

Problem A. An Experiment in Optics Lab

The Δy is represented as: $\tan \alpha \cdot \text{width}$. If it were \sin instead of \tan , sum would be easy to find using segment tree.

So we need to move from \sin to \tan . Let's say $\tan \alpha = f(\sin \alpha)$. The idea is to approximate function f with something that is easy to sum. Let's build the Taylor series for this function:
$$f(x) = \frac{x}{\sqrt{1-x^2}} = \sum_{i=0}^{\infty} \frac{(2n)!}{4^n (n!)^2} x^{2n+1}.$$

Now choose some k first elements of this polynomial that give the good approximation, and now we can solve the problem using k segment trees.

Problem B. Banners Before QC

Due to the rule about D-jobs and T-jobs, in each coordinator's own order all their D-jobs must be completed before any of their T-jobs. Otherwise that coordinator would complete a T-job before one of their own D-jobs, which is forbidden.

Let S_M and S_L be the total durations of all D-jobs assigned to Mira and Leo. Suppose, without loss of generality, that Mira finishes her D-jobs not later than Leo, so $S_M \leq S_L$. Then all possible violations of the global rule can only happen when Mira completes her first T-job: every later T-job of Mira and every T-job of Leo is completed after that moment or after Leo has already finished all his D-jobs.

Therefore, if Mira has at least one T-job, its first T-job has to have duration strictly greater than $S_L - S_M$. We can iterate over Mira's T-jobs, count how many of them can be chosen as the first one, and then order all remaining jobs arbitrarily. If Mira has no T-jobs, there is no extra restriction.

So the answer is the product of the factorials for the independent groups of jobs: all D-jobs of Mira, all D-jobs of Leo, all T-jobs of the coordinator whose D-jobs finish later, and all remaining T-jobs of the earlier coordinator. The only additional multiplier is the number of valid choices for the earlier coordinator's first T-job. The case $S_L < S_M$ is handled symmetrically. All computations are done modulo 998 244 353.

Problem C. Choosing Best Friend

Basic, but slow, solution is to find the maximum matching in bipartite graph between V and 2^V , where each vertex is connected with all subsets of its neighborhood, and we need to find the matching covering the first part minimizing the total weight of vertices of the second part covered by this matching.

Instead of 2^V , let us just for each vertex of the left part only keep n neighbors of minimum weight, it is enough.

Problem D. Deletion-Proof Badge

First, notice that a prime-proof code can contain only the digits 0, 1, 4, 6, 8, and 9. Indeed, if one of the digits 2, 3, 5, or 7 appeared in the code, the inspector could leave only this digit and obtain a prime number. Also, every prefix of a prime-proof code must also be prime-proof: any prime number obtainable from the prefix would be obtainable from the whole code as well. Let $d[n]$ be the number of prime-proof codes of length n .

Consider the last digit of a prime-proof code. If it is even (0, 4, 6, or 8), then this last digit cannot be part of any newly created prime number: alone it is not prime, and together with previous digits it gives a number greater than one that ends with an even digit. Therefore this case contributes $4 \cdot d[n-1]$ codes.

It remains to understand the codes whose last digit is odd (1 or 9). A brute-force search for small lengths reveals all exceptional cases, and for sufficiently large lengths only the following regular families can appear:

- 800..001,
- 466..669,

- codes consisting only of the digits 0, 6, and 9.

For small lengths there are a few extra prime-proof codes with an odd last digit. They are

91, 901, 981, 649, 949, 9801, 9081, 6049, 6649, 9469, 60049, 66049, 66649, 94669,
600049, 660049, 666049, 6000049, 6600049, 6660049.

Thus we set $d[1] = 5$ and, for larger n , use the recurrence

$$d[n] = 4 \cdot d[n - 1] + 2 + 2 \cdot 3^{n-2} + \text{extras}[n],$$

where $\text{extras}[n]$ is the number of extra codes of length n from the list above. The term 2 corresponds to the first two regular families, and the term $2 \cdot 3^{n-2}$ counts the codes of length n over digits 0, 6, and 9 with a non-zero first digit and last digit 9.

