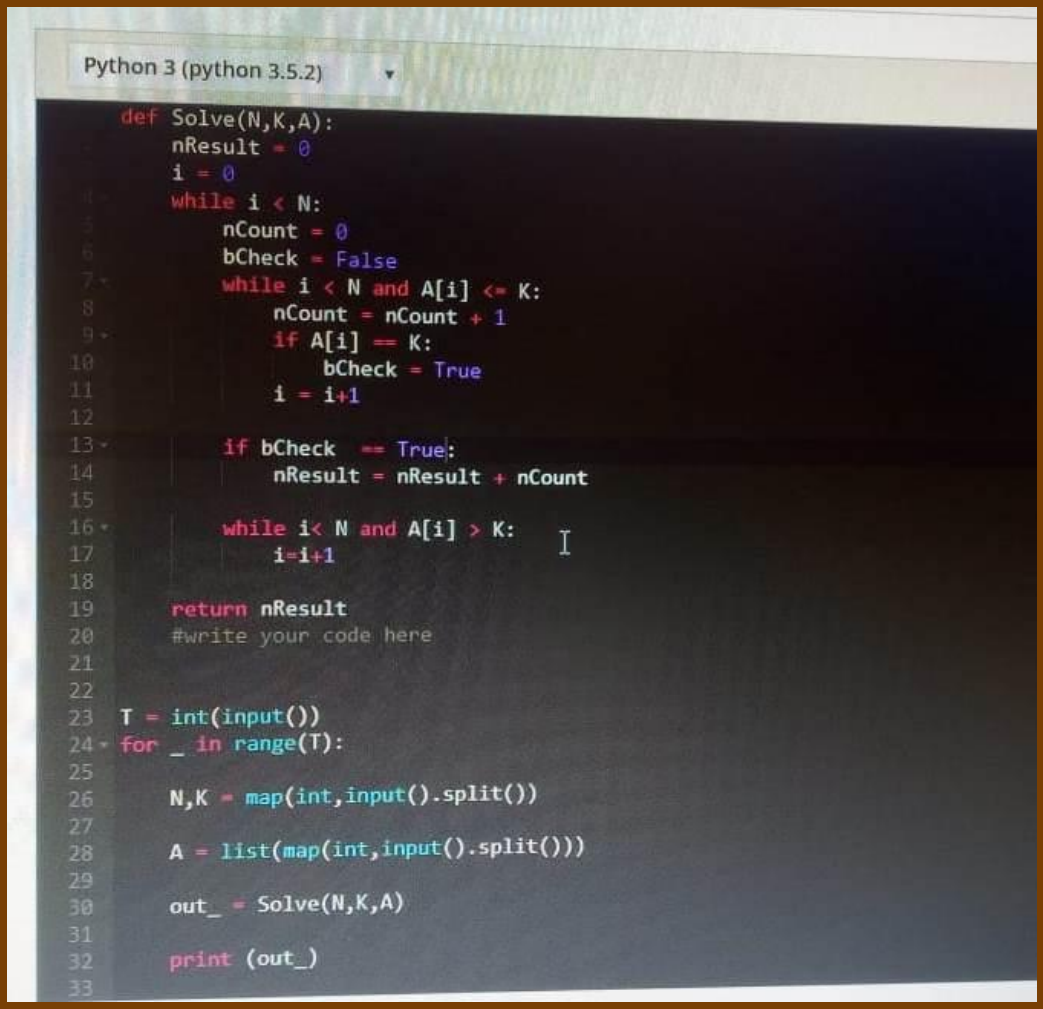
Write a program to find the sum of the lengths of the sub-arrays with *K* as its maximum number. Any two sub-arrays that are considered should not overlap each other. Find the **maximum possible sum**.

Input format

- First line: *T* (number of test cases)
  For each test case
- First line: *N* and *K*
- Second line: *N* space-separated integers (denoting the elements of the array)

Output format

For each test case, print the maximum sum.



3). **Problem Statement:**

You have given a string. You need to remove all the duplicates from the string.
The final output string should contain each character only once. The respective order of the characters inside the string should remain the same.
You can only traverse the string at once.

Example:
Input string: ababacd
Output string: abcd
You can use any programming languages like C/C++, Java, Python or anything you are comfortable with…

Remove all Duplicates from a given String in Python

In Python, we can use the set() method, to remove all the duplicate characters. But it will not preserver the ordering of the character.
print("".join(set("ababacd")))
Output:
acbd

Note: Every time you run this code, you will get a different output. The set operation returns characters in different orders.
Our output should be abcd.
Here is the solution.
Algorithm:

- Create the list of size 26. Initialize all the elements in the list to zero. (This is nothing but the array/list of 26 flags. The first element in the list will be for the character ‘a’, the second element in the list will be for character ‘b’ and so on…).
- Initialize output string as an empty string.
- Loop over all the characters in the given string from right to left.
    - Check the flag value for the given character.
    - If the flag is False (0), the given character is occurring the first time. Append that character to the output string. Set the flag to True (1).
    - If the flag is True (1), don’t do anything.

Python Code

```
1   msg = " ababacd"
2   li = [0] * 26
3   print(msg)
4   out= ""
5   for ch in msg:
6       ind = ord(ch) -
7   ord('a')
8       if li[ind] == 0:
9           out=out+ch
10          li[ind] = 1
11
    print(out)
```

Output:
abcd
Complexity:
You are traversing all the characters in the string only once, it takes linear time i.e. O(n) where ‘n’ is the length of the string.
It takes extra space to store the flag for all the 26 characters. It will be always the same size despite the length of the string. The space complexity will be O(1)
4).

