Competitive Programming

1. Codechef (Basics & Problems Understanding)
2. LeetCode (few solved Problems)
3. C++ for Competitive Programming

(If you want to write Notes about your new learning in CP or GK then write in Drive in Google. Make Excel like thing. It will be accessible anytime)

Scanned with OKEN Scanner

Competitive Programming

Book - 101

Basic Codechef Understanding And Few Problems Understandings

→ Prime Seive

→ WA, Sgv error mean in code chef etc.

Scanned with OKEN Scanner

Page Cover Writing

(1) Sometimes you get NZEC error, even your code run fines in IntelliJ IDEA, then we use

```
try { Scan your code here}
Catch (Exception e) {
                return;
}
```

→ In your full code.

e.g. See in (#5) Extra Knowledge

---

(2) Sometimes sigsegv error in your code although your code is all correct.

⇒ Because you don't need to allocate array for for the same. See the constraints, It will go out of memory.

e.g. $0 \le n \le 10^{12}$

try to get ans with storing values in array and just finding the end ans. (Update array with each new ans in used index only rather than going to higher index.)

e.g. in fibonacci, you can get 15000th element by loop but you don't have to store all in the array, but just use single 'ans' integer to store continously until ans and return it.

(Don't use array upto 10^12 and you can't make an array that big. Simply use loops upto n and constantly operate over the values).

Scanned with OKEN Scanner

(3) If in some question given for since sum is large, compute it to modulo 1,000,000,007 (10^9+7)

↳ Then

long mod = 1000 000 007

sum = sum + arr[i] - i → Wrong

Correct → sum = ((sum % mod) + ((arr[i] - i) % mod)) % mod

In simple competitive programming I used int → 3 out of 10 case passed

Then when I used long → 10 out of 10 case passed

It had constraint (1 ≤ N ≤ 10000000 (1e7))

0 < Q < LLONG_MAX

So used long

INDEX

S.No.	Date	Topic	Page No.	Sign/ Remarks

EXTRA KNOWLEDGE

100 Using bufferedwriter bw.write(num+" ") gives AC whereas system.out.println gave TLE. So using bufferedwriter in case you've to print 2D Matrix is efficient

EFFICIENT CODING WITH EXAMPLE PROBLEMS.

Sorted

⇒ Sometimes when constraint is given as $1 \le N \le 1000$ then make array of size like 10000 or more. Such was also Not AC. After doing 1000 to 10000 AC we got

EFFICIENT CODING ALGO's MOST OFTEN USED.

(1) Dynamic Programming → Find out all possible solutions and then pick up the best or optimal one
↳ Coins sum type

(2) Greedy Method → Always goes for min value at each step and hence gives an optimal solution

Scanned with OKEN Scanner

⇒ "Wrong Answer" (WA). This means, that your solution doesn't compute the correct result for all test cases.

Codechef has a lot of hidden test cases. So even if your program works with 2-3 test cases from the problem statement, it still can be wrong and receive WA.

Sometimes these hidden test cases contain very special cases, that you've to handle differently. Sometimes these hidden test cases contain very large instances, e.g. arrays of length 10^5, sometimes just normal small testscases.

ither way if you receive WA your program mputes (sometimes) wrong answers. Check it again,

⇒ However in long challenges Partial marks is provided, for given constraints specified in the question.

⇒

5***

(#) In case you want to integer or keep on inputting until the certain integer value is found then Code as :-

```
P          S          V M () {
|
|   Scanner sc = new Scanner (System.in);
|   int n;
|
|   while ((n = sc.nextInt()) != 0) {
|   |
|   |       System.out.println(n);
|   }
|
}
```

⇒ Input :-

```
1
2
88
0
5
99
```

→ It will stop taking inputs

output →

```
1
2
88
```

(5) How to get rid of getting NZEC runtime error while submitting solution in Java?

NZEC → Non zero exit code → mostly during segmentation fault

reason) :-

① Infinite recursion — or basically when you run out of stack memory

② Incorrect Memory Access :-

⇒ Try using the format :-

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main (String[] args) throws IOException
    {
        try {
            //your code
        } catch (Exception e){
            return;
        }
    }
}
```

EXTRA KNOWLEDGE

#① Fast I/O in Java with single line inputting options :-

⇒ We know fastreader self defined can work everytime for single time but for it we've to write lengthy code and also it's not as efficient as BufferedReader so.

⇒ So now we'll see about inputting in single line via BufferedReader.

⇒ Before using any method first always try to mention "throws java.lang.exception" in the main function as :-

```
public static void main (String[] args) throws java.lang.exception{
}
```

⇒ While debugging your code don't put the break points on inputting lines. It won't work properly otherwise.

Scanned with OKEN Scanner

§ We could also use scanner to write inputs in single line using spaces.

⇒ Also sometimes it's better to use scanner when simply its use.

code as :-

```
public static void main (String[] args) throws java.lang.Exception {
    Scanner sc = new Scanner(System.in);
    int testCase = sc.nextInt();

    for(int t=0; t<testCase; t++){
        long a = sc.nextLong();
        long b = sc.nextLong();
        long c = sc.nextLong();

        //solving part:-
        long x = Math.abs(2*b-a-c);
        long y = x/2 + x%2;

        System.out(y);
    }

}
```

⇒ It's inputs :-

```
4
-5 0 5
0
-5 7 6
7
-10 -100 20
105
51 23 10
8
```

⇒ Also in this scanner we cannot put multiple strings in same line by space. (This statement below not valid any more ☺ bcoz)

⇒ But if there's a string in numbers form then you can take input in the form of int and then convert it into a string.

e.g. code → next page →

ok this riddle solved

You can take scanner input to input multiple strings in same line with space.

We just have to use [illegible] instead of

e.g. code :-

```
Scanner sc = new Scanner(System.in);
int testCase = sc.nextInt();

for(int i=0; i<testCase; i++){
    int i1 = sc.nextInt();
    int i2 = sc.nextInt();

    String str1 = Integer.parseInt(i1);   // toString
    String str2 = Integer.toString(i2);

    Sys o(Solution(str1, str2));
}
```

inputs :-

```
2
1 1 1 0   1 0 0 1
20
1 1 1 1   1 0 1
10
```

⇒ Scanner can also take Array of integer or character values also in single line with spaces.

Code as :- rem! to put throws.java.lang.Exception

Code as :-

```
Scanner sc = new Scanner(System.in);
int testCase = sc.nextInt();

for(int t=0; t<testCase; t++){
    int len = sc.nextInt();

    int[] arr = new int[len];

    for(int i=0; i<arr.length; i++){
        arr[i] = sc.nextInt();
    }

    Arrays.stream(arr).forEach(System.out::print);
}
```

inputs :-

```
2
5
1 2 3 4 5
1 2 3 4 5
8
9 8 7 6 5 4 3 2
9 8 7 6 5 4 3 2
```

Scanned with OKEN Scanner

⇒ Scanner can also be used to put multiple strings in single line using spaces by using "sc.next()".

code as :-

```
Scanner sc = new Scanner (System.in);{
int testCase = sc.nextInt();

for(int t=0 ; t<testCase ; t++) {
    String s1 = sc.next();
    String s2 = sc.next();

    Syso ( s1 + " && " + s2);
}
```

inputs

```
2
WhatIs YourName
WhatIs && YourName

MyName IsVicky
MyName && IsVicky
```

⇒ Scanner can also be used to put array of strings as :-

code as :-

Imp :- (Here also even if you write 3 string in upper line & rest 2 in lower line than also as a size 5 will take all five of them. Like in self defined fastReader)

```
Scanner sc = new Scanner (System.in);
int testCase = sc.nextInt();
for (int t=0; t<testCase; t++) {
    int len = sc.nextInt();
    String[] strings = new String[len];
    for (int i=0; i<strings.length; i++) {
        strings[i] = sc.next();
    }
    Arrays.stream(strings).forEach(System.out.print);
}
```

inputs

```
2
5
What Is Your Name
WhatIsYourName

5
My Name Is Vicky
Kumar

MyNameIsVickyKumar
```

← So this way also it works like in user defined fastReader.

Scanned with OKEN Scanner

⇒ So its all about Scanner.

Now come to BufferedReader :- ⇒

⇒

Here also always use
throws java.lang.Exception in its main function.

Types

① For inputting multiple integers or long or double in same line by spaces.

② Array of int, long, double in same line

③ Array of string in same line

④ multiple string in same line

multiple times → multiple integer or long and also array of integer in next line.

→ Still not found soln to inputting 2D array of integers.
*** for now just use scanner.

ⓗ BufferedReader is fastest among all.

⇒ rem! :- To write in single line with spaces we make use of assigning "br.readLine()" to string Lines as `String lines = br.readLine();` and then in this single line input we seem each element by spaces. so consecutive code as :- `String[] str = lines.trim().split("\\s+");`

⇒ so take care that after assiring 'br.readLine()' to 'String lines' we cannot further use 'br.readLine()', else it will show compile time error.

⇒ Also here we need to use in single line only all the n^{th} numbers of item, if you try adding by enter for that n times it will give you ArrayIndexOutOfBoundException. so it's not like that of Scanner or User Defined fastReader.

⇒ It is decided to single line all the n inputs of the array or whatever you are putting for 'str[0]' part after assigning to lines.

Scanned with OKEN Scanner

⇒ BufferedReader to input multiple integers, or long, double in single line as :-

de as :-

```
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
int testCase = Integer.parseInt (br.readLine());

for (int t=0; t< testCase; t++) {
        String lines = br.readLine();
        String [] str = lines.trim().split ("\\s+");

        int i1 = Integer.parseInt (str [0]);
        int i2 = Integer.parseInt (st[1]);

        Sys o (i1+i2);
}
```

input :-

```
2
1 4
5
4 5
9
```

Sir

If you put input as

```
2
1
4
```

It give ArrayOutOfBound Exception

So it's directed to single line then single line line.

⇒ BufferedReader to input Array of int, long, double

⇒ code as :-

```
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
int testCase = Integer.parseInt (br.readLine());

for (int t=0; t < testCase; t++) {
        int len = Integer.parseInt (br.readLine());
        int[] arr = new int[len];

        String lines = br.readLine();
        String[] str = lines.trim().split ("\\s+");

        for (int i=0; i< arr.length; i++){
            arr [i] = Integer.parseInt (str[i]);
        }

        Arrays.stream (arr).forEach (System.out::print);

        System.out.println ("");
}
```

Input :-

```
2
5
1 2 3 4 5
12345 <----
8
8 7 5 6 1 2 3
8756123
```

if not this then you're to start from here since Line println not used it's not too much to worry it's just way of printing.

Scanned with OKEN Scanner

⑤ ⇒ BufferedReader for input Array of Strings

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int testCase = Integer.parseInt(br.readLine());
for (int t=0; t<testCase; t++){
    int len = Integer.parseInt(br.readLine());
    String[] arr = new String[len];
    String lines = br.readLine();
    String[] str = lines.trim().split("\\s+");
    for (int i=0; i<arr.length; i++){
        arr[i] = str[i];
    }
    Arrays.stream(arr).forEach(System.out::print);
    System.out.println("");
}
```

inputs :-

```
2
5
What Is Your Name Plz
WhatIsYourNamePlz
3
Iti Vicky Kumar
ItiVickyKumar
```

