# A

### A. Polynomial in a Black Box

Jury guessed a polynomial over modulo $10^9 + 7$. Its degree $d$ is at most 10, but exact value is unknown.

Determine the degree by evaluating the polynomial in at most $d + 3$ points.

### A. Polynomial in a Black Box

The polynomial of degree $d$ is uniquely determined by its values in any $d + 1$ points. It can be explicitly restored by the values using some interpolation algorighm (e.g. Lagrange interpolation).

Let us have already asked $k$ queries and know that the degree is at least $k - 1$. Let $P(x)$ be the jury polynomial and $Q(x)$ be the polynomial we get by interpolation.

Make a random request $z$. If $P(z) \neq Q(z)$ then the degree is greater than $k$ and you should continue guessing.

If $P(z) = Q(z)$, then with high probability your interpolated polynomial is the one guessed by jury.

### A. Polynomial in a Black Box

If $P(z) = Q(z)$, then with high probability your interpolated polynomial is the one guessed by jury.

Why? Because if $P(z) = Q(z)$ then $z$ is the root of the polynomial $[P - Q](x)$. It has at most $d$ roots and you just randomly guessed one of them with probability around $d/10^9 \approx 10^{-8}$.

To be safer, you can choose another random number and validate your conjecture, thus using exactly $d + 3$ queries.

# B

### B. Board Trick

Given a board of $8 \times 8$ bits and a number $x$ from 1 to 64, change exactly one bit such that the number $x$ could be restored by looking at the board only.

### B. Board Trick

Decrement $x$ by 1. Now it is a 6-bit integer. Let $x_0, \ldots, x_5$ be the digits of $x$ in binary.

Indices of the board are 6-bit numbers. Let $s_i$ be the XOR of all bits where the $i$-th bit in the index is set to 1. We want to have $s_i = x_i$ for all $i$ after a single flip.

Consider the number $p$ where the $i$-th bit is set only if $s_i \neq x_i$. If we flip the $p$-th field of the board then the desired equality will be satisfied.

### B. Board Trick

**Strategy for Mia**

Calculate $s_i$ and flip the field with index $p$, where the $i$-th bit of $p$ is $s_i$ xor $x_i$.

**Strategy for Danila**

Calculate $s_i$ and report $x$, where the $i$-th bit of $x$ is $s_i$.

# C

### C. Connectivity

There is an undirected graph randomly chosen among all graphs with $n$ vertices and $m$ edges. You have to determine whether it is connected.

You can ask at most $2 \cdot n$ queries. Each query is a vertex. If there is an edge incident to this vertex which was not yet given to you, it is returned. Otherwise it the jury reports that no such edge exist.

### C. Connectivity

Deus ex machina fact: if a random graph is not connected then it *almost always* (with probability approaching 1) has a component which is either a tree or a unicyclic graph.

We maintain a modified DSU which stores a flag if a connected component has a cycle and iterate through vertices in any order. Whenever a new connected component is found, we keep asking about its vertices. Several things may happen:

- No more edges exist from the component to the outside. The entire graph is not connected.
- New vertex is added to the component.
- The cycle inside the component is found. If it is the first cycle, mark it in the DSU. If there already is a cycle in a component, withdraw and move to the next component.

# D

### D. Absolute Pairwise Distance

Answer several queries over an array. Each query is two subarrays $A$ and $B$, and the answer is $f(A, B) = \sum_{x \in A, y \in B} |x - y|$.

### D. Absolute Pairwise Distance

Observe that $f([l_1, r_1], [l_2, r_2]) = f([1, r_1], [1, r_2]) - f([1, l_1 - 1], [1, r_2]) - f([1, r_1], [1, l_2 - 1]) + f([1, l_1 - 1], [1, l_2 - 1])$ (inclusion-exclusion). We will now try to answer $4q$ queries to the function $g(a, b) = f([1, a], [1, b])$.

We will use SQRT-decomposition for the first argument of $g$. Let $S_1, \ldots, S_k$ be the partition of $[1, n]$ into $k \sim \sqrt{n}$ segments of length $\sim \sqrt{n}$. For each segment we will use compute a bunch of data and then process all queries by increasing of the second argument of $b$.

### D. Absolute Pairwise Distance

Consider the function $f_i(x) = \sum_{j \in S_i} |x - a_j|$. This function is piecewise linear with at most $|S_i| + 1$ parts. Let $f_{i,t}(x) = A_{i,t}x + B_{i,t}$ be the linear function on the $t$-th segment of $f_i$, and the type$(x)$ be the number of the segment of $f_i$ containing $x$.

