মেট্রোপলিটন ইউনিভার্সিটি
Metropolitan University

Editorial of
Metropolitan University Tech Fest 2017 Programming Contest (Senior)

A - Help Ribo

Problem Setter : Bishwajit Purkaystha (CSE 12 SUST)
Editorial by : Nafiul Adnan Chowdhury

There are many approaches to solve this problem. One of these is-
use a structure or pair array/vector to store the start and finish time. Just sort it.
Take a variable last = 0.
Now iterate through the array, if i'th finish time is greater than last: change the last, cnt++;
else: continue.

B - How many ways

Problem Setter : Tahmid Hossain Olee (CSE 12 SUST)
Editorial by : Nafiul Adnan Chowdhury

Take a map to store and count the array element. Check the numbers within k/2. If we can make k-arr[i] the ans will be increased as frequency of arr[i] multiplied by frequency of k- arr[i]. And try to figure out the increasing when k-arr[i] == arr[i].

Oww! You've noticed that, elements can be negative, right?

Wait! You don't need to do that much work for this problem as n<=10^4 :D you can just do it in (n*n) by two nested loop i and j. And check if(arr[i] + arr[j] == k) :D

C - Horrible Queries

Problem Setter : Maruf (CSE 15 SUST)
Editorial by : Nafiul Adnan Chowdhury

Sure it is segment tree or Moo's right? ;) Nope Bro, you don't have to think it hard ;) you can solve it with Moo. But there is an easy solution.

Take a 2D matrix, where we will store each numbers frequency from (1-50) at every index (1 to N) it'll look like arr[55][N+5] . After scanning each value with a loop (i= 1 to N), we will increase arr[val][i] by 1 And do a cumulative sum for (k in 1 to 50) : arr[k][i] += arr[k][i-1].

And for every query you can just run a loop from [1 to 50] for the L and R and have the counts of every elements. for(k = 1 to 50): cnt = arr[k][R] – arr[k][L-1] , if the cnt >= k increase the ans by 1 ;)

Complexity? To count the frequencies : 10^5 * 50 and for each query it's a 50 just ;)

D - Matchstick Man

Problem Setter : Arnab (CSE 13 SUST)
Editorial by : Foysol Ahmed Shuvo

At first, sort the given array in increasing order. Iterate through 1 to N and check for every position: if sum of every elements before the i'th element (current position) is greater than array[i], you update your result with the index [res = i].
Finally, you check if the result is less than 3, output will be Impossible, otherwise print the result.

E - Playing with Digits

Problem Setter : Tahmid Hossain Olee (CSE 12 SUST)
Editorial by : Priojeet Dash Priyom

It's a Digit Dp problem.
Count each digit(0,9) in range 1 to (a-1) and in range 1 to b and store them in two arrays say aa and bb.
for every i from 0 to 9, store the difference of bb[i] and aa[i] in a separate array named temp. Use modular arithmetic to handle case of aa[i]>bb[i].

now cnt is the summation of
all elements of temp and sum is the summation of all temp[i]*i (i->0,9)

AND! A can be greater than B. ;)

F - Game of Sticks

Problem Setter : Moudud khan Shahriar (CSE 13 SUST)
Editorial by : Nafiul Adnan Chowdhury

It's a very Basic probability problem. Multiply all the probability, subtract it from 1, you got the answer.

G - Bubble Shooter

Problem Setter : Nirob (CSE 12 SUST)
Editorial by : Foysol Ahmed Shuvo

You know you've to take input in a 2D matrix, right? ;) for each row, iterate from left to right, cnt how much obstacle you found through it. If obstacle == max_obstacle, break and Count how much bubble you got before, and do the same from right to left, push the max(left_bubble,right_bubble) in a vector. Now try to maximize the ans from the vector for M bullets.

Wait! M can be Bigger than N.

H - Functions are Beautiful

Problem Setter : Mridul (CSE 13 SUST)
Editorial by : Nafiul Adnan Chowdhury

Too hard problem ;) I'm giving you the solution. Figure it out please:
F[1] = 1, F[2] = 1.
For(i=3; i<=1000000; i++) F[i] = (F[i-1] + F[i-2]) %1000000007;

Now for each input n, answer is F[n].

I - A Weird Dragon

Problem Setter : Nirob (CSE 12 SUST)
Editorial by : Majharul Islam Rafat

It is a basic geometry problem which has 3 cases.

1. If the length of rope is less then min(L,W) then the area is simply the area of the circle which is pi*r*r where pi=acos(-1).

2. If the length of rope is greater than diagonal length of the rectangle then it can cover the whole area of this rectangle so in that case the area which Dragon can cover is the area of the rectangle(L*W)

3. The last case is the tricky part. What will happen if some portion of circle goes outside of the rectangle? We have to cut it out from the area of the circle.

We know the height of the segment in that case. Which is R-L if L is minimum. and to calculate the area of that segment we can apply the default formula which is >>area=(R^2*acos((R-H)/R)-((R-H)*sqrt((2*R*H)-(H*H))); where H=Height Of The Segment; if the extra portion exceeds both length and width just calculate for the other

one and you'll get the answer which is Area of the circle-extra portions which is outside of the rectangle.

Thank You

From

Department of CSE, Metropolitan University, Sylhet.