⑥ BufferedReader for Inputting multiple strings in single line using space

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int testCase = Integer.parseInt(br.readLine());
for (int t=0; t<testCase; t++){
    String lines = br.readLine();
    String[] str = lines.trim().split("\\s+");
    String s1 = str[0];
    String s2 = str[1];
    System.out.println(s1 + "&&" + s2);
}
```

inputs :-

```
2
Hello There
Hello&&There
How AmI
How&&AmI
```

Scanned with OKEN Scanner

#② Efficient Algorithm to check whether a number is prime or not :-

Code :-

Time complexity $O(\sqrt{n})$

```
static boolean isPrime(int n){
    if(n==1) return false;
    if(n==2) return true;
    for(int i=2; i<=Math.sqrt(n); i++){
        if(n%i==0){
            return false;
        }
    }
    return true;
}
```

total factors e.g. Factors of 32

1 × 32
2 × 16
4 × 8

Here only we can identify all factors on going for 1 to $\sqrt{32}$

So from 2 to $\sqrt{n}$

If there is any factor found then it is not prime else prime.

To find whether a number is prime or not.

Logic is if a number has more than 2 factors then it is prime.

e.g. For 10 → 5
25 → 5
factor 1 → 1
factor 2 → 'n' (number itself)

So for any number it's last factor would be it's half i.e. $\frac{n}{2}$

⇒ Don't you ever use BigInteger's isProbablePrime(1) method. Since it's complexity is high and often or always returns in TLE during competitive programming.

⑤ Method-2 :- Prime Sieve (like sieve of stratosthenes) You know Odd this fn. ready made. And not directly to use in competitive coding main write up. So this will run only once. Saves time complexity.

Code :-

```
boolean[] primes = new boolean[100000];

static void fillSieve(){
    Arrays.fill(primes, true);
    primes[0] = primes[1] = false;
    for(int i=2; i<primes.length; i++){
        if(primes[i]){
            for(int j=2; i*j<primes.length; j++){
                primes[i*j] = false;
            }
        }
    }
}

public boolean isPrime(int n){
    return primes[n];
}
```

First fill all as true as taken all as prime

Pick ②

For first only that i to be multiplied whose is prime and as true and not false. If false means they are already taken care how

To every multiple of two until max array size mark them as false all 2 is factor.

So this is the concept

To them.

Time complexity :- ~~O(n log n)~~ ~~O(n log log n)~~ O(n log n)

⇒ But this is ~~good~~ for case you are finding all prime numbers upto number n.

#③ To check if after adding some value whether it is being already present or not.

Input / Output

e.g: Vicky → No
Vicky → Yes it's already present

Input / Output
3
Vicky → No
Vicky → Yes
Kumar → No.

You can use either Map or set which don't add duplicates in it.

Code :-

```
static steing[] ans (String[] arr){
    List<string> hs = new Arraylist<>();
    Hashset <string> store = new Hashset<>();
    for (int i=0; i<arr.length ; i++){
        if (store.contains (arr[i])){
            hs.add ("YES");
            store.add (arr[i]);
        }
        else{
            hs.add ("NO");
            store.add (arr[i]);
        }
    }

    string[] arr2 = new String [hs.size()];

    for (int i=0 ; i<hs.size(); i++){
        arr2[i] = hs.get (i);
    }

    return arr2;
}
```

input	output
3	
Vicky	NO
Vicky	YES

#④ Sum of first n odd numbers in O(1)

```
static int oddSum (int n) {
    return (n*n);
}
```

Sum of first n number $\frac{(n)(n+1)}{2}$

Approach ⇒ Since for n=2 ⇒ output = 1+3 = 4 = 2*2
n=5 ⇒ output = 1+3+5+7+9 = 25 ⇒ 5*5.

So sometimes questions in Problem has inner trick in it. The real trick is how you are solving the problem.

Trick-2 :- Number of occurrence of some digit e.g. (2) from 0 to n. So general formula

Input :- 22
Output :- 6

~~Count(digit) = 9*(count)~~

{2, 12, 20, 21, 22}

code: ~~static int numOf2sInRange (int n)~~ {

~~int co~~

Algorithm :- Efficient solution

count of numbers from 0 to 9 → 1
count of number from 0 to 99 → 1*9 + 10 = 19
count of number from 0 to 999 → 19*9 + 100 = 271

Scanned with OKEN Scanner

#(1) continues

To enter ~~0-5~~ multiple times ~~array~~ separated by space in same line :-

e.g. :-

```
1
4 3          → first
1 1 1 1 3 3 4 5 9 10 12 → second
```

```
BufferedReader br = new InputStream - - - - ;
int testCase = Integer.parseInt(br.readLine());
for(int t=0; t<testCase; t++){
    String lines = br.readLine();
    String[] str = lines.trim().split("\\s+");
    int len1 = Integer.parseInt(str[0]);
    int len2 = Integer.parseInt(str[1]);

    string lines2 = br.readLine();
    String[] str2 = lines2.trim().split("\\s+");

    for(int i=0; i<len1*len2; i++){
        arr[i] = Integer.parseInt(str2[i]);
    }

    Syso(ans);
}
```

To find ^nC_r efficiently in Mod & 10^9+7

However this is not exactly ^nC_r.
But we can see it's how we find factorial.

code :- In C++

using recursion in mod 10^9+7.

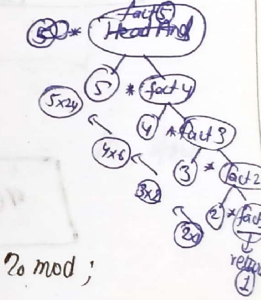
Also get concept here to find factorial

```
#include <bits/stdc++.h>
using namespace std;
long mod = 1000000000+7;
long fact(long M){
    if(M==0) return 0;
    if(M==1) return 1;
    else return ((M%mod)*(fact(M-1)%mod))%mod;
}

long powe(long N, long M){
    if(N-M==0) return 1;
    long x = N-M;
    long y = 1;
    while(x--){
        y = ((y%mod)*((N-M)%mod))%mod;
    }
}

int main(){
    long N, M;
    cin>>N>>M;
```

e.g. M = 5




```
    long mfact = fact(M);
    long mpow = powe(N,M);
```

Scanned with OKEN Scanner

A very normal factorial finding in modulo of 10^9+7. in Java

```
static long mod = 1000000000+7;

static long fact (long m) {
    if (m==0) return 1;
    if (m==1) return 1;
    else return ((m % mod) * (fact (m-1) % mod)) % mod;
}
```

can do this by DP also

DP[0]=1
DP[1]=1
DP[2]=2

otherwise if not modulo then,

```
return m * fact (m-1)
```

```
for(i=3; n)
    DP[i] = i * DP[i-1]
```

so do write

^nC_r we can now do as :- $^5C_2 = \frac{5!}{2!3!}$

```
long upper = fact(5);
long lower1 = fact(2);
long lower2 = fact(3);  (5-2)

long down = (lower1 * lower2) % mod;

long ans = (upper / down) % mod;

Syso(ans);
```

output = 10

$^5C_2 = \frac{5!}{2!3!} = 10$ Ans.

Scanned with OKEN Scanner

#④ continues.

Number of occurrence of number from 1 to n (e.g. that have 4 as a digit).

⇒ Count of numbers from 0 to 9 = 1
count of number from 0 to 99 = 1*9 + 10 = 19
count of number from 0 to 999 = 19*9 + 100 = 271

⇒ In General we can write

$$Count(10^d) = 9 * Count(10^{d-1}) + 10^{d-1}$$

// This can't be remembered just recap in notes.

code:

```
static int countNumbersWith4(int n) {
    if (n<4) return 0; //Base case

    int d = (int) Math.log10(n); //Number of digits minus one in n. For 328, d is 2.

    //computing count of numbers from 1 to 10^(d-1).
    //d=0; a[0]=0;   d=1; a[1]= Count of numbers from 0 to 9 = 1
    //d=2; a[2]= Count of numbers form 0 to 99 = a[1]*9+10 = 19
    //d=3; a[3]= " " from 0 to 999 = a[2]*9 + 100 = 271

    int[] a = new int[d+2];
    a[0] = 0;
    a[1] = 1;

    //Main Algorithm General Count.
    for(int i=2; i<=d; i++) {
        a[i] = a[i-1] * 9 + (int) Math.ceil(Math.pow(10, i-1));
    }

    int p = (int) Math.ceil(Math.pow(10,d)); //computing 10^d.

    int msd = n/p; //Most significant digit of n.
                   For 328, msd is 3 which can be obtained by 328/100.

    (1) //If MSD is 4, for example if n=428, then count of numbers
        //is sum of following.
        // (1) count of numbers from 1 to 399
        // (2) count of numbers from 400 to 428 which is 29.

    if (msd==4) {
        return (msd) * a[d] + (n%p) + 1;
    }
```

3 cases of MSD.
msd == 4
msd > 4
msd < 4

```
// case 2 if msD>4. for example if n=728. then
// count 8 numbers will be as following.
// ① count 8 numbers from 1 to 399 and count 8 numbers
     from 500 to 699, i.e., "a[2] * 6",
// ② count 8 numbers from 400 to 499, i.e. 100.
// ③ count 8 numbers from 200 to 228, recur for 28.

if (msd > 4) {
    return (msd-1) * a[d] + p + countNumbersWith4(n%p);
}

// If MSD<4. for example if n is 328, then count 8
// number is sum 8 following
// ① Count 8 numbers from 1 to 299
// ② count 8 numbers from 200 to 328, recur for 28.

return (msd) * a[d] + countNumbersWith4(n%p);

}
```

Scanned with OKEN Scanner

#5 Some important codechef concept Coding problem :- From october code

Cakewalk (Problem code: CKWLK) :- You are playing EVE online. Initially, you have one dollar, but you have somehow acquired two cheat codes. The first cheat code multiplies the amount of money you own by 10, while the second one multiplies ~~the~~ amount ~~of money~~ it by 20. The cheat codes can be used any number of times.

You want to have exactly N dollars. Now you are wondering: can you achieve that by only using some sequence of cheat codes :-

e.g.
1000 → (10 ^ 3) 10^3 ✓
2000 → 20×10×10 = 20×10² ✓
2040 → ×
2200 → ×
200 → ✓
400 → 20^2 ✓

{very good problem with good explanation & code.}

Example Input :-
4 → No. of testcases
200
90
1000000000000
1024000000000

Example output
Yes
No
Yes
No

All initially only K increase & check. once k will have value > n then increase

You can understand this code much better when you debug it.

o/c all understood it's working Hesaf.

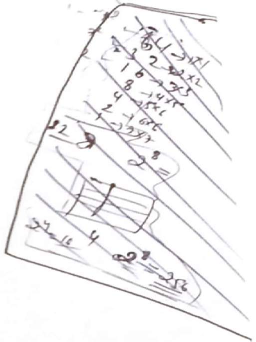
if n==Mathvalue → yes, ...

Solution code :-

```
P S V M {
  Scanner sc = new Scanner(System.in);
  int t = sc.nextInt();
  while (t-- > 0) {
     long n = sc.nextLong();
     long p = 1; //it's no use
     long i = 0, K = 0;
     while (K >= 0) { //Actual this condn always true until we reach some conclusion "Yes", "No" so try catch used.
        if (n > Math.pow(20,i) * Math.pow(10,K)) {
           K++;
        }
        else if (n == Math.pow(20,i) * Math.pow(10,K)) {
           System.out.println("Yes");
           break;
        }
        else {
           K = 0;
           i++; //i only increased after above two condn. unable.
           if (Math.pow(20,i) > n) {
              System.out.println("No");
              break;
           }
        }
     }
  ...
  Scanner ...All your code here
  }catch (Exception e) {
     return;
```

→ No idea why try catch was used. → This was used to remove runtime NZEC error.

OK Rest understood



Scanned with OKEN Scanner

Efficient Coding With Example Problems.

#1 To print the elements in 2-D array in single loop only :-

n → Array length

⇒ It's however like converting $O(n^2)$ in $O(n+n)$ but it's not.

⇒ However not sure if its exactly $O(n+n)$ since printing all element is however is stating $O(n*n)$ only. We are only just making only use of single while loop.

e.g. code :-

```
int[][] arr = {{1,2,3,4},{5,6,7,8},{10,20,30,40}};
int i=0;
int j=0;
while(i<arr.length){
    Sysо(arr[i][j]);
    if(j<arr[i].length){
        j++;
    }
    if(j==arr[i].length){
        i++;
        j=0;
    }
}
```

⇒ But this still in $O(n^2)$

Output ⇒ 1 2 3 4 5 6 7 8 10 20 30 40

Scanned with OKEN Scanner

#(2) TLE in finding pairs in array such that

arr[i] - arr[j] = k for given k.

⇒ Given arr = [1, 2, 3, 4]

key = 1

out pairs ⇒ (2,1), (3,2), (4,3)

Hence output = 3.

⇒ In O(n²) which got stuck in TLE.