Precompute powers of 3 and the values $d[i]$ up to the requested n , doing all operations modulo 998 244 353.

Problem E. Expose The Werewolf

If $m = (n - 1 - m)$, it's obviously impossible.

Assume that we don't ask questions, then we have a graph, for each connected component we have a pair of numbers (a_i, b_i) : we know that in one of the parts there are these a_i vertices, and in the other one there are b_i vertices. Now we need to solve the knapsack problem: for every pair (a_i, b_i) pick one of numbers in such a way, that the sum is equal to m . We can simplify this problem. Let's say that we add $\min(a_i, b_i)$ first, and then decide if we need to replace it with $\min(a_i, b_i)$. So we need to pick some of numbers $|a_i - b_i|$ with total sum equal to $m - \sum(\min(a_i, b_i))$. This is just a regular knapsack problem.

Let's look at the number of solutions for this knapsack problem. If it's greater than 4, then it is obviously impossible to find out which one is correct, asking two questions. Interesting fact is that if the number of solutions is at most 4, then two questions is always enough.

Now the only problem is to solve knapsack problem in given limits. One way is to use FFT knapsack in $O(n \log^2 n)$. Also there is a trick in $O(n\sqrt{n})$. Group all items of the same size (now we have $O(\sqrt{n})$ groups). If in some group there are more than 4 items, we can only take 0 or all of them, because for all other cases there are more than 4 ways to do it.

Problem F. Fix The Bad Ping

If we make a query $(1..x) \times (1..10^9)$, the result will be equal to $x \cdot 10^9 + k$, where k is the number of bad connections from exchange servers $1..x$. We can make two binary searches to find two exchange servers x_1 and x_2 with bad connections. Then in the same way we can find two fund servers y_1 and y_2 with bad connections. Now we just need to know if x_1 has bad connection with y_1 or y_2 , we can find this using one more query. So in total we need $4 \log 10^9 + 1$ queries.

Problem G. Good Coach

If $n > k$, then there are not enough digits. Even $n = k$ is impossible because 0 is not a valid bus number (it should be positive). Therefore, $n < k$. Let us look at the optimal answer. If there are two numbers of lengths that differ by at least two, we can move a digit from the longer number to the shorter one, and the maximum does not increase; so we may assume that all numbers have almost equal lengths. Moreover, we know exactly that the longest number (which will be the answer) will have length $l = \lceil \frac{k}{n} \rceil$, and there will be exactly $c = k - n(l - 1)$ numbers of length l . To minimize the maximum of them, firstly we need to assign the first digits to them: $1, 2, \dots, c$. Then the number starting with c will be our answer, and, to minimize it, we need to finish it as $c, 0, c + 1, c + 2, \dots, c + l - 2$.

Problem H. Hall of Prizes

Firstly, instead of a suffix, let's work with the whole array. If $a_i < a_{i+1}$, we could transfer one token into the future: we could make one refresh less after the i -th attraction and one refresh more after the $(i+1)$ -st attraction.

So if we take our increasing array and just greedily make the maximum number of refreshes that we can, transferring some residue to the next attraction, it will be the correct answer, but the resulting sequence is not exactly increasing. The number of refreshes decreases by at most 1 each step: if we reached the maximum M , we will always have $\geq M - 1$, which is enough to get the optimal answer.

Let's learn how to solve this problem from right to left. How will the array look after this "rebalancing to the right" operation? Some Young diagram centered in the right-bottom corner. Stack of pairs $\langle x, \text{number of occurrences of } x \rangle$. Its top is at the left bound; it is increasing from shallow to depth.

When a new element comes to the beginning of the stack, we need a rebalancing. It is quite easy to recalculate the stack. But we cannot maintain the answer directly.

Let us store two more numbers:

- **answer** — the answer on the suffix if there is literally nothing more except this suffix, and the festival route starts at the beginning of this suffix;
- **surcharge** — how many tokens we need to add to increase the answer by 1 (never able to increase by 2: if by ≥ 2 , then actually the stack wasn't rebalanced).

