2026 Wuhan Invitational Editorial

Renmin University of China

June 7, 2026

Acknowledgements

We thank every problem setter and tester for their hard work.

We thank the Wuhan University training team for their guidance on the problems.

We thank Liu Jiameng and Xing Sutong for their irreplaceable contributions during problem preparation.

We thank Qingyu for providing support through the judging platform.

Without them, this contest would not have been possible!

Notice

This contest will be uploaded to the Universal Cup later.
Please do not share the statements or editorial externally.

Judges' Expected Difficulty

Very Easy: C
Easy: B, H
Easy-Medium: D, I
Medium: F, J
Medium-Hard: A, K, L
Hard: E, G, M

Statement

Given a simplified game state from Slay the Spire, determine whether it is possible to win.

$n \leq 100$.

Author: Cocoly1990, Prepared by Milmon

Solution

Simulate according to the statement.

Note that if the monster can be killed directly in some round, then there is no need to play block in that round. The samples hint at this point.

Time complexity: $\mathcal{O}(n)$.

Statement

There are m operations and a sequence of length n. Each operation replaces every element by its mex with a given number, or by its gcd with a given number. Determine whether there exists a way to fix all undetermined operations so that the final results of all elements in the sequence are equal.

$n, m \leq 3 \times 10^5$.

Author: irris & Cocoly1990, Prepared by Milmon

Solution

Once all numbers become equal at some time, they will remain equal forever.

Since $x_i \geq 1$, whenever $a_i \neq 0$, a mex operation can turn all numbers into 0.

Note that after applying a gcd operation, all numbers become nonzero.

Therefore, if there is a gcd operation before a mex operation, the answer must be Yes.

Solution

Also, if there are two positions with $c_i = 0$, then assigning them to gcd and mex respectively can make all numbers equal.

Now we only need to consider the case where there is at most one $c_i = 0$. Enumerate the choice for this c_i, and check whether there is a gcd operation before a mex operation.

In all remaining cases, the operations must be in the form of a prefix of mex operations followed by gcd operations.

It is easy to maintain the composition of multiple mex and gcd operations on the same number, so we can directly maintain the resulting value after these operations.

Time complexity: $\mathcal{O}(n \log V + m)$.

Statement

Given a rectangle, each operation makes one horizontal cut or one vertical cut. After each operation, find the current maximum subrectangle area.

$n, m \leq 10^9$, $q \leq 5 \times 10^5$.

Author: yemuzhe, Prepared by yemuzhe

Solution

The area of a rectangle is its length times its width, and the two dimensions are independent in this problem.

Therefore, the current maximum subrectangle area is the largest segment length on the x-axis multiplied by the largest segment length on the y-axis.

Use sets to maintain the cut points and maximum gap in each dimension.

Time complexity: $\mathcal{O}(n \log n)$.

Statement

Given a number n, two players take turns operating on it according to the rules until $n = 1$. The first player wants to maximize the number of operations, while the second player wants to minimize it. Assuming both players are perfectly rational, find the final number of operations.

$n \leq 10^{18}$.

Author: yemuzhe, Prepared by yemuzhe

Solution

Lemma: $\text{ans}_n \leq \text{ans}_{n+1}$.

The proof is easy. Use induction and consider the case of n. If the first player changes it to a number not equal to n, then the case of $n+1$ can also reach that number.

Otherwise, the case of $n+1$ can move to $2n$, and the second player's decision is strictly better than in the case of n.

Therefore, the second player will always divide by the largest prime factor of the current number.

Solution

A trivial strategy is for the first player to change the number each time into the largest reachable power of 2.

Consider the number obtained after this trivial strategy followed by the second player's operation; it must be at least $\frac{n}{2}$.

Suppose the second player chooses the prime p in some move. Then the first player should guarantee $\frac{2n}{p} \geq \frac{n}{2}$, which means $p \leq 4$.

Therefore, the prime chosen by the second player should only be 2 or 3.