```
static int pairs (int k, int[] arr){
    Arrays.sort(arr);
    int count = 0;
    for(int i=0; i<arr.length-1; i++){
        for(int j=i+1; j<arr.length; j++){
            if(arr[j]-arr[i]==k){
                count++;
            }
        }
    }
}
```

⇒ Now in O(n) using Two pointer. → Best for ?

⇒ ~~[illegible]~~.

⇒ The following code got accepted. (You may have to sort Arrays first or may be not) And first you've to sort it

```
// First sort the array.
static int pairs (int k, int[] arr){
    int i=0; int j=1; int n=arr.length;
    int count=0;

    while(j<n){
        int diff = arr[j]-arr[i];
        if(diff==k){
            count++;
            j++;
        }
        else if(diff>k){
            i++;
        }
        else if(diff<k){
            j++;
        }
    }

    return count;
}
```

O(n log n)

Can be done by Set better way

Scanned with OKEN Scanner

⇒ Also there's another solution for this question
O(n) time complexity by Umesh using set.
Hashset finds element in O(1) technique used here.

⇒ It will work for sorted, unsorted, duplicates or not.

Code in C++; O(n) time complexity.

It's Java code next page see

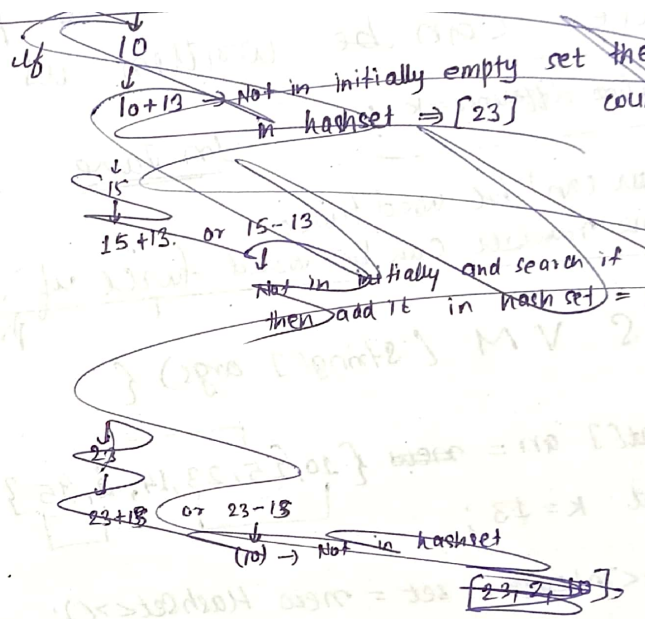
```
int main() {
    vector<int> arr = [10,15,23,14,2,15];
    int K=13;
    int n=arr.size(), count=0;
    set<int> s;
    for(int i=0; i<n; i++) {
        if(s.find(arr[i]+k) != s.end()) {
            count++;
        }
        else if(s.find(arr[i]-k) != s.end()){
            count++;
        }
        s.insert(arr[i]);
    }
    cout<<count;
    return 0;
}
```

Do in Java will do

write behind concept also

See next page.

output 3



Finally Now I understood it's work.

OK now understood

K=13

[10 15 23 14 2 15]

Not understood totally C++ stuff.

10 not in Hashset → add in hashset [10]

(10-13) or (10+13) Not Contain in Hash

count=0

Sel	count	i=	diff/sum
10	0	arr[0]=10	23, -3
[10,15]	0	arr[1]=15	2, 27
[10,15,23]	1	arr[2]=23	10(present) 33
[10,15,23,14]	1	arr[3]=14	27, 1
[10,15,23,14,2]	2	arr[4]=2	15 (repeat) 11
[10,15,23,14,15]	3	arr[5]=15	2, 27 (present)

But you might think (23) was there if we already have put all elements in Hashset, But it will take two count for (10,23) and (23,10) so it's better we take into count from 23 only and keep on steady adding. So understood now fully,

Scanned with OKEN Scanner

Simply in Java the previous code can be written as below:-

2) And no. of pairs whose difference = k :-

In Java

one pair can't be used twice
one number can be used twice if it is present twice

```
P S V M (String[] args) {

    int[] arr = new { 10, 15, 23, 14, 2, 15 };
    int k = 13;

    Set<Integer> set = new HashSet<>();
    int count = 0;

    for (int i=0 ; i<arr.length ; i++) {

        if (set.contains (arr[i]+k)) {
            count++;
        }
        else if (set.contains (arr[i] - k)) {
            count++;
        }
        set.add (arr[i]);
    }

    Syso (count);  // output = 3
}
```

i.e. 23 → 10 ⇒ 23-13 ⇒ 10
2 → 15 → 2+13 → 15
15 → 2 → 15-13 ⇒ 2

Scanned with OKEN Scanner

③ BigInteger 'OR' method to do bitwise 'OR' operation.

e.g

```
BigInteger bigInteger = new BigInteger ("2300");
BigInteger val = new BigInteger ("3400");

BigInteger ans = bigInteger.or(val);

Syso(ans); // 3580
```

Binary of 2300 = 100011111100
Binary of 3400 = 110101001000

OR of 2300, 3400 → 110111111100 → 3580.

⇒ Similarly we have

① and() method
② xor()
③ not()
④ getLowestSetBit().

Application of this OR in BigInteger in PES

② Hackerrank ACM ICPC Team

Given string[] arr ⇒ input ⇒

	member 1	2	3	4	5
1 →	1	0	1	0	1
2 →	1	1	1	1	0
3 →	0	0	0	1	1

Adding or ORing any two we want max possible 1's.

e.g.

Members (combination of two from)	Subjects	
(1,2)	[1,2,3,4,5]	⇒ max topic a pair team can know. = ⑤.
(1,3)	[1,3,9,5]	no. of such pair = ①
(2,3)	[1,2,3,9]	

So output 5,1

Code :-

```
static int[] acmTeam (String[] topic) {
    BigInteger[] bi = new BigInteger[n];
    int n = topic.length;

    for (int i = 0; i<n; i++) {
        bi[i] = new BigInteger (topic[i], 2);
    }
    int maxTopic = 0;
    int teamCount=0;

    for (int i=0; i< n; i++) {
        for (int j = i+1; j<n; j++){
            BigInteger iuj = bi[i].or(bi[j]);
            int bitCount = iuj.bitCount();
            if (bitCount > maxTopic) {
                maxTopic = bitCount;
                teamCount =1;
            } else if (bitCount == maxTopic) {
                teamCount++;
            }
        }
    }
```

Scanned with OKEN Scanner

```
    int[] result = {maxTopic, teamCount};
    return result;
}
```

So for input :-

```
4 5
10101
11100
11010
00101
```

→ output

```
5
2
```

(But it's $O(n^2)$ soln.) check if it worked in Hackerrank.

explanation :-

(1,2) → 4
(1,3) → 5
(1,4) → 3
(2,3) → 4
(2,4) → 4
(3,4) → 5

The 2 team (1,3) & (3,4) know all 5 topics which is maximal.

Scanned with OKEN Scanner

#4 Window Sliding Technique :-

⇒ This technique shows how a nested for loop in few problems can be converted to single for loop and hence reducing the time complexity.

⇒ Problem statement :-

Given an array of integeres of size 'n'. Our aim is to calculate the maximum sum of 'k' consecutive elements in the coreay.

input[] arr ⇒ arr[] = {100, 200, 300, 400}

K = 2

output = 700.

So it's for K consecutive elements or K subarray elements

Since consecutive elements of size k so why to sort it first so

we don't have to do sorting first to apply sliding window technique if we want consecutive

⇒ using brute-force our code will be as :-

```
static int maxSum (int arr[], int n, int k) {
    int max_sum = Integer.MIN_VALUE;

    for(int i=0; i<n-k+1; i++) {
        int current_sum = 0;
        for(int j=0; j<k; j++) {
            current_sum = current_sum + arr[i+j];
        }
        max_sum = Math.max(current_sum, max_sum);
    }
    return max_sum;
}
```

⇒ The above code time-complexity is O(k*n) as it contains 2 nested loops.

Scanned with OKEN Scanner

Window sliding Technique :-

for sub Array → Mean K consecutive elements

consider an array arr[] = {5, 2, -1, 0, 3}

and K = 3 , n = 5 .

our initial window

our next window and so on.

add subt.

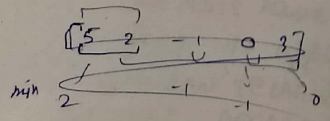
current_sum = window_sum

current_sum = window_sum + 0 - 5

So in this window just add the next element and remove (subtract) the 1st element of window and so on. it goes in single loop.

→ But first we've to create this window and then later use this window in our single for loop.

code as.



→ code as :- using

```
static int maxSum (int arr[], int n, int k){
    // k must be greater
    if(n<k){
        Syso("Invalid");
        return -1;
    }

    //compute sum of first window of size k
    int max_sum = 0;
    for (int i=0; i<k; i++){
        max_sum += arr[i];
    }

    //compute sums of remaining windows by removing first element of
    // previous window and adding last element of current window
    int window_sum = max_sum;
    for (int i=k; i<n; i++){
        window_sum += arr[i] - arr[i-k];
        max_sum = Math.max (max_sum, window_sum);
    }
    return max_sum;
}
```

→ Here Time complexity is O(n)

#5 Two Pointers Technique :-

(Array must be sorted)

⇒ Two pointers is an easy and effective technique which is typically used for searching pairs in a sorted array.

⇒ Problem statement :- Given a sorted array A (sorted in ascending order), having N integers, find if there exists any pair of elements (A[i], A[j]) such that their sum is equal to X.

⇒ Lets see naive brute force approach. Time = $O(n^2)$

```
static boolean isPairSum (int A[], int N, int X){
    for (int i=0; i<N; i++){
        for (int j=0; j<N; j++){
            if (A[i]+A[j] ==X){
                return true;
            }
            if (A[i] + A[j] >X){
                break;   //as the arr is sorted.
            }
        }
    }
    return false;
}
```

⇒ Now by two pointer. one for first element other for last element in the array. we add the values kept at both the pointer.

If their sum is smaller than X then we shift the left pointer to right (since sorted in ascending) or if their sum is greater than X then we shift the right pointer to left, in order to get closer to the sum. we keep moving the pointer until we get the sum as X. (Also this can be used to count pairs whose sum is X)

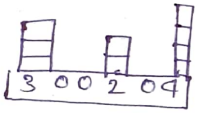
code as :

```
static boolean isPairSum (int A[], int N, int X) {
    int i = 0;
    int j = N-1;
    while (i<j) {
        if (A[i] + A[j] == X){
            return true;        count++; (in case we want to find number of pairs whose sum is X.
        }
        else if (A[i] + A[j] < X){
            i++;
        }
        else {
            j++;
        }
    }
    return false;
}
```

Scanned with OKEN Scanner

#6 Trapping Rain Water :- (Some optimized technique)

⇒ This tells a way to use techniques to do in O(n) rather than O(n²).



Total trapped water = 3+3+1+3 = 10.

⇒ A simple solution is to traverse every array element and find the highest bars on left and right sides. Take the smaller of two heights. The difference between the smaller height and height of the current element is the amount of water that can be stored in this array element.
Time complexity = O(n²).

⇒ An efficient solution is to pre-compute highest bar on left and right of every bar in O(n) time. Then use these pre-computed values to find the amount of water in every array element. Time complexity = O(n).

⇒ code as :-

```
static int findWater(int n){

    int left[] = new int[n];

    //left[i] contains the height of tallest bar to the left of
    //ith bar including itself.