Problem Statement / Task: You are given an array of integers (includes both positive and negative numbers). You have to find the sum of that subarray that has the highest sum among all the sub-arrays that can be formed including the array itself.
Example:
Input : -2 -1 -3 1 1 1 -4
Output : 3
Explanation: The subarray [1, 1, 1] has the highest sum (which is 3) in all the sub-arrays.
This coding question was asked in **Microsoft coding interview**.

4). MAXIMUM SUM SUBARRAY

Note: In this program, we are using Kaden's Algorithm you can find more **here**.

Python Program

Prerequisite:

- **Complete Python Tutorial**
- **Python map() function**

Code:

```
1  def sub_Array_Sum(lst):
2      max_now = 0
3      max_end = 0
4      for i in range(len(lst)):
5          max_now = max_now+lst[i]
6          if max_now < 0:
7              max_now = 0
8          elif max_end < max_now:
9              max_end = max_now
10     return max_end
11
12 print("Enter values for array:")
13 l = list(map(int, input().split()))
14 print("Maximum Sum Subarray: ",
   sub_Array_Sum(l))
```

Output:

Enter values for array:
-2 -1 -3 1 1 1 -4
Maximum Sum Subarray: 3

Output:

Max sum subarray: 3

Complexity

In interview, sometimes you will be asked to find the complexity of the your written code.
The complexity of the above program is O(n) as we are traversing each element in the list at once.

Similar Competitive Coding Challenges:

- **Smallest Subarray with Sum Greater Than Given Number**
- **Split Array into Equal Sum Subarrays**

This is all about the maximum sum subarray problem asked in the Microsoft interview. You can implement this logic in any other programming language of your choice.

5). **Problem statement:** You have given a circular rotated array. Write a program to sort the circular rotated array.

Example
Input (Circular Rotated Array): [3, 4, 5, 1, 2]

Output (Sorted Array): [1, 2, 3, 4, 5]
This question was asked in one of **Byju's coding interview**. Basically, they asked to find the given element in a circular rotated array in order of O(log(n) time.
To solve this problem, you have to rotate the array and then do the binary search.
Let's continue with the sorting circular rotated array.

Algorithm

- Find the index of the smallest element in the array (says index_smallest).
- Reverse all the left side elements of the smallest element in the array (excluding element at index_smallest)
- Reverse all the right side elements of the smallest element in the array (including element at index_smallest)
- Now, reverse the complete array.

Explanation

- Let's take an example of a circular rotated array [3, 4, 5, 1, 2].
- Smallest element is at position "3" (index_smallest = 3).
- Reverse all elements from index "0" to "2". The resultant array will be [5, 4, 3, 1, 2].
- Reverse all elements from index "3" to "4". The resultant array will be [5, 4, 3, 2, 1].
- Now reverse all the elements in the array. You will get a complete sorted array as [1, 2, 3, 4, 5].

You can solve this problem in any programming language of your choice.

Python Program

In Python, you can **reverse the list using extended slice syntax** (just one line code). But, in competitive programming, you will be encourage to write your own logic.
Code

```
# function to reverse the array list
def reverse_array(arr, start=0, end=0):
    if end == 0:
        end = len(arr)-1
    while start<=end:
        temp = arr[start]
        arr[start] = arr[end]
        arr[end] = temp
        start += 1
        end -= 1

# function to sort cicular roated array list
def sort_circular_rotated_array(arr):
    index_smallest = 0
    for i in range(len(arr)-1):
        if arr[i]>arr[i+1]:
            index_smallest = i+1
            break

    reverse_array(arr=arr,
end=index_smallest-1)
    reverse_array(arr=arr,
start=index_smallest)
    reverse_array(arr=arr)
    return arr


arr = [3, 4, 5, 1, 2]
print(sort_circular_rotated_array(arr))
```

Output

[1, 2, 3, 4, 5]

Complexity

It takes O(n) time to find the smallest element in the array. For reversing array it will take O(n) times in the worst-case scenario.

We are calling the reverse array function thrice but each element will get displaced twice so the overall complexity for reversing array is 2*O(n).

Total complexity is O(n)+2*O(3) which is equivalent to O(n). So, the overall time complexity of this algorithm is O(n).

We are not creating any new array and all the elements in the array are getting changed in place. So, the space complexity is O(1) i.e. constant space.

Hope you learned a new trick to sort the circular rotated array and to find the element in the circular rotated array in minimum time.

6). Problem Statement: Write a Python program to validate if the given string is a valid IP address or not.

This is one of the **questions asked in Juniper coding interview**.
In Computer Network, the IP address uniquely defines each host or node in the network. You can read mode about **IP address in Computer Networking**.

Basically, this is a pattern matching problem where we can use a regular expression (RegEx).

IP Address Validation using Python RegEx

Here we are using the search method from the re module. You can find more detail in the **Python RegEx tutorial**.

Code

```
1   import re
2
3   regexIP =
4   '''^(25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
5      25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
6      25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
7      25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)'''
8
9   def validateIP(strIP):
10
11     if(re.search(regexIP, strIP)):
12        print(f"{strIP} is a valid IP address.")
13
14     else:
15        print(f"{strIP} is an invalid IP address.")
16
17
18  if __name__ == '__main__' :
19
20     strIP = "122.134.3.123"
21     validateIP(strIP)
22
23     strIP = "44.235.34.1"
24     validateIP(strIP)
25
26     strIP = "277.13.23.17"
       validateIP(strIP)
```

Output

122.134.3.123 is a valid IP address.
44.235.34.1 is a valid IP address.
277.13.23.17 is an invalid IP address.

Note: When we say IP address, by default its IPv4 address which is different from IPv6 address.

This is a simple program to validate IP address using regex in Python.