Preprocess all numbers whose prime factors are only 2 and 3, then simulate the game process.

Time complexity: $\mathcal{O}(\log^2 n)$.

Statement

This is a communication problem. The first player is given a tree with n vertices. Each operation may flip the color of one vertex and transmit the current number of bichromatic edges. The second player is required to construct an isomorphic tree from this information. At most $n-3$ operations may be used.

$n \leq 3 \times 10^5$.

Author: Cocoly1990, Prepared by Milmon & Celebrate

Solution

If vertices are flipped in a certain order, then the transmitted information is equivalent to the degree of the current vertex.

Simulate a dfs/bfs process. It is easy to do this in n operations.

Without loss of generality, suppose we simulate dfs. Consider the last three vertices. If they are not all attached under the same vertex, then the case is trivial.

Otherwise, in fact one operation can easily distinguish them. Thus, we reduce the number of operations to $n-2$.

Solution

One approach is to strategically choose a root so that the remaining 4 vertices can be trivially distinguished with one operation.

Alternatively, one can make a more detailed case analysis on the remaining 4 vertices, possibly with several conventions at the beginning for special cases, and still distinguish them with one operation.

An easier method is to choose a leaf as the root, which saves one operation at the beginning.

Any of the above methods can achieve $n-3$ operations. Combining the two methods can achieve $n-4$. Due to the intended difficulty of the problem, the official contest used the $n-3$ version.

Time complexity: $\mathcal{O}(n)$.

Statement

Given the values of all n prizes in a lottery, one may reroll at cost c. After each reroll, the cost increases by k. Find the expected prize value minus reroll costs under the optimal strategy.

$n, c, k, A_i \leq 4 \times 10^5$.

Author: yemuzhe, Prepared by: yemuzhe

Solution

Observe that if $k > 0$, then the number of lottery rounds will not exceed A_{max}.

Let f_i be the expected profit from lottery round i onward. Then $f_{A_{\text{max}}+1} = 0$, and

$$f_i = \frac{1}{n} \sum_{j=1}^{n} \max \left\{ A_j, f_{i+1} - \text{cost}(i+1) \right\}$$

Since f is monotone, after sorting A, the indices j for which the max takes the A_j term form a prefix.

The split point can be maintained with either binary search or two pointers.

Solution

Next consider the case $k = 0$. This case is straightforward: the strategy will not change with the round number.

In other words, we set a threshold x. If we draw a prize with value $> x$, we stop; otherwise, we continue. Enumerate this threshold and directly compute the expectation.

Time complexity: $\mathcal{O}(A_{max} + n \log n)$.

Statement

Given the values and play times of n Snake Bite cards, each enhancement may choose one Snake Bite card and increase its value by 1. Find the maximum total final damage after using at most k enhancements.

$n \leq 3 \times 10^5$, $k \leq 10^9$.

Author: Coffee_zzz, Prepared by Milmon

Solution

We directly state the conclusion: all enhancements will be given to a single card.

This can be proved without difficulty by an exchange argument.

Solution

Enumerate which card is enhanced. We can view the poison at each moment as two parts: the original part and the part caused by the enhancements.

Whenever the poison amount decreases, assume without loss of generality that the decrease is taken from the original poison first.

Then the enhanced poison has the form of a combination of plateaus and decreasing segments, and the decreasing positions only depend on the original poison.

Enumerate the enhanced card from back to front. Since the total enhancement amount is the same, use a queue to maintain consecutive segments where plateaus and decreases alternate.

Time complexity: $O(n)$.

Statement

Given a permutation of length n, each operation uniformly randomly chooses a prefix and sorts that prefix and the remaining suffix separately. Find the expected number of operations until the permutation becomes fully sorted.

$n \leq 500$.

Author: Cocoly1990 & Milmon, Prepared by Milmon

Solution

After one operation, the sequence can always be represented as two sorted segments. In other words, we can describe the current state with a 01 string, where 0 means that the number is in the first segment.

