A - Have Some Water

Author: Iftekhar Hakim Kaowsar
Alter: Safayet, Mridul
Topic: NT, Sparse Table, DP, Adhoc

Solution Idea: First of all, the optimal answer must not exceed the sum of the array (if all parts have length 1). Let dp[L] be the answer for the A[L…N]. You need to track only those R where LCM for subarray starting from L changes. How many LCM-changing positions of R do we need? Certainly, not more than $\log_2 \sum A_i$. The rest is just calculating the dp.

Code: Have Some Water

B - Break The Bar

Author: Ahasan Kabir
Alter: Nirjhor, Mridul
Topic: Adhoc

Solution Idea: The answer is either -1 or P-1 or P or P+1. It's always optimal to break one side from the border horizontally or vertically. When P is equal to N*M , the answer is P-1. When P < min(n,m) the answer is surely P+1. As constraint of P is small, you can loop through min(n,m) when P > min(n,m). Notice that each move increase the number of pieces by one. That means when answer is P, total number of pieces will be P+1 and there will be only one unused & unbroken piece. If it's not possible to break the bar within P moves than answer will be P+1.

Code: Break The Bar

C - Three Arrays

Author: Safayet Hossain Masum
Preparation: Nirjhor
Alter: Ishtiaq
Topic: Greedy

Solution Idea:
The Bitwise AND value of B will always be [0,1] and B will always be the lexicographically smallest or the second smallest possible such B.

Let’s disregard condition 3 and try to build the lexicographically smallest B. We can do it greedily starting at the end, at each index we try to minimize the B/ maximize C keeping it consistent with the indexes previously processed.
Notice B[n] will always be 1, which means the AND of the lex smallest B will be in range [0,1]. If it is 0, we’ve already minimized the bitwise AND. If not, check whether there exists any other

valid B, if not that is the answer. If exists, find the second lexicographically smallest B.

Code 1: Code 1
Code 2: Binary Search

D - Max SMEX

Author: Safayet Hossain Masum
Alter: Iftekhar, Mridul
Topic: DP, Window Sliding/2Pointers

Solution Idea:
Let c be the distinct number of characters in the string. The expected complexity is O(n*c*c). There are several ways to achieve similar complexity. We will discuss the author's solution here. See that the answer will be at most 26. There are only n*26*26 substrings that might contain the Max SMEX. If we fix the start index of a substring, see that the indexes where the frequency of character at that index <26 is only important to us, thus n*c*c. We can just calculate SMEX for those.
Now, if you look at the important end indexes for substring starting at i and i+1, see that they will differ by at most 1 element. And amongst the common elements, only the frequency of character at index i changes. So we can easily calculate the important indexes and their frequencies in O(c*c).
Now to calculate the SMEX of those important substrings, see that we can also use smex calculated at the previous step as only the frequency of one character changes.

Code: Max SMEX

E - Missing Integer

Author: Safayet Hossain Masum
Alter: Apurba, Iftekhar
Topic: Math

Solution Idea:
Let's say you know that A has i bits. If you divide A by 2^i, A will become 1. After that, you can do a binary search.
To find the number of bits in A, see that,
for any integer B <= A if you do A\` = A*(B+1)/B, A' > A
for any integer B > A if you do A\` = A*(B+1)/B, A' = A.
Check this for decreasing B = 2^i. For the first i, we get A\`>A, is the number of bits in A.

Now make A\`= A again, to do that, notice that after the last operation A\` will always be A+1.
Now we can divide A by 2^i and do the binary search.

How many operations does it take?
Let x be the MSB of A. We will need (29-x)*2 operations to find the number of bits. 2 operations to reduce A` by 1, 1 division operation to make A`=1.
*After that, the binary search will take x**2 steps.
In total: 61 steps.[<64] The author's solution takes 59 in the worst case.

Code: Missing Integer

F - Solo Leveling

Author: Ishtiaq Islam
Alter: Ahasan
Topic: Implementation

Solution Idea: Suppose the player can defeat maximum ***X*** number of monsters. Here, amulets will be used to defeat the ***A*** monsters with the largest hit points and the remaining ***(X-A)*** monsters will be defeated following the first and second ways described in the problem statement. To find ***A*** number of monsters having the largest hit points, you can use priority queue.

Code: Solo Leveling

G - Demon Slayer

Author: Ishtiaq Islam
Alter: Reyad
Topic: Basic Geometry

Solution Idea: Let's choose each pair of points from ***M*** points and form a line, then count the maximum number of points which are on the line. Now for each i $(1 \leq i \leq N)$, print the maximum number of points on a line going through $(i, 0)$. Time complexity is $O(N + M^3)$. Also it can be solved in $O(N + M^2 logM)$ after some calculation!

Code: Demon Slayer

H - Turn Them Off

Author: Ahasan Kabir
Alter: Apurba, Nirjhor
Topic: Data Structure, Math

Solution Idea: It's always possible to turn of all lights using magic steps. Let there is P number ones currently and POS_i is the index of those one's. Then total number of one's needed would be ,

$$2 \times \sum_{i=1}^{P} POS_i - P^2$$

Try to derive this equation by thinking at which order the lights are turned off. Find the number of moves needed to turn off each light separately. Then simplify your equation. Finally to solve the problem you can maintain the sum of positions of ones and total number ones in a segmen tree with lazy propagation.

Code: Turn Them Off

I - Mexy

Author: Md Reyad Hossain
Alter: Apurba, Nirjhor
Topic: DSU on Tree, Data Structures
Solution Idea: I - Mexy Editorial
Author code: https://paste.ubuntu.com/p/sCCgmnSb2b/
Alter Code: Mexy

J - Burden Of Expectation

Author: Apurba Saha
Alter: Nirjhor
Topic: Linerarity of Expectation, DP

Solution Idea: The first observation is that the final expected value is:

$$\sum_{i=1}^{k}\sum_{j=1}^{k} P(i,j)$$

Where $P(i,j)$ is the probability of i-th token in a broken node and j-th token in a non-broken node after q operations.I
Next we need to look at the operations. The operation is now either we select the i-th token, or we select the j-th token or any other token. We can iterate the number of times i-th token will be selected. Then the maximum number of time j-th token will be selected is fixed. This part can be precalculated with a DP.
Final Complexity: $O(k*k*q + q*q*n + q*(n+m))$
Code: Burden Of Expectation

K - Not a GiveAway

Author: Jubayer Nirjhor
Topic: Implementation
Solution Idea: The square is bigger only when n is 3.
Code: Not a Giveaway