Note that all $A_{i,t}, B_{i,t}$, and type$(a_j)$ for all $j \in [1, n]$ can be computed in $O(n)$ for each segment $S_i$.

### D. Absolute Pairwise Distance

Observe that for any set $X$ $\sum_{x \in X} f_i(x) = \sum_t A_t \cdot \text{sum}_t(X) + B_t \cdot \text{count}_t(X)$, where $t$ ranges over all types, and $\text{sum}_t(X)$ and $\text{count}_t(X)$ are the sum and the number of elements of type $t$ in the set $X$.

With this, we can compute all values $f(S_i, [1, b])$ for all $b$ in linear time, since introducing a new element into $[1, b]$ changes $\text{sum}_t$ and $\text{count}_t$ for only one type.

Thus, values of $g_1$ can be found for all queries in $O(n\sqrt{n})$.

### D. Absolute Pairwise Distance

To compute $g_2(a, b)$ we need to compute one value of $f(S'_i, [1, b])$, where $S'_i = [l, r']$ is a prefix of $S_i = [l, r]$. To do this, observe that $f_{i,r'}(x) = \sum_{j \in [l,r']} |x - a_j|$ is still linear on all segments of $f_i$, hence we still write $\sum_{x \in X} f_{i,r'}(x) = \sum_t A_{t,r'} \cdot \text{sum}_t(X) + B_{t,r'} \cdot \text{count}_t(X)$, and compute $A_{t,r'}, B_{t,r'}$ for all prefixes $[l, r']$ of $S_i$.

The total complexity is $O(n\sqrt{n})$.

## E

### E. Multiplication and Division by 2

Using only multiplication and division by 2 in an unsigned 32-bit integer type, is it possible to make a number $y$ from a number $x$?

It is convenient to see three operations:

1. remove the least significant 1 (divide by 2 until it is gone, multiply by 2 to shift the number back)
2. remove the most significant 1 (similarly)
3. shift all bits to the left or to the right

The answer is Yes if and only if the significant part of $y$ (between the most and the least significant ones) is a substring of $x$ (in binary).

# F

### F. Festive Baobab

You are given a rooted tree and have to place $t$ decorations into its vertices. The decoration at the vertex $v$ brings $d_v$ joy. There can be at most $w_v$ decorations in the subtree of the vertex $v$.

What is the maximum achievable joy?

### F. Festive Baobab

We fill the subtrees recursively. For each subtree, the recursion returns a set $(j_1, c_1), (j_2, c_2), \ldots$, meaning that in the optimal answer for the subtree there are $c_1$ decorations bringing $j_1$ joy each, $c_2$ bringing $j_2$ joy each, etc.

This set is sorted by $j$.

To build an answer for a vertex, we compose the answers for its children using small-to-large techique. If the $w_v$ constraint is violated, excessive elements are removed from the beginning of the set.

### F. Festive Baobab

Small-to-large technique is a method when we merge two sets by iterating through the smaller set and inserting its elements into the larger one.

It can be shown that, when this technique is used for a tree DP, each element is inserted at most $\log n$ times, thus there are $n \log n$ insertions overall.

# G

### G. Flatland Elections

There are $n$ points on plane. One need to choose a line passing though origin, so that total distance from $k$ nearest points to line is maximized.

Let's choose normal vector of line by rotating it through upper half-plane.

Points have some initial order. Order of two points changes twice — when line parallel to line connecting them, and when it's passing through theirs middlepoint. One can sort all events, and track order over all halfplane in $O(n^2 \log n)$ time.

Between consecutive events, sum of $k$ nearest points is fixed. Distance is maximized either on segment end, or when this vector is coliniar with normal.

### G. Flatland Elections

Technical problem — many events can happen in one point.

Solution 1. Instead of sorting, track next event of changes of adjacent points in order swaps in heap. Now, order of events in same point doestn't matter.

Solution 2. Let's track events for each point separaty. Let's compare equaly-distance point by distance from initial line position, than by id. Now order is unique at each point. For each point we can find half-segments of angle, where it is in k smallest. Put them altogether and sort.

# H

### H. Eggs

Gosha has a refrigerator capable of storing at most 16 eggs. Two kind of events may happen:

- If there is at least one egg, Gosha may eat one of them.
- If there are at most 5 eggs, Gosha may buy 10 eggs.