    //right[i] contains height of tallest bar to the right of
    // ith bar including itself.
    int right[] = new int[n];

    int water = 0;

    //fill left array.
    left[0] = arr[0];
    for(int i = 1; i<n; i++){
        left[i] = Math.max(left[i-1], arr[i]);
    }

    //fill right array:
    right[n-1] = arr[n-1];
    for(int i = n-2; i>=0; i--){
        right[i] = Math.max(right[i+1], arr[i]);
    }

    //calculate the accumulated water element by element: consider the water amount on ith bar, the
    // amount of water accumulated on this particular bar will be equal to min(left[i],right[i])-arr[
    for(int i=0; i<n; i++){
        water += Math.min(left[i],right[i]) - arr[i];
    }
    return water;
}
```

Scanned with OKEN Scanner

#7 Some Concept understanding Problems in Codechef :-

⇒ Don't copy solution. But learn the concept and solve yourself. ⇒ Couldn't totally understand the logic of this question soln. check it and find it.

⇒ In codechef not all question you've to apply loop and take time complexity to O(n).

⇒ If while running in O(n) you are getting TLE, means understand there is some conceptual, mathematical beide question which can be solved in O(1) only so they put the question to TLE if you do in O(n).

⇒ Example Question from codechef :- September Cook

(Q) Choose exactly two integers (not necessarily distinct) from the interval [1, R]. Find the maximum possible value of it the bitwise OR of the chosen integers.

Example input :-

	output
4 → No. of test case	
1 3	3
4 5	5
1031 93714	131071
1000000000121 1000000123453	1099511627775

(S) Before moving to answere directly lets first understand one basic maths operation in Java.

E.g Given a number lets say '11' (11) who binary representation as = 1011

Here it's highest one bit is ↑ this one

so it's highest one bit value = 1000 ⇒ 8

1 0 1 1 → 11

↑ Highest one bit

Highest one bit value 1000 → 8.

Now to make all these Last 3 bit's also zere than we need a number 1 less than 8. i.e. 7. whose binary repsn as 0111 → 7.

∴ (OR)

1000	⟷	1011 → 11
0111		(OR) 0111 → 7
1111 → 15 Ans		1111 → 15 Ans

so in our question asked back there we want maximum possible 1's in our number resultant from OR of two numbers.

Scanned with OKEN Scanner

Example code to understand `Long.highestOneBit(long)` method.

```
long l = 250
Syso("Binary repn" = " + Long.toBinaryString(l));
Syso("Number of one bit's =" + Long.bitCount(l));
Syso("Highest one bit=" + Long.highestOneBit(l));
```

output :-

```
Number = 250
Binary repn = 11111010
Number of one bits = 6
Highest one bit = 128 (i.e. 10000000)
```

⇒ Also rem! precedence of

(*, /, %, +) > (|, ^, &)

Arithmetic op. Bitwise operation

So our code is as :- Ans for our problem :-

```
static long maxXorInRange(long L, long R){
    if(L==R){
        return L;   // means only [11] -> then return 11 only.
    }
    else{
        long value = L^R;
        long h = Long.highestOneBit(value);
        return (R/((h*2)-1));
    }
}

public static void main{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int test = Integer.parseInt(br.readLine());
    while(test-->0){
        StringTokenizer str = new StringTokenizer(br.readLine());
        long L = Long.parseLong(str.nextToken());
        long R = Long.parseLong(str.nextToken());
        System.out.println(maxXorInRange(L,R));
    }
}
```

Input

4
1 3
9 5
1031 93714
100000000012 100000012345678

→ output ⇒

3
5
121071
100000019251?1

All correct In O(1) time complexity

let's understand it's working :- ex

example L = 9
R = 12 then

```
L      1 0 0 1   → 9
R (xor)1 1 0 0   → 12
       0 1 0 1   → value (5)
```

long h = highestonebit(5) ⇒ 101

100 → i.e 4

∴ h = 4

∴ Ans ⇒ (R/((4*2)-1))

⇒ (12/7) ⇒

```
(OR) 1 1 0 0
     0 1 1 1
     1 1 1 1  → 15 Ans
```

100***

even though '7' not in range

Soln

our main aim was to bring

(#) Other way to solve this :-

```
long l = Long.parseLong(str[0]);
long r = Long.parseLong(str[1]);

long ans = 0;
long j = l;

while ((j/j+1) < r) {
    j = j/(j+1);
}

ans = j/r;
sysо(ans);
```

e.g.

9 (L) 12 (R)

9/10 → and(11 < 12)

∴ j = 11

12/12 → while (12 ≯ 12)

∴ j = 11 . out of loop.

∴ ans = 11/12

ans → 12 Ans .

e.g. if

6 (L) 9 (R) .

while ((6/7)^7 < 9)

j = 7

while (7/7+1 ≯ 9)

7/7+1 = 15

∴ out of loop

and j = 7 only

∴ ans = 7/9

So output 15 Ans .

So ok understood this pgm also.

Scanned with OKEN Scanner

#@ Count distinct elements in every windows of size K.:-

⇒ This is one of the most frequently asked coding problem. Let's see how to solve it in O(n) rather than in O(nk). (which often leads to TLE).

⇒ Example:- input:- arr[] = {1, 2, 1, 3, 4, 2, 3}
K = 4

Output:-
3 → {1 2 1 3} → distinct 3
4 → {2 1 3 4} → distinct 4
4 → {1 3 4 2} → distinct 4
3 → {3 4 2 3} → distinct 3 elements

⇒ 2y

⇒ You can understand this code by debugging it it's optimal code (#B).

⇒ Although now you understand everything.

Map

⇒ arr[i] → key ; No. of its element → value (duplicates count)

#A Simple solution :- Time = O(nk²)

```
//count distinct elements in window of size K
static int countWindowDistinct(int win[], int k){
    int dist_count = 0;
    //traverse the window
    for(int i=0; i<k; i++){
        //check if element arr[i] exists in arr[0..i-1]
        int j;
        for(j=0; j<i; j++){
            if(win[i] == win[j])
                break;
        }
        if(j == i)
            dist_count++;
    }
    return dist_count;
}
```

```
//count distinct elements in all windows of size k
static void countDistinct(int arr[], int n, int k){
    //traverse through every window
    for(int i=0; i<=n-k; i++){
        System.out.println(countWindowDistinct(Arrays.copyOfRange(arr, i,
                arr.length), k));
    }
}
```

#(B) An efficient solution :- is to use the count of the previous window while sliding the window.
The idea is to create a hash map that stores elements of the current window. When we slide the window, we remove an element from the hash and add an element.

For further understanding of algo see from GeeksforGeeks: count-distinct-elem

Code :-

Time complexity : O(n)

```
static void countDistinct (int arr[], int k) {
    //create an empty hashMap hM
    HashMap<Integer, Integer> hM = new HashMap< >();
    //initialize distinct element count for current window.
    int dist_count = 0;
    //Traverse the first window and store current count of every
    //element in hash map.
    for(int i=0; i<k ; i++) {
        if (hM.get(arr[i]) == null) {
            hM.put(arr[i], 1);  //Key is arr[i] and number of it's content as value
            dist_count++;
        }
        else {
            int count = hM.get(arr[i]);
            hM.put (arr[i], count+1);
        }
    }
    //print count of first window
    Syso (dist_count);
    //Traverse through the remaining array.
    for(int i=k ; i<arr.length ; i++) {
        //Remove first element of previous window
        //If there was only one occurrence, then
        //reduce distinct count.
        if (hM.get(arr[i-k] == 1) {
            hM.remove(arr[i-k]);
            dist_count--;
        }
        else {  //reduce count of the removed element
            int count = hM.get(arr[i-k]);
            hM.put(arr[i-k], count-1);
        }

        //Add new element of current window
        //If this element appears first time,
        //increment distinct element count.
        if (hM.get(arr[i]) == null) {
            hM.put(arr[i], 1);
            dist_count++;
        }
        else {  //Increment distinct element count.
            int count = hM.get(arr[i]);
            hM.put(arr[i], count+1);
        }

        //Print count of current window
        Syso (dist_count);
    }
}
```

Scanned with OKEN Scanner

Before going further; let's first understand some codes in HashMap. However it's internall working is explained in detail in Java - 2 Note book :-

OR Just Map in Java

⇒ code :-

```
HashMap<Integer, String> map = new HashMap<>();
map.put (1, "hello");
map.put (2, "world");
map.put (3, "how");
map.put (4, "do");

for (int i=1 ; i<=map.size(); i++) {
    Syso (map.get (i);
}
```

//output :-
hello
world
how
do.

***

```
HashMap < key, value > map = new HashMap<>();
map.put (key, value);
map.get (key);
map.remove(key);
```

~~map.getKey();~~

For further details see in coding blocks (our book).

Scanned with OKEN Scanner

Code chef contest questions :-

⇒ My code run well in IntelliJIdea but showed wrong Ans in codechef while submitting it.

⇒ It also initially was showing NZEC runtime error. which after putting try{ }catch(exception e) return; it was do removed.

⇒ Although still my code gave WA after submitting.

① Gilfoyle Vs Dinesh :- Problem code : GVR :-

- The number of prime numbers in the first half of the array is equal to the number of prime numbers in the second half.
- The first prime number in the first half is more than the last prime number in the second half.

If the given array satisfies the above conditions, then the array is PERFECT otherwise its IMPERFECT.

Input :

4 → size of Array
1 3 2 4 → PERFECT

[1 prime in first half (3)
1 prime in second half (2)
(1=1) ⟷ (3>2) Perfect]

6
1 1 5 6 4 8 → IMPERFECT

[1 prime in first half
0 prime in second half
↳ (1≠0) → imperfect.]

My code :- (WA) (wrong Ans)

```
static boolean isPrime(int num){
    if(num==0) return false;
    if(num==1) return false;
    if(num==2) return true;
    for(int i=2; i<=Math.sqrt(num); i++){
        if(num%i==0){
            return false;
        }
    }
    return true;
}