Each later operation is equivalent to changing several leading 1s into 0s, or changing several trailing 0s into 1s. The sequence is fully sorted if and only if all 0s are before all 1s.

Record the positions of the last 0 and the first 1, then optimize the DP with prefix sums.

Time complexity: $\mathcal{O}(n^3)$.

Bonus: solve this problem in $\mathcal{O}(n^2)$.

Statement

Given a sequence S of length n, each operation may choose two equal positions i, j, delete the positions in $[i+1, j]$, and gain the corresponding score. You may copy the sequence any number of times, so that under the final optimal strategy the remaining score is as small as possible. Find the minimum remaining score and, under this condition, the minimum number of copies.

$n, S_i, a_i \leq 3 \times 10^5$.

Author: yemuzhe & Cocoly1990, Prepared by yemuzhe & Cocoly1990 & Milmon

Solution

Consider a graph-theoretic model. For each $S_i \neq S_{i+1}$, add an edge $S_i \to S_{i+1}$ with weight a_{i+1}. We want the shortest path from S_1 to S_n to be the answer.

By analyzing the meaning of this model, it is not hard to see that the number of edges on the shortest path is exactly the answer to the second question.

Since the number of vertices in the graph equals the alphabet size, the number of edges on the shortest path will not exceed $m = |\{S_i\}|$.

Time complexity: $\mathcal{O}(n \log m)$.

Statement

Given n strings, find the sum of the numbers of borders over all pairwise concatenations.

$n \leq 10^6$, $|s_i| \leq 10^6$.

Author: Cocoly1990, Prepared by Izqy & Milmon

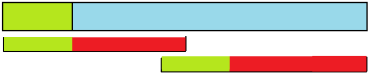
Solution

Consider classifying the border by its position in $t = s_1 + s_2 (|s_1| \leq |s_2|)$.

For the case where the border lies completely inside s_1, the border is a prefix of s_1 and a suffix of s_2. This part of the answer depends on s_1 and s_2 independently, and can be handled with hashing.

Solution

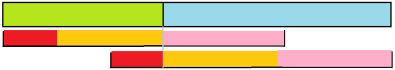
For the case where the border crosses s_1 and lies completely inside s_2, the border has the following form:



Formally, for each border of s_2, count how many whole strings s_1 can be prepended to the suffix part of this border. This is a classic AC automaton problem; preprocess prefix sums on fail-tree chains.

Solution

For the case where the border crosses both s_1 and s_2:



A prefix of s_2 is a suffix of s_1, and the remaining suffix of s_2 and the remaining prefix of s_1 are borders of the original strings respectively. The handling is very similar to the first case; hashing suffices.

Depending on the hashing implementation, the time complexity is $\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$.

Statement

Given N, M, K, compute the number of ways to place exactly K rooks on an $N \times M$ board such that every row and every column contains at least one rook.

$N, M, K \leq 10^5$.

Author: LosMem, Prepared by LosMem

Solution

First allow rooks to overlap. Then the row and column coordinates of the rooks are independent. Treat the rooks as labeled objects and the rows and columns as labeled boxes. The number of ways is $\left\{ K \atop N \right\}\left\{ K \atop M \right\} N!M!$.

After adding the restriction that rooks cannot overlap, we can use Stirling inversion to transform the count with overlaps into the count without overlaps.

Let $g_i = \left\{ i \atop N \right\}\left\{ i \atop M \right\}$ be the number of ways for i rooks when overlaps are allowed, and let f_i be the number of ways for i rooks when overlaps are not allowed.

Solution

Therefore, the answer is $N!M!\sum_{i=0}^{K}(-1)^{K-i}\left[{K \atop i}\right]\left\{{i \atop N}\right\}\left\{{i \atop M}\right\}$.

Preprocess one row of Stirling numbers of the first kind and one column of Stirling numbers of the second kind, then compute the answer directly.