Devise a strategy of eating and buying eggs such that the egg eaten is one of the oldest eggs present. That is, for each reachable configuration you should post an instruction, which egg to eat and where to but the new eggs Gosha buys.

### H. Eggs

Enumerate the slots with integers from 1 to 16. In our strategy the eggs will always occupy a single segment (possibly circular, e.g. 14, 15, 16, 1, 2). Older eggs will always be to the left, newer to the right. At any point there are at most 15 eggs in the fridge, so endpoints of the segment can be determined uniquely.

When Gosha wants to eat an egg, he should take the leftmost one (e.g. (**14**, 15, 16, 1, 2) $\longrightarrow$ (15, 16, 1, 2)).

When he buys eggs, he should extend the segment to the right (e.g. (15, 16, 1, 2) $\longrightarrow$ (15, 16, 1, 2, **3**, **4**, ..., **12**)).

You can enumerate all reachable configurations by choosing the length of the segment and its first element. Do not forget about the empty fridge.

# I

### I. Removing Pairs

You are given two strings $s$ and $t$. Is it possible to get $t$ by removing adjacent pairs of characters from $s$?

The first character of $t$ should match the 1-st, 3-rd, 5-th, ... character of $s$, because any even number of characters can be removed from the front. Similarly, the second character of $t$ should match the 2-nd, 4-th, ... character of $s$.

Generalizing, the $i$-th character of $t$ should match the character of $s$ with the same parity of index. Additionally, the part after the last matched character of $s$ has to have even length (this is ensured by $s$ and $t$ having the same length parity).

Solution: iterate through $s$ and keep a pointer to the current character of $t$. If two characters match and the indices are of the same parity, take the character from $t$.

# J

### J. Boxes on a Shelf

There is a shelf of length $L$, and several rectangular boxes. We can put any number of boxes on the shelf as long as they are disjoint, and for each box its center of mass is strictly over the shelf segment. Determine the maximum number of boxes that we can put on the shelf.

### J. Boxes on a Shelf

For each box its smaller dimension should be horizontal, thus the problem is one-dimensional.

Any single box can be put on the shelf: just put its center of mass over the shelf.

Suppose there is more than one box on the shelf. Let $S$ be the sum of their widths, and $w_l, w_r$ be the widths of the leftmost and the rightmost boxes respectively. Then we can fit these boxes if and only if the segment between the centers of mass of the leftmost and the rightmost boxes shouldis be shorter than $L$, i.e. $S - w_l/2 - w_r/2 < L$ or $2S - w_l - w_r < 2L$.

Observe that it only makes sense to try and put several smallest boxes, with the largest two of those being the extreme. Hence, sort all boxes by width and try all prefixes.

# K

### K. Two Slicers

Given a circle, make two sets of cuts: $a$ equally spaced radii, and $b$ equally spaced radii. Minimize the maximum size difference among all pairs of $a + b$ resulting pieces.

### K. Two Slicers

Without loss of generality, make $a$ cuts at angles $0, \frac{2\pi}{a}, \ldots, (a-1) \cdot \frac{2\pi}{a}$ with the $0x$ axis. Then the $b$ cuts are made at angles $\varphi, \varphi + \frac{2\pi}{b}, \ldots, \varphi + (b-1) \cdot \frac{2\pi}{b}$ for some value $0 \leqslant \varphi < \frac{2\pi}{b}$.

We argue that the optimal value of $\varphi$ is a multiple of $\frac{2\pi}{2ab}$. Observe that position of each cut is a multiple of $\frac{2\pi}{ab}$, possibly, plus $\varphi$, and the size of each piece is a difference between consecutive cut positions, hence, equal to a multiple of $\frac{2\pi}{ab}$, possibly, $\pm\varphi$.

### K. Two Slicers

As $\varphi$ changes between two consecutive multiples of $\frac{2\pi}{2ab}$:

- if a cut passes another cut, then we have $\varphi =$ a difference between multiples of $\frac{2\pi}{ab}$, hence it doesn't happen;

- if two cuts' comparison by size is reversed, then either $\varphi$ of $2\varphi$ is a multiple of $\frac{2\pi}{ab}$.

It follows that between consecutive multiples of $\frac{2\pi}{2ab}$ the answer is a linear function in $\varphi$, and its minimum is at one of the endpoints.

We can now try $\varphi = 0, \frac{2\pi}{2ab}, \ldots, (a-1) \cdot \frac{2\pi}{2ab}$. Each option takes $O(a+b)$ time to evaluate, thus the total complexity is $O(a(a+b))$ (small additional factors are ok).