P        S        V        M() throws java.lang.Exception{

    try{
        BufferedReader br=new ----in));
        int len = Integer.parseInt(br.readLine());
        String lines = br.readLine();
        String[] str = lines.trim().split("\\s+");
        int[] arr = new int[len];
        for(int i=0; i<arr.length; i++){
            arr[i] = Integer.parseInt(str[i]);
        }
```

Scanned with OKEN Scanner

```
// Main code

int firstElement = 0, lastElement = 0;
int firstHalf = 0, secondHalf = 0;
int mid = arr.length/2;

for (int i = 0; i < arr.length; i++) {
    if (i < mid) {
        if (isPrime(arr[i])) {
            firstHalf++;
            if (firstElement == 0) {
                firstElement = arr[i];
            }
        }
    }

    if (i >= mid) {
        if (isPrime(arr[i])) {
            secondHalf++;
            lastElement = arr[i];
        }
    }
}

if (firstHalf == secondHalf) {
    if (lastElement < firstElement) {
        Sysо("PERFECT");
    }
    else {
        Sysо("IMPERFECT");
    }
}
else {
    Sysо("IMPERFECT");
}

} catch (Exception e) {
    return;
```

can be this guilty error instead we could have taken (-1)

(#) From my opinion his time complexity $O(\frac{n}{2})$ min $O(n)$ } to maybe his accepted otherwise overall both's time = $O(n)$, since we take max possible so.

⇒ There was 2 Advantages in his solution code which got accepted

(1) $O(\frac{n}{2})$ → It's check first and second half separately. So $O(\frac{n}{2})$.

(2) It uses long type for array.

(3) Isprobableprime to check prime

Mine

(1) I did $O(n)$ in single did all checking for both first half and second half.

(2) I use int[] array.

(3) $O(\sqrt{n})$ to check prime.

Scanned with OKEN Scanner

Someone's code which was accepted (AC) All correct:

User :- zeeshan12 (1*)

```
static boolean checkPrime(long n){
    BigInteger b = new BigInteger(String.valueOf(n)); // converting long to BI
    return b.isProbablePrime(1);
}

P S V M() throws java.lang.Exception {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    long a[] = new long[n];
    for(int i=0; i<n; i++){
        a[i] = sc.nextLong();
    }
    long first = -1, last = -1;
    int c1 = 0, c2 = 0;

    for(int i=0; i<n/2; i++){
        if(checkPrime(a[i])){
            c1++;
            if(first == -1){
                first = a[i];
            }
        }
    }
    for(int i=n/2; i<n; i++){
        if(checkPrime(a[i])){
            c2++;
            last = a[i];
        }
    }
    if(c1==c2){
        if(first>last){
            Syso("PERFECT");
        }
        else{
            Syso("IMPERFECT");
        }
    }
    else{
        Syso("IMPERFECT");
    }
}
```

⇒ This soln. is totally same as mine.

Difference

(int used) Mine (WA)	Someone's (AC) (long used)
① O(n) (once)	① O(n/2) twice to separately done
② I' used buffered Read	② scanner
③ Array was ~~long~~ int type	③ Array was long type
④ Prime check O(√n)	④ Prime check using BigInteger

Scanned with OKEN Scanner

(#) Someone's code which was accepted (AC) All correct:

User :- zeeshan12 (1*)

```
static boolean checkPrime(long n){
    BigInteger b = new BigInteger(String.valueOf(n)); // converting long to Big
    return b.isProbablePrime(1);
}

P S V M() throws java.lang.Exception {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    long a[] = new long[n];
    for(int i=0; i<n; i++){
        a[i] = sc.nextLong();
    }
    long first = -1, last = -1;
    int c1 = 0, c2 = 0;

    for(int i=0; i<n/2; i++){
        if(checkPrime(a[i])){
            c1++;
            if(first == -1){
                first = a[i];
            }
        }
    }
    for(int i=n/2; i<n; i++){
        if(checkPrime(a[i])){
            c2++;
            last = a[i];
        }
    }
    if(c1==c2){
        if(first>last){
            sysO("PERFECT");
        }
        else{
            sysO("IMPERFECT");
        }
    }
    else{
        sysO("IMPEREECT");
    }
}
```

⇒ This soln. is totally same as mine.

Difference		
(int used)	Mine (WA)	Someone's (AC) (long used)
①	O(n) (once)	① $O(\frac{n}{2})$ twice to separately done
②	I' used buffered read	② scanner
③	Array was ~~long~~ int type	③ Array was long type
④	Prime check O(√n)	

Some easy codechef contest Question done easily :-

⇒ My code (AC) :- accepted

⇒ But here we've to take care of all different cases and worked out.

Q Nettila in Hyderabad :- Code = ENOC2

We've two number A and B and could do the following operations on them.

- A can be incremented to 2A (only multiply by 2 for A)
- B can be decremented to B-2. (only (-1) by 2 for B)

By doing these operations any number of times (including zero). can A and B be made equal?

→ To me it looks dynamic programming approach to apply.

Sample Input :-

2 → No. of test case
4 8 → A & B → Yes
5 15 → A & B → Yes.

4*2 = 8 or . 8-2-2=4 (Yes)

15 - 2 - 2 - 2 - 2 - 2 = 5 (Yes)

My Approach :-

(A ≤ B) always
even even → Yes (always)
odd odd → Yes (always)

A ≤ B
odd even
(3*2) < 10 → OK
(2 10) X No
Yes.

A ≤ B
even odd
(2*2) 5 → never

```
static void ans (int A, int B) {
    if (B >= A) {
        if (A%2==0 && B%2==0) {
            Syso ("YES");
            return;
        }
        if (A%2==1 && B%2==1) {
            Syso ("YES");
            return;
        }
        if (A%2==1 && B%2==0) {
            if ((A*2) <= B) {
                Syso ("YES");
                return;
            }
            else {
                Syso ("No");
                return;
            }
        }
        if (A%2==0 && B%2==1) {
            Syso ("No");
            return;
        }
    }
    else {
        syso("No");
        return;
    }
}
```

```
public SVM1 {
    Scanner sc = new Scanner(...);
    int t = sc.nextInt();
    while (t-->0) { int A = sc.nextInt;
                    int B = sc.nextInt;
```

All correct

Scanned with OKEN Scanner

(7) Codechef Fast I/O also sometimes may taking part in returning AC.

e.g.

(8) Help Akash!:- Given number N, a → its digits

return $x_i = a_i \oplus a_{i+1}$

$x_n = a_n \oplus a_0$.

e.g. N=1234 →

$x_1 = 1 \oplus 2 = 3$

$x_2 = 2 \oplus 3 = 1$

$x_3 = 3 \oplus 4 = 7$

$x_4 = 4 \oplus 1 = 5$

Hence answer = 3175.

5 →

N=572354

2 5 1 0 1 3 1 1 → output

(9) My code not accepted (WA):-

```
Scanner sc = new Scanner(System.in);
int t = sc.nextInt();
while (t-- > 0) {
    int input = sc.nextInt();
    Integer integer = input;
    String str = integer.toString();
    String ans = "";
    for (int i = 0; i < str.length() - 1; i++) {
        int point = (int) str.charAt(i) ^ (int) str.charAt(i+1);
        ans += point;
    }
    ans += (int) str.charAt((str.length()-1)) ^ (int) str.charAt(0);
    sysO(ans);
}
```

→ Also tried input as strip only still (WA).

run fine in IntelliJ IDEA with

input

3

1 2 3 4

6 0 2 3 5

5 7 2 8 5 4

output

→ 3175

→ 6 2 1 0 1 3 3

→ 2 5 1 0 1 3 1 1

All sample case working fine

(5) But someone's else code in my same way works fine and has been accepted

```
Reader reader = new Reader();
Writer writer = new Writer();

int iterations = reader.nextInt();
while (iterations-- > 0) {
    String number = reader.nextString();
    int len = number.length();
    if (len == 1) {
        writer.println(number);
        continue;
    }
    StringBuilder sb = new StringBuilder("");
    for (int i = 0; i < len - 1; ++i) {
        int first = number.charAt(i);
        int second = number.charAt(i+1);
        sb.append((first + second));
    }
    int last = number.charAt(len-1);
    int around = number.charAt(0);
    sb.append((last + around));
    writer.print(sb);
}
writer.close();
```

Same as this testcase you were not handling

Also in this code there were self defined Reader class and writer class for fast I/O operations.

⇒ But main problem may be that I was not handling the test case where (len == 1) so might my ans was showing (WA).

ok all understood.

⇒ yeah got it that's the mistake I was making:-

So for input: 1 → No. of test case, 5 → num → output = 5

However my code output will be 0. so of course it will show (WA).

⇒ For writer and Reader class code see in IdeaProjects/OctoberF2 → folder in github.

Again done in codechef repo in Github.

myCodechefPractise/FastReaderWriterClass

(8) There must be something I'm not able to get. Bcoz even after applying that skipped testcase & len == 1 my code was still not accepted.

i.e. in my code after edition :-

```
try {
    Scanner
    |
    |
    |
    for
    !
    }
    ans +=
    |
    if (str.length() == 1) {
    !     syso (input);
    |   }
    |   else {
    |   !     syso (ans);
    |   !
    |   ]
    }
    Catch (Exception e) {
    !     return;
    !
    ]
```

→ Applying the missing test case.

Even though this code on submission gave (WA).

(9) At last someone's code I've to submit! And it was submitted with AC.

His Reader class is as :-

```
class Reader {
        private byte[] buf = new byte [1024];
        private int index;
        private InputStream in;
        private int total;
        public Reader () { in = System.in; }
        public int next() throws IOException {
            if (total < 0) {
            !     throw new InputMismatchException();
            }
            if (index >= total) {
                index = 0;
                total = in.read(buf);
                if (total <= 0)
                !     return -1;
                }
```

Rest code is in mentioned folder.

⇒ Also rem! → Don't only try to go with single way, like here we are concatenating it as `str += A^B` we can do by `StringBuilder.concat()`.

Scanned with OKEN Scanner

Some very unusual tricky question with very simple easy soln.

Source Hackerearth :-

Monk and Binary Array :-

	Sample Input	Sample output
arr	3	4
Constant	1 0 0	

output :- subarray such that their XOR = 1.
max possible such thing.

eg. Also you can only can flip bit from 0 to 1 or 1 to 0.

$\therefore$ 1 0 0 → {1}, {1,0} → {1,0,0} → 3 possible
(1 ⊕ 0 = 1, 1 ⊕ 0 ⊕ 0 = 1)

1 0 1 → {1}, {1,0}, {0,1}, {1} → 4 possible subarray

(101)x → 1⊕0⊕1 → 0 to.