So we're adding a new element to the stack; during the rebalancing we have some residue — let's compare it with the surcharge. Possibly it increases the answer by 1, and now we need to recalculate the surcharge.

Problem I. Inconvenient Journey: IJ

Idea: build a graph whose vertices are pairs (x, y) , where y is a platform of the station in city x . A train line between (a_i, ta_i) and (b_i, tb_i) gives an edge with cost $(0, 1)$: it does not add a shuttle trip, but it adds one train ride. Changing from one platform to another inside the same station gives cost $(1, 0)$. Then the shortest path in lexicographic order gives exactly the answer: first the number of shuttle trips, then the number of train rides.

Now we need to reduce the size of this graph. First, notice that we only need platforms that are used by at least one ICJ train line. There are at most $2m$ such platforms in total. Instead of connecting every pair of platforms inside the same station, create one extra transfer vertex $(x, 0)$ for each city x . Add directed edges from every used platform (x, y) to $(x, 0)$ with cost $(1, 0)$, and from $(x, 0)$ back to every used platform (x, y) with cost $(0, 0)$. Passing through this transfer vertex models exactly one internal shuttle bus trip.

To start the journey, use the transfer vertex of city s as the source with distance $(0, 0)$, because Iris may choose any platform in the departure city. The answer is the minimum distance over all used platforms of city f (or, equivalently, to an extra sink connected from them with cost $(0, 0)$). The resulting graph has $O(n + m)$ vertices and edges, so Dijkstra's algorithm with lexicographic distances works fast enough.

Problem J. Just A Friendly Trick

Firstly, only the whole array. Let $dp[i][j]$ denote: the first i elements, j cycles. Either a new cycle, or an existing cycle (the number of ways is equal to the number of previously seen elements of the same color).

The generating function satisfies:

$$p_{i+1}(x) = p_i(x)(c_i + x).$$

To calculate only one coefficient, in the end we multiply by:

$$\sum x^{n - \text{favorite number}}.$$

This is $O(n \log^2 n)$.

What about all prefixes? To run a divide-and-conquer, we first calculate the answer for the left half. Then we notice that we don't need all the coefficients, but only the last $N/2$: we can divide by $x^{N/2}$ and truncate all negative powers. Then in total it will also be $O(n \log^2 n)$.

Problem K. Keyword and Numeral

Read a string, output the block up to and including = as is, then extract two summands. According to the problem statement, there are only two possibilities: if the first character is a letter (then we also output it as is) or if the first character is a digit, in which case the summand is a number (meaning there is no need to check for the presence of letters inside). For the number, set the variable *cnt* to the number of digits in it, and then print the numbered digit by digit, decreasing *cnt* by 1 after each digit. Every time *cnt* becomes a non-zero multiple of 3, print an apostrophe.

Problem L. Long and Random

Consider DP solution where $d[i][j]$ is the optimal answer where we take the prefix of array a of size i and remove exactly j elements from it. If you implement this solution and look at the number of removed elements in the optimal answer, you will notice that it is very close to $\frac{n}{9}$. Actually, experiments show that the optimal j with high probability is in the range $\frac{i}{9} \pm c\sqrt{i}$. So we can calculate only these values in DP, and skip the rest. The final time complexity will be $O(n\sqrt{n})$.

Problem M. Math, Nero and Seneca

With a trial division up to $B = 3999$ we can represent $n = p_1 p_2 \dots p_k q$ where $p_1 \leq p_2 \leq \dots \leq p_k$ are primes less than or equal to B , and q is a positive integer all of whose prime divisors are greater than B . If $q > 1$, there is no representation at all because none of the Roman numerals is divisible by any prime divisor of q , thus the answer is 0. If $q = 1$ and $k \geq 2$ and $p_1 p_2 \leq B$, then there are two different representations $n = p_1 p_2 \dots p_k = (p_1 p_2) \dots p_k$, thus the answer is 0 again. Otherwise the answer is 1: the only representation is $n = p_1 p_2 \dots p_k$ because in any other product there would be at least one composite number, but any feasible composite number is at least $p_1 p_2 > B$.