A

A. Ancient Cards

Given a subset of playing 52 cards. We need to rearrange them in a way that for any two adjacent cards it is true that they either share the suit, or the value.

Outline: Brute force.

A. Ancient Cards

If there is a solution that there should be a solution of one of two types:

First type: all cards of suit A, then all cards of suit B, then all cards of suit C, then all cards of suit D, so we can group all cards with suits.

Second type: all cards of suit A, then some cards of suit B, then all cards of suit C, the the rest cards of suit B, and finally all cards of suit D.

With these observations that is easy to implement the full search.

B

B. Barbara's Robot

Given a board that consists of these symbols: `<`, `>`, `^`, `v`, `\`, `/`, `|`, `-`, `x`, `o`. Then we performed several (up to 10^6) operations of rotation or flips (horizontal, vertical or diagonal). Symbols were rotated as well with their positions: for example. symbol `|` after rotation to the right becomes `-`, and symbol `>` after the main diagonal flip becomes `v`. You need to output the result board.

B. Barbara's Robot

Naive approach could be: honestly apply all operations one by one to the board.

Complexity: $O(n \times n \times \#$ of operations$)$.

Too slow :(And we need to code all of these rotations and transformations...

B. Barbara's Robot

Observation: we don't need to know the state of entire board in the middle of that process. We need to know the positions of its corners.

Let's store the positions of each corner of the board as a vector of 4 numbers. For example the initial position is: $0, 1, 2, 3$ if we enumerate each corner starting from upper left clockwise direction.

Now every operation can be represented as a permutation of 4 numbers. For example right rotation is permutation $3, 0, 1, 2$ (left upper corner becomes left lower corner and so on).

Let's apply all of the operations to our state so in the end we have some permutation of 4 numbers which represents our board.

B. Barbara's Robot

Now we need to restore the answer. To do that we can implement only 2 transform operations: rotate right and flip over diagonal.

For example, if the result permutation is $2, 3, 0, 1$ we need to run 2 right rotations.

If the result permutation is $0, 3, 2, 1$, then we need to flip the diagonal.

Note that you need to carefully implement all of these rotations of symbols.

C

C. Colors on a Stadium

Given a square of letters $n \times n, n \leqslant 26$. The size of alphabet is also n. Every row and every column consists of distinct letters.

We replace the letter at some position (only one