```
#include <stdio.h>
int main() {
    long n,x;
    scanf("%ld",&n);
    x=n/2;
    printf("%ld",(x+1)*(n-x));
    return 0;
}
```

→ Also this ans is independent of actual bit values.

→ This simply code passed all the testcases

But for 000 → It won't never pass so this is some editors mistake

So.

It also even don't require input of arr elements.

in Java also it works :-

```
long n,x;
Scanner sc = new Scanner(System.in);
n=sc.nextLong();
x=n/2;
System.out.println((x+1)*(n-x));
```

All test case passed.

⇒ It's so unusual to see such things in coding competition

⇒ Ans got from comment section

Scanned with OKEN Scanner

Codechef Contest Question understanding :-

RUN, CHEF, RUN !! : Problem code:- ENOC4 :-

Chef took part in a race event, in which the length of racing tracks is N miles. However there are checkpoints at an interval of K miles from the starting point onward, i.e., at the distance of K, 2K, 3K ... from the starting point. On reaching each checkpoint Chef has to return back to the starting point to restart his race, and the visited checkpoint is removed.

Find out the total distance he has to cover in order to complete the race successfully. The race is said to be successfully completed if all the checkpoints at a distance d ($d \leq N$) is removed.

The result can be very large, so compute it modulo 1000000000+7 (10^9+7).

Input:-

2 → No. of test case

N ← 3 2 → K

N ← 4 2 → K

→ Output:-

7

16

1st case :- N=3, & K=2;

therefore, there is only one checkpoint in the racing track. so chef run 2 miles and reaches the checkpoint. From her he has to return back to starting point i.e. 2 miles. Now there no more checkpoints. rest +3,

∴ 2+2+3 = 7 miles

My code :- Time $O(n/k)$ worst case $k=1$ ∴ $O(n)$

(TLE)

```
static long mod = 1000000000+7;

p       s       v       M ( )  {
    Scanner sc = new Scanner (System.in);
    int t = sc.nextInt();
    while (t-->0) {
        long N = sc.nextLong();
        long K = sc.nextLong();
        long i = 1;
        long sum = 0;
        while (K<=N) {
            sum = (((sum)%mod) + ((K*2)%mod))%mod;
            i++;
            K = (((K)%mod) * (i%mod))%mod;
        }
        Sysa (sum+N);
    }
}
```

Scanned with OKEN Scanner

(11) someone's AC code :- Time O(1)
All correct (AC)

```
P        S        V        M() throws Exception {
    BufferedReader br = new BufferedReader(new Inp--- (System.in));
    StringTokenizer st;
    PrintWriter pw = new PrintWriter(System.out);
    int t = Integer.parseInt(br.readLine());
    while (t-->0) {
        st = new StringTokenizer(br.readLine());
        long n = Long.parseLong(st.nextToken());
        long k = Long.parseLong(st.nextToken());
        long checkpoints = n/k;
        long ans = ((checkpoints % 1000000007) * ((checkpoints+1)%1000000007))
                   % 1000000007 * k;
        ans %= 1000000007;
        ans += n;
        ans %= 1000000007;
        pw.println(ans);
    }
    pw.flush();
    pw.close();
}
```

copied

After rewriting my code as :- scanner only.

```
while (t-->0) {
    long n = sc.nextLong();
    long k = sc.nextLong();
    long checkprint = n/k
    ;
    ;
    ;
    ans %= 1000000007;
    sysO (ans);
```

copied from someones

My code runned well with (AC). All correct ans.

So here you can conclude programming coding is more like maths than coding)

⇒ So conclusion, try those questions especially maths type to solve in single O(1) rather than looping it, use more maths way to solve problem.

Scanned with OKEN Scanner

(8) Maths involved in this program was :-

⇒ so always try to manipulate your coding & programming like a mathematician more than a coder.

⇒ Most of the problems works in this way. The trick is finding how maths is involved.

⇒

N	K
3	2

→ checkpoints = $\frac{3}{2} = 1$

long ans = $((1) * (1+1)) * 2$ ← K

$= (1 * 2) * 2$

$= 4$

(ans += n)

ans = ans + n = 7.

∴ Ans = 7

N	K
4	2

⇒ checkpoints = $\frac{4}{2} = 2$

long ans = $((2) * (2+1)) * 2$ ← K

(ans = (2 * 3) * 2 = 12)

ans += n ⇒ (ans = 12 + 4 = 16)

∴ Ans → 16

Scanned with OKEN Scanner

(#) Codechef questions :- In which you not fully cared about the very good testcase :-

i.e. If hes hal $l=0$ already, then he can't even cross the first ~~at~~ barrier also if not in his range then going and checking further barrier will make no sense. so this part I missed in the testcase so overall my whole code is wrong. That's why WA.

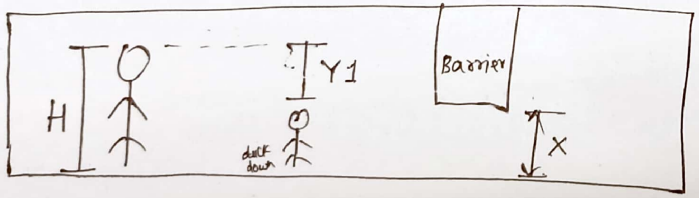
D'code contest

(2) FULL BARRIER Alchemist :-

Edward Elric is chasing after Scar. To stop Edward, Scar creates N barriers in the way, numbered from 1 to N. Each barrier Scar created is either one of the following two types.

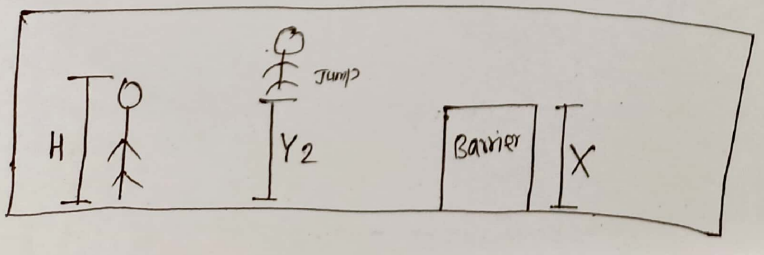
- Type 1 barrier - This barrier start from ~~origin~~ a height X above the ground and extended till the sky.

Type 1



- Type 2 barrier - This barrier start from the ground and extend up to height X above the ground.

Type 2



The height of Edward is H units and he has an alchemic life force L units.

Moreover, he can duck by Y_1 units and jump by height Y_2 units (as explained in fig). He started crossing barriers in sequence, starting from barrier ~~one~~ 1 till the barrier N. whenever he can't pass a barrier by ducking and jumping (considered passed even when the barrier just touches him). he uses Alchemy to break the barrier. However, this costs him a single unit of his alchemic life force.

If after breaking a barrier no life force is left, Edward gets completely exhausted, unable to pass that barriere.

How many barrier can Edward cross ?

Sample input

~~1~~ → No. of testcase

N	H	Y1	Y2	L
6	5	1	2	3
2	2			
2	1			
1	10			
2	8			
2	4			
1	2			

→ output : 5

Exp

Given $N=6$, $H=5$, $Y_1=1$, $Y_2=2$ & $L=3$

He passes the first three barriers by either ducking or jumping. He use alchemic life force for breaking ~~the~~ 5th & 6th barrier because he cannot pass them by either jumping or ducking. He gets exhausted after breaking the 6th Barrier and is unable to pass it. so in total he passes

#My solution WA :- Didn't include all the test case so :-

Although all the test case passed but that not include all the special & edge case.

```
int n = sc.nextInt();
int h = sc.nextInt();
int y1 = sc.nextInt();
int y2 = sc.nextInt();
int l = sc.nextInt();

int notzero = l;
int[][] arr = new int[n][2];

for(int i=0; i<n; i++){
    arr[i][0] = sc.nextInt();
    arr[i][1] = sc.nextInt();
}

int count = 0;

for(int i=0; i<n; i++){
    if(arr[i][0] == 1 && arr[1][1] >= (h-y1)){
        count++;
    }
    else if(arr[i][0] == 2 && arr[i][1] <= y2){
        count++;
    }
    else if(l>0){
        count++;
        l--;
    }
}

if(l==0 && notzero!=0){
    count--;
}

Sysо(count);
```

#Someone's correct Ans (AC) :-

Because he checked all the edge & special case which you missed (you/me)

```
int n, h, y1, y2, l = sc.nextInt();
int[][] arr = new int[2][n];
for(int i=0; i<n; i++){
    arr[0][i] = sc.nextInt();
    arr[1][i] = sc.nextInt();
}

int count = 0;
for(int i=0; i<n; i++){
    if(l>0){ //This take care if at start only he can't cross. And l=0 then he can mean cross first so check first worm possible so break in else part.
        if(arr[0][i] == 1){
            if((h-y1) <= arr[1][i]){
                count++;
            }else{
                if(l!=1){
                    count++;
                }
                l--;
            }
        }else{
            if(y2 >= arr[1][i]){
                count++;
            }else{
                if(l!=1){
                    count++;
                }
                l--;
            }
        }
    }else{break;}
}

Sysо(count);
```

Also if l=0 then still he can't cross since his l=0. He gets exhausted so. Even though the next barrier can be crossed if his l=0 He stops there only so.

Scanned with OKEN Scanner

Codechef Questions :- (Learning to understand type of questions)

It's one of a kind of question which is very difficult to approach.
Let's see how such coding problems are approached to solved without getting TLE.

1) Code contest :- Grumpy Granny :- Problem code :- GRUPMA :-

After the death of their mother, Alphonse and Edward now live with Pinako and Winry.

Pinako is worried about their obsession with Alchemy, and that they don't give attention to their studies.

So to improve their mathematical solving ability, every day she gives a mathematical problem to solve. They can go out to practice Alchemy only after they have solved the problem. Help them by solving the given problem, so that they can go early today for their Alchemy practise.

Problem

Given an array A of N non-negative and two integeres K and M. Find the number of subsequences of array A of length K which satisfies the following property :-

Suppose the subsequence is $S = S_1, S_2, \ldots S_k$, then for all i such that $1 \le i \le k$.

$$S_i \% M = i \% M$$ should hold true,

where S_i denotes i^{th} element of the subsequence, using 1-based indexing.

As the number of subsequences may be very large, output the answer modulo 1000000007.

Constraints
$1 \le N \le 10^9$ → so only O(n2)
$0 \le A_i \le 10^9$ → so take Mod

Sample Input

1 → No. of test case.

12 4 3 → M.
Size of subsequence
Length of A array

4 5 6 7 1 4 6 9 0 0 10 2

→ output = 8.

taken separately for each 0.

[1 2 0 1 1 1 0 0 0 0 1 2] ← you can have followed this approach also. It's simple LCS pattern matching

Explanation :- The subsequences of length 4, satisfying the given criteria are.

[4,5,6], [4,5,6,10], [4,5,6,10], [4,5,6,1], [4,5,9,10], [4,5,6,4], [4,5,0,10], [4,5,0,10].

This accounts for a total of 8 valid subsequences.

Let us take one subsequence and see why it satisfies the given property. Consider

[4,5,9,10]

$S_1 \% M = 4 \% 3 = 1 = 1 \% 3 = 1 \% M$
$S_2 \% M = 5 \% 3 = 2 = 2 \% 3 = 2 \% M$
$S_3 \% M = 9 \% 3 = 0 = 3 \% 3 = 3 \% M$
$S_4 \% M = 10 \% 3 = 1 = 4 \% 3 = 4 \% M$

→ By 3 it's always 1,2,0,1

→ However

#However I was not able to solve the code so comments code (AC) in C++

Time = 0.19 sec

```
#include <bits/stdc++.h>
using namespace std;

int mod = (int) 1e9+7 ;

int main() {
    int t ;
    cin >>t ;
    while (t-->0) {
        int n, k, m ;
        cin >>n>> k >>m ;
        int a[n] ;
        for(int i=0 ; i<n ; i++) {
            cin >>a[i] ;
            a[i]=a[i] %m ;          (The same approach & correctly all already in modular form)
        }

        int b[k+1] ;  //Take first k (+1 element) as 0
        for (int i = 0 ; i<=k ; i++) {
            b[i]=0 ;
        }

        for (int i=n-1 ; i>=0 ; i--) {
            while (a[i] <=k) {
                if (a[i]==0) {
                    a[i]+=m ;
                    continue ;
                }
                if (a[i]==k) {
                    b[a[i]] = (b[a[i]]+1) %mod ;
                }
                else {
                    b[a[i]] = (b[a[i]]+b[a[i]+1]) %mod ;
                }
                a[i]+=m ;
            }
        }
        cout << b[1] <<endl ;
    }
}
```

#I however converted this code in Java from c++ and submitted it. However only input way was to be changed. rest same code it runned fine and (AC) submitted.

Time = 0.70 sec → Since used Scanner. If used Buffer input it can take 0.16 sec.

Code :-