Assuming N, M, K are of the same order, the time complexity is $\mathcal{O}(N \log^2 N)$.

Statement

Given a vertex-colored DAG, support modifying the color of a vertex and querying the size of the set of colors reachable from a vertex.

$n, q \leq 1.5 \times 10^5$, $m \leq 3 \times 10^5$.

Author: critnos & nzhtl1477, Prepared by critnos

Solution

First, the original problem is strictly harder than directed graph reachability, so the complexity cannot be better than $\mathcal{O}(\frac{n^2}{w})$.

Preprocess $f_{i,j}$ in $\mathcal{O}(\frac{n^2}{w})$, indicating whether vertex i can reach vertex j. Also preprocess, for each vertex, whether it can reach each color.

Then divide the timeline into blocks of size B.

Inside a block, brute-force the reachability between modified vertices and queried vertices. The time complexity is $O(qB)$.

Solution

For the rebuild step, use the property that bitsets make additions easy but deletions hard.

Therefore, for each modified color, create a new id separately, and use a mask to record which ids are currently valid.

Before each time block starts, hide the vertices modified inside this block. For the colors of these B modified vertices, process the graph once in topological order to compute whether each vertex can reach these colors after the modifications.

This part costs $\mathcal{O}(\frac{q}{B} \times (\frac{B}{w} + (n + m)))$.

Solution

After each time block ends, for the B colors that were modified, process the graph once in topological order to compute whether each vertex can reach these colors after the modifications. Push the answers for the new colors to the end of the bitsets, then put the old ids of these colors into the mask.

For every query, take the bitset of color answers and AND it with the mask of the current time, so each color is counted only by its latest id.

Since the space usage is unacceptable otherwise, we need to process block by block.

Overall time complexity: $\mathcal{O}(\frac{(n+m)q+n^2}{w})$.

Statement

Given a random deck of $2n$ cards, where there are n kinds of cards and each kind appears exactly twice, process the cards from left to right. Each card is placed into an existing stack or into a newly opened stack. Construct a stack assignment that can be eliminated according to the given rules, while using no more than 260 stacks in total.

$n \leq 5 \times 10^5$.

Author: PrincessQi & HHY_zZhu, Prepared by Cocoly1990 & HHY_zZhu

Solution

First ignore the randomized property. We can use the following method: choose a parameter x, and prepare x stacks in advance. Extract the second occurrence positions of all numbers, split them into blocks of size x, and put them into the prepared x stacks, one per stack.

For the first occurrence positions of these x numbers, put all corresponding positions into one newly opened stack.

It is easy to prove that this is correct. The total number of stacks is $\frac{n}{x} + x$.

Solution

Now consider the randomized property, temporarily ignoring the previous method.

Number each value by the position of its first occurrence. If, in every constructed stack, these numbers are strictly increasing from bottom to top, then it is easy to prove correctness.

Maintain this construction greedily from left to right: place the current card onto the available stack whose top number is largest; if none exists, open a new stack.

The number of stacks used by this greedy algorithm equals the length of the longest non-increasing subsequence of the numbering sequence.

For a random sequence of length n, the expected length of the longest non-increasing subsequence is asymptotically $2\sqrt{n}$. Thus we obtain a construction using $2\sqrt{n}$ stacks.

Solution

Combine the two methods. Consider the latter x stacks: if they are still numbered by first occurrence order, then we only need the numbers in each block to be in increasing order inside the prepared stacks.

By the analysis on the previous page, we actually only need to open $2\sqrt{x}$ prepared stacks.

Therefore, the number of stacks is $\frac{n}{x} + 2\sqrt{x}$.

Take $x = 6300$; the total number of stacks is 237. Due to random fluctuations, the problem limit on the number of stacks is slightly larger than this value.

Therefore, we solve the original problem using $\mathcal{O}(n^{\frac{1}{3}})$ stacks.

Bonus: try solving this problem within 180 stacks.