ACM ICPC Regional, Kolkata Kanpur Site, 2017 Editorial

A. Appearance Count

Number of occurrence of a string in all possible string of length **n** doesn't depend on the content of the string. It only depends on the length of the string. Any string of length **m** appears in all possible strings of length **n** exactly $\mathbf{(n-m+1)*26^{n-m}}$ times.

Setter: F A Rezaur Rahman, PhD, Michigan State University

B. Get The Gold

For this problem, we need to calculate the normal on the forest carefully. Cross multiplication of vector **SA** and **SB** will produce a normal to the plane. However, as the origin must be above the surface, this normal should be on the same side of the plane as the origin.

Let's say **S' = S+n,** where **n** is a normal. If the dot product between vector **SS'** and **SO** is negative, just change the direction of **n** by multiplying it with -1. We will get the desired normal. After getting the correct normal vector, we just need to use cross multiplication carefully to reach to the gold.

Setter: Dr. Md. Kaykobad, Bangladesh University of Engineering and Technology

C. Club of Riders

We can use two pointer here. First of all, sort the two arrays (guards' skills and scooters' performance arrays). Iterate over one array in ascending order and the other array in descending order. We can easily find out the possible number of scooters for each of the guards. Some basic combinatorics will be enough to find out the total number of valid assignments.

Setter: Sheikh Shakib Ahmed, CISCO

D. Triangle Count

The answer is 1 + 2 + 3 + … + (L-K+1). That's it, plain as simple. Use the summation formula if you wish.

Setter: Kazi Hasan Zubaer, Kona Software Lab

E. Password Riddle

Try to find a way to match a segment with the consecutive occurrence of a digit. Let's say, a digit occurs 42 times. Prime factorization of 42 is $2^2 \times 7$. What does it tell you? There must an odd or seven in the riddle. There can't be a three or five in the riddle. There must be at least one even or four in the riddle. There can't be three or more evens. Carefully handle all the cases. Alternative approach might be searching and pruning. With some optimization, this works. Once you get the way to match a riddle with a password, you can run DP to store upto which segment of the riddle matches with upto which portion of the password.

Setter: Kazi Hasan Zubaer, Kona Software Lab

F. Huge Number of Trees

This is a dynamic programming problem. We can keep track of the number ways to make a tree from a certain level when we have some number of nodes, n, the previous level had k nodes, and the minimum degree requirement is d (dp[n][k][d]). We need to use a variation of Stirling number of the second kind to assign the n nodes under k parents where each parent gets at least a certain number of children.

Setter: Anindya Das, IWOA State University

G. Palace of King

We can a iterate over the possible length and width of the palace in O(max(M,N)) (a two pointer approach). Now, to check if a palace of certain length and width is feasible we need some intuitions and swipe line technique.

Let's say we want to know the cost of a palace to be built with lower left corner at (x,y), length l, and width w. We can check for every land if the intersect with the rectangle with lower left corner at (x,y) and upper right corner at (x+l, y+w) **OR** we can extend each of the lands' lower left corner to the left (l units) and down (w units) and check how many of them covers point (x,y). This is the cost of that point. We will go for the second approach. After updating all the rectangles (in O(L) time) we will look for the point with lowest cost. We will run a swipe line here from left to right (after updating the rectangle) and maintain a segment tree. Whenever we enter a rectangle, we will update the range with the cost of that land. While leaving the rectangle, we will subtract the cost. We will look for the lowest value always. If the lowest value is less than or equal to C, the length and width we started with is feasible.

Setter: Kazi Hasan Zubaer, Kona Software Lab

H. Unpredictable Array

Setter: Repon Kumar Roy, MSc, Bangladesh University of Engineering and Technology

Solution Sketch:
1. First calculate the initial value of the array.
2. Now for each update x->y, the occurance of x is replaced by y. So for each occurance of x, we subtract the contribution of x and add the contribution of newly formed y.

Explanation:
Part 1: calculate the value iterating trough the whole array. This is the initial value of the array.
Part 2:

Let's solve a small subproblem.
update are the same. x->y denoting x will be replaced by y.
but now we have queries like tell me the value stored in index i. How to solve this?

A brute force approach will be to update the whole array, (atleast the occurance of x) and for each query, we can give result in O(1). Updating the array can take O(n) in the worst case. So complexity is O(q*n)

We have another approach which uses input buffering where update is O(1) and query is O(sqrt(q)).

Here is another approach. At any time, the array can be divided into several equivalence classes containing the same number. Each step merges at most one pair of such classes.

eq(x) -> denotes the equivalence class for x
pos(x) -> denotes the occurance of x.
B(i) -> denotes the membership of the value at position i.

identity of an equivalence class eq(x) is x.

Each eqivalence class has a vector of occurance and a pointer. The value pointed by the pointer denotes the identity.
Now for each update x->y, we always merge the smaller to larger equivalence class.

if eq(x) is smaller than eq(y) then, for each occurance occ in pos(x) , we update B(occ) to y, and push them in pos(y)

else, for each occurance occ in pos(y) , we update B(occ) to x, and push them in pos(x)
and the value pointed by the pointer of eq(x) is updated to y.

So, the total operation in updates will be O(nlog n) and each query can be answered in O(1) through B(i) and pointer of the equivalence class.

Back to main problem

adj(x) -> denotes the vector of elements adjacent to x.

The value of array is defined as the sum of absolute difference of adjancent element in the array. So, let's store the values adjacent to x in adj(x) for each x.

So the contribution of x can easily counted by iterating adj(x). Now if x is replaced by y, can we calculate the contribution of this elements for y quickly?

To quickly answer this query, we need to store the elements of adj(x) in Binary Indexed Tree(BIT).
BIT should answer this type of queries:
i) How many numbers is <= z
ii) Summation of numbers which value is <= z

In this BIT, we can calculate the contribution of newly formed y easily.

Now, if we want store adj(x) in BIT for each x, it will take 10^5 * 10^5 memory which is infeasible.

let's call an equivalence class with size >= C is large. For each such class, we store a BIT tree denoting the frequency of each integer adjacent to each integer in the class. Now, to calculate the contribution of x,

if eq(x) is large, then get answer from BIT.
else iterate through the position and calculate contribution. This is possible because after each update we can query the value of an index.

Whole Procedure:

for each update x->y, we always merge the smaller equivalence class to larger equivalence class. First subtract the contribution of x, add the contribution of y, update the equivalence classes. Now for each large equivalence class, update their BIT so that all the occurance of x is replaced by y.
During any update , if the size of equivalence class >= C, we create a bit for that class.

I. Secrets

This is a BFS problem. The total number of states here is 10! in plain sight. However, we can consider the envelope of a person and the envelope of his/her best as identical ones. This observation reduces the number of highest possible states to 10!/(2^5) which is O(10^5).

Setter: Kazi Hasan Zubaer, Kona Software Lab

J. Reached Safely or Not

This is probably the easiest one. Just keep track of the position and check the end position if it's dangerous, or relative's home, or something else.

Setter: Kazi Hasan Zubaer, Kona Software Lab