```
P S V M () {
    Scanner sc=new Scanner (System.in);
    int t=sc.nextInt();
    while (t-->0) {
        int n=sc.nextInt(); int k=sc.nextInt(); int m=sc.nextInt();
        int mod = (int)1e9+7;

        int[] a = new int[n];
        for (int i=0; i<n ; i++) {
            a[i] = sc.nextInt();
            a[i] = a[i] % m ;
        }

        int[] b = new int[k+1];
        for (int i=0; i<=k ; i++) {
            b[i]=0;
        }

        for (int i = n-1 ; i>=0 ; i--) {
            while (a[i] <= k) {
                if (a[i] ==0) {
                    a[i]+=m;
                    continue ;
                }
                if (a[i]==k) {
                    b[a[i]]= (b[a[i]]+1) % mod;
                }
                else {
                    b[a[i]] = (b[a[i]] + b[a[a[i]+1]]) % mod;
                }
                a[i]+=m;
            }
        }
        Syso(b[1]);
    }
}
```

Scanned with OKEN Scanner

(5) Now let's ~~he~~ understand how this code is working by debugging it.

⇒ But can't understand this code about how it's working. even by debugging also.

⇒ If you want to check the code in future then check code in github repository.

(MyCodeChefPractise/DCodeChallengeGrumpyGranny)

(#) You have till date (20th April) understood the below stuffs :-

⇒ competitive Programming is all Mathematics and to find pattern within it.

⇒ Almost every question can be solved in O(n) or O(1) if you are able to find required pattern from the given questio

⇒ Directly apply bruteforce only when brute force Time complexity ≤ (8

some examples

⟹

However we've done and seen many. We've

Q.1 Count Substrings :-

Given a string S consisting only 1s and 0s, find the number of substring which start and end both in 1.

substring → sequence of continous character.

e.g for 1101 → substrings are {1} {1} {0} {1}
{11} {10} {01}
{110} {101}
{1101}

e.g

input	output
2 → test case 4 → len 1111 5 10001	10 → all substrings. 3 → {1} {1} {1000 1}

(#) Brute-force solution → O(n²) :- or O(nlogn)

```
P S V M {
    try {
        Scanner sc = new Scanner(System.in)
        int t = sc.nextInt();
        while (t-->0) {
            int len = sc.nextInt();
            String str = sc.next();
            String count = 0;

            for (int num=0; num<len; num++){
                for (int i=0; i<len-num; i++){
                    if(str.charAt(i)=='1' && str.charAt(i+num)=='1'
                        count++
                    }
                }
            }
            Sysо (count);
        } catch (Exception e) {
            return;
        }
    }
```

⇒ Output is correct.

But TLE ⇒ Time = 2.01 sec.

Scanned with OKEN Scanner

② After find the hidden pattern. - O(1)

```
Scanner sc = new Scanner(System.in);
int t = sc.nextInt();
while(t-->0){
    int n = sc.nextInt();
    string input = sc.next();
    long count = 0
    long sum = 0
    for(int i=0; i<n; i++){
        if(input.charAt(i)=='1'){
            count++;
            sum+=count;
        }
    }
    System(sum);
}
```

Also both Ans count or sum as long only otherwise will give wrong Ans.

However constraints $0 < N < 10^5$ even though $sum = N*(N+1)/2$ this will goto 10^{10} too last wrong word.

Time = 0.50 second (AC) All correct.

⇒ Pattern is ⇒

1 0 0 1
↓ ↓
+1 +2 ⇒ Ans = 1+2 = 3

1 1 1 1
↓ ↓ ↓ ↓
+1 +2 +3 +4 → Ans = 1+2+3+4 = 10

OR

$(count * (count+1))/2$

$1+2+\dots+n = \frac{n(n+1)}{2}$

Scanned with OKEN Scanner

#3 Subset, Subsequence and Subarray

⇒ **Subarray** :- ~~Continuous~~ contiguous subsequence

e.g. arr = [1,2,3,4]

Subarrays are ⇒ [1,2,3], [2,3,4] [3,4]
[1], [2], [3], [4], [1,2], [2,3]
etc.

⇒ **Subsequence** :- subsequence contain elements whose subscripts are increasing in the original sequence.

e.g. arr = [1,2,3,4]

subsequence can be ⇒ {1,3}, {1,4}, {2,4}, {1,2,3}, {1}, {2}, etc.

⇒ **subset** :- Contains any possible combinations of original set.

e.g. arr = [1,2,3,4]

subset = $2^{n \rightarrow \text{length of arr}}$ = {1}, {2}, {3}, {4}, {1,2}, {1,3}, {1,4} {1,4}, {2,3}, {2,4}, {3,4}, {1,2,3}, {1,2,4} etc.

* {1,2} ≠ {2,1} ⇒ Then arrangements not considering only not element.

#1 #2 Check whether two ~~grams~~ string are Anagrams :-

⇒ Anagrams → Must contain same numbers of ~~eat~~ characters and same characters in both.

e.g. LISTEN ↔ SILENT
(Anagrams)

ABC = CAB ⇒ Anagrams

AAB ≠ AB ⇒ Not Anagrams.

Approach :- Given char1[]
char2[]

Step① ⇒ Sort char1 & char2

Step② ⇒ Compare arrays. e.g (Arrays.equal(char1,char2)

⑤ Other Approach :- For finding minimum platform require

Time ⇒ O(N) | Bad space ⇒ O(Number of All * 2400 bits)

⇒ Make ~~at~~ each train a fixed size array filled with 1 and 0 where 1 means occupied and 0 means not occupied based on the arrival and departure time. Add all array together and find argmax(sum of arrays) O(N) complexity. No need for sorting or complex logic what so ever. This is most intuitive method.

e.g [000 005 025 0016 006] → Arrival
[010 010 025 0030 008] → respective Dept.

Array of size 2400 at most

[1 1 1 1 1 1 1 1 1 0 0 ... 2400] → 1st train

For second train :-

[0 0 0 0 0 1 1 1 1 1 1 0 ... 2300]

similarly for other trains

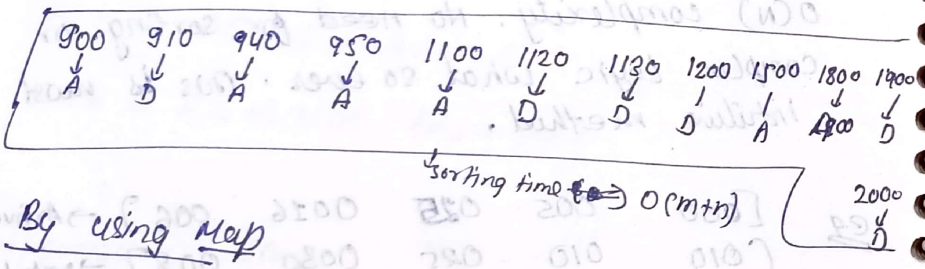
+

Add all these ⇒ we get max value some where.

Just traverse the array from 0-2300 store the max value and

Minimum Platforms needed on railway station = Max^m Platforms needed at any time
= 3.

⇒ This Approach assumes that trains are arriving and departing on the same date.



By using Map

Map each integer of arr with A
& dep with D.

Map <Integer, Character> map

If it's 'A' then add 1 to plat-needed
If it's 'D' then minus 1 to plat-needed.

Ans ⇒ O(m+n)

And max at any point store it.
That will be our answer.

Approach :-

The idea is to consider all events in sorted order. Once the events are in sorted order, trace the number of trains at any time keeping track of trains that have arrived, but not departed.

e.g

arr[] = {900, 940, 950, 1100, 1500, 1800}
dep[] = {910, 1200, 1120, 1130, 1900, 2000}

All events are sorted by time.

Total platforms at any time can be obtained by subtracting total departures from total arrivals by that time.

Time	Event Type	Total Platforms needed at this time
9:00	Arrival	1
910	Departure	0
940	Arrival	1
950	Arrival	2
1100	Arrival	3
1120	Departure	2
1130	Departure	1
1200	Departure	0
1500	Arrival	1
1800	Arrival	2
	Departure	

Scanned with OKEN Scanner

Q1 :- Minimum Number of Platforms Required for a Railway Station.

900 = 9:00

input :- arrival[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}

dept[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}

respective

Efficient Approach ⇒ Time complexity ⇒ O(n) or O(2n) i.e. Time to travel each arr.

⇒ one traversal of both the array is needed.

· space complexity ⇒ O(1)

Some Basic Tricky Interview ⟸ Based Questions

Scanned with OKEN Scanner

#General Knowledge

(1) Different domains of questions in competitive problems. Try to get introduced to all type and solve as many as problem you can to understand it better :-

(5) Domains :-

1. String
2. Arrays (Also included set, Map, vector in it)
3. Stack and Queues
4. LinkedList, Tree and Graph (common traversal questions)
5. Graph Theory
6. Bit manipulation
7. Dynamic Programming
8. Greedy
9. Fenwick Tree, Segment Tree.
10. Convex hull

Scanned with OKEN Scanner

Codechef Understanding

§ What you lack.

① Find difficulty in understanding the question itself.
- Reason 1 → Lack of pratice
- Reason 2 → very few test cases given with no explanation in the answer.

② Give a very good observation in that question and try to find out patterns hidden in it

③ Try and think about every edge cases and hidden testcases in it.

#① Difficulty in Understanding the question

§① From "December Challenge 2020 Division 2"

§ Positive Prefixes :- (4th question 21.58% done)

Problem statement :-

You are given two positive integers N and K, where $K \leq N$.

Find a sequence $A_1, A_2, \ldots, A_N$ such that :

- for each valid i, A_i is either i or $-i$
- there are exactly K values of i such that $1 \leq i \leq N$ and $A_1 + A_2 + \ldots + A_i > 0$

if there are multiple solutions, you may print any one of them. It can be proved that at least one solution always exists.

Input

T
K N

T → No. of test cases
K →
N → as in problem above.

Output

↳ For each test case, print a single line containing N space-separated integers $A_1, A_2, \ldots, A_N$.

Example Input

1
3 3

Example Output

1 2 3

Scanned with OKEN Scanner

Sol:- Now I will explain what question actually mean. (Seen from video editorial in codechef)

⇒ So for given two integeres N and K.

let $N = 5$, $K = 2$ $\quad \{1, -2, 3, -4, -5\}$ ← sequence $\{A_i \dots A_n\}$

$A_i = i$ or $-i$

So $\{1, -2, 3, -4, -5\}$ is one valid sequence for above given condition.

⇒ the 4th term "there are exactly K values of i such that $A_1 + A_2 + \dots + A_i > 0$."

So basically what this lines means that there should be exactly K prefixes from A_1 to A_2 till A_i that there sum to be positive or greater than 0.

⇒ e.g. in $\{1, -2, 3, -4, -5\}$,

sum (1) (1 + (-2) = -1) (1 + (-2) + 3 = 2) (1 + (-2) + 3 + (-4) = -2) (1 + (-2) + 3 + (-4) + (-5) = -7) ← prefixes are as

So these two prefix sum are positive in sequence of N=5.

So K=2 holds in this sequence.

⑤ Solution approach :-

Since now you understand the question.

⇒ Now let's see what approach to be taken to solve this problem.

⇒ Let's take one valid sequence initially in which every alternative element is negative.

i.e. for N=8. $K \le N$

we take ⇒ $\{1, -2, 3, -4, 5, -6, 7, -8\}$

1 (+ve) −1 (−ve) 2 (+ve) −2 (−ve) 3 (+ve) −3 (−ve) 4 (+ve) −4 (−ve)

So we've here K=4 positive prefixes

So we can see a very clear pattern here. In an alternative −ve element taken we've −ve value in it.

So we've found a very constructive sequence.

⇒ So now let's say we've given K=3 ($K \le N$) → Means we w

N = 8 only.

sum: 1, 1 + (−2) = −1, and so

$\{1, -2, 3, -4, 5, -6, 7, -8\}$

(+ve) (−ve) (+ve) (−ve) (+ve) (−ve) (+ve) (−ve)

1 2 3 upto here only we get K=3. our sequen. i.e. 3 positive prefixes. −6 7 to −7 convert −8

So now start from back and make all negative to keep K=3 only

So our answer will be $\{1, -2, 3, -4, 5, -6, -7, -8\}$ for K=3 & N=8.

Scanned with OKEN Scanner

also rem! there are $\lceil \frac{n}{2} \rceil$ ceiling value number of positive prefixes for N element.

⇒ If K=6,

Just start changing from back only in the sequence so that the beginning positive sequence prefixes has no change in left.

So

{1, -2, +3, -4, 5, -6, 7, -8}

we've $\lceil \frac{N}{2} \rceil = \lceil \frac{8}{2} \rceil = 4$ positive prefixes already.

But K=6 so we need two more positive prefixes. so start from right end of sequence and make a two of the from -ve to positive.

so our final ans will be

{1, -2, 3, -4, 5, -6, 7, -8}

1 ✓ 2 ✓ 3 ✓ Make +ve (-6 to +6) ← OK ← Make +ve (-8 to +8)

4 ✓ 5 ✓ 6 ✓

So now we've made this sequence having K=6 positive prefixes. hence final sequence is

{1, -2, 3, -4, 5, 6, 7, 8} Ans

Scanned with OKEN Scanner

#3 Longest Substring Without Repeating Characters :- (Medium level)

Question ka i.e. testcode only

Sample testcase 1 :-

"abcabcbb" → output 3 (abc)

Test Case 2 :- "bbbb" → output 1 (b)

Test Case 3 :- "pwwkew" → "wke" → output 3

Test Case 4 :- s = "" → output 0.
s = " " (space) → output = 1
→ edge cases

My Approach → "To use (set) only."
Not working. use 2 pointer method, it will be more better.

Constraints ⇒ $0 <= s.length <= 5 * 10^4$

s = english letter, digits, symbols & spaces.

Test Cases :- ⇒ "abc" → all different → output = 3.

"dvdf" → output 3
(My set is

My Approach :-

→ To use set only to find Longest substring.

→ Approach step :-

→ Handle edge cases for when string.length = 0 or 1 then return the same.

→ Take two pointer i & j and add element in a set. initially ~~i=j~~ & j=i & we add arr[j++] element in the set unless we find any duplicate element in it. Store count = j - i and update 'max' and then empty the set. And then update or increment i by 1 until the whole string length.

→ ~~After coming out of the loop make sure~~

My Code (which got accepted)

(But this two pointer approach was not quite efficient enough)
since we're emptying the set. Instead we could have removed only i th element & then i++.
So it will be sliding window technique

→ Code as :-

O(n²) runtime | O(n²) Memory

```
public static int lengthOfLongestSubstring(String s) {
    Set<Character> set = new HashSet<>();
    int count = 0;
    int max = 0;
    if (s.length() == 0){
        return 0;
    }
    if (s.length() == 1){
        return 1;
    }

    int j=0;
    for (int i=0; i<s.length(); i++){
        j= i
        while (set.add(s.charAt(j)) && j<=s.length()){
            if (j<s.length()){
                j++
            }
            if (j==s.length()){
                break;
            }
        }
    count =j-i;
    if (max < count){
        max = count;
    }
    set.clear()
}
return max
```

Scanned with OKEN Scanner

(5) Here we could have made our code more efficient, if instead of using two pointer only we've also added sliding window technique in it.

⇒ Because every time we add duplicate element arr[j] in set, we instead of emptying the set and adding again from 'i=i+1' upto ~~we~~ j++ (until duplicate); we could have removed 'i' arr[i] element and then we will have ~~i=i+1~~ i+1 to j^th element in it already and only keep increasing j only to add new element in set. In this way we can ~~do~~ make our ~~for~~ code more efficient and memory saving using this sliding window technique.

(5) Code for sliding window technique is as follows:-

O(n) runtime O(n) memory

```
public int lengthOfLongestSubstring(String s) {
    int a_pointer = 0;   // Pointer to start of the window. (i
    int b_pointer = 0;   // Pointer to end of the window we are usi
    int max = 0;

    ~~int~~
    HashSet<Character> hash_set = new HashSet<>();

    while(b_pointer < s.length()) {
        if(!hash_set.contains(s.charAt(b_pointer))) {
            hash_set.Add(s.charAt(b_pointer));
            b_pointer++;
            max = Math.max(hash_set.Size(), max);
        }
        else {
            hash_set.remove(s.charAt(a_pointer));
            a_pointer++;
        }
    }
    return max;
}
```

Scanned with OKEN Scanner

(4#) (4) Median of Two Sorted Arrays (Hard)

See it in leetcode only

(1) Problem statement -

→ Given two sorted arrays. arr1, arr2, size m & n.

→ Return the median of the two sorted arrays.

⇒ (Median. is just middle element as we know) (in sorted list).

⇒ overall complexity ist. says is O(log (m+n)).

(2) Test Cases :-

① arr1 [1,3] arr2 [2] → merged array = [1,2,3]
→ median = 2.00000 (output)
Return in double so.

② arr1 [1,2] arr2 [3,4] → merged array = (1,2,3,4)
median = $\frac{2+3}{2}$ → 2.5000 output.

③ [0,0] [0,0] → Output 0.00000

④ [ ] [1] → output → 1.00000
empty

⑤ [2] [ ] → output → 2.00000

Scanned with OKEN Scanner

My Approach :-

- Identify median from each array and then later add both & then and divide by 2 will be then answer.
  (Just find the middle element for odd size)
  (2 middle element for even size & then divide by 2).
- Edge cases
  - If one of the arr is empty then discard that array.

- In test case 1 :-

[1,3] → $\frac{1+3}{2} = (2)$

[2] → $(\frac{2}{2}) \rightarrow 2$

$\frac{2+2}{2} = 2$ Ans so is the expected output.

[1,3] [2,7] → Expected output = (1 2 3 7) → $\frac{2+3}{2} = 2.5$ expected

mine's Approach output

$\frac{1+3}{2} = 2$

$\frac{2+7}{2} = 4.5$

$\frac{4+5+2}{2} = \frac{6.5}{2} = 3.25$ Ans → my output

Wrong it is wrong

So we need to see other approach

⇒ More testCases :-

① [1, 9] [100, 101, 102]

merge

[1, 9, 100, 101, 102]

output = 100.0000

②

[101 102 103] [100 110 120]

merge

[100 101 102 103 110 120]

$\frac{102+103}{2} = 102.5$ output

Out of Leetcode (A very good 'stack' based question, learnt for tuevo hiring)

Next Greater Element :-

Q. Given an aveay, peint the Next Greatee Element (NGE) foe eveey element.

The Next greatee element for an element 'x' is the fiest greatee element on the eight side of 'x' in the aveay. Elements foe which no greatee element exist, considee the next greatee element as -1.

e.g. arr [13, 7, 6, 12]

NGE → -1 12 12 -1

Approach :- foe O(n), we need to use stack

↳ Approach :-

Approach :-

→ Push the fiest element to stack

→ Pick rest of the elements one by one and follow the following steps in loop.

1. Mawk the current element as 'next'
2. If stack is not empty, compaee top element of stack with 'next'
3. If 'next' is geeatee than the top element feom stack. 'next' is the next geeatee element foe the popped element.
4. Keep popping feom the stack while the popped element is smallee than 'next'. 'next' becomes the next geeatee element foe all such popped elements.

→ Finally, push the next in the stack

→ After the loop in step 2 is ovee, pop all the elements feom the stack and peint -1 as the next element foe them :-

Scanned with OKEN Scanner

Given 'arr[]' of length 'n' code will be as below

Java

```
Stack s = new Stack()

Stack<Integer> s = new Stack<Integer>();
int element, next;
int elementIndex;
s.push(arr[0]); // Push the first element to stack
s.push(0)
// Iterate for rest of the elements
for (int i=1; i<n; i++) {
    next = arr[i];
    if (!s.isEmpty()) { // If stack is not empty, then pop an element
                        // from stack
        element = s.pop();
        elementIndex = s.pop();
        element = arr[elementIndex];
        // If the popped 'element' is small than 'next', then
        // (a) print the pair (b) keep popping while elements are
        // smaller & stack is not empty.
        while (element < next) {
            Sys.o(element + "-->NGE" + next);
            if (s.isEmpty() == true) {
                break;
            }
            element = s.pop();
            elementIndex = s.pop();
            element = arr[elementIndex];
        }
        // If element is greater than next, then push the element back
        if (element > next) {
            s.push(element);
            s.push(elementIndex);
        }
    }
    // Push next to stack so that we can find next greater for it.
    s.push(next);
    s.push(i)
}
```

mark it as comment.

```
// After iterating over the loop, the remaining elements in stack
// do not have the next greater element, so print -1 for them
while (s.isEmpty() == false) {
    element = s.pop();
    next = -1;
    arr[elementIndex] = -1
    Sys.o(element + "-->NGE" + next);
}
```

→ This will then not used to retrieved - Just print the o/p. that is the arr.

For input [1, 2, 3, 4, 5, 4, 3, 2, 1]

output

1 -->NGE	2
2 -->NGE	3
3 -->NGE	4
4 -->NGE	5
1 -->NGE	-1
2 -->NGE	-1
3 -->NGE	-1
4 -->NGE	-1
5 -->NGE	-1

→ But to print in array order & sequence only, instead of pushing & pop. arr elements push arr index.

Scanned with OKEN Scanner

Understanding C++ Codes

1. You can write scanf() printf() along with cout<< and cin>> in same C++ program.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int a;
    scanf("%d",&a);
    printf("%d",a);
    cout<<a;
    return 0;
}
```

Input:- 5

output:- 5
5

2. Basic Data Types :-

int ("%d") | long ("%ld") | char ("%c") | float ("%f")

Double ("%lf"); | string ("%s")

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a;
    long b;
    char c;
    float d;
    double e;
    cin>>a>>b>>c>>d>>e;
    printf("%d\n%ld\n%c\n%f\n%lf",a,b,c,d,e);
    return 0;
}
```

printf("%d\n%c",a,c); → output 123 c

printf("%d%c",a,c); → output : 123c

printf("%d %c",a,c); → output :- 123 c

cout<<a<<endl<<c; → output :- 123 c

cout<<a<<" "<<c; → output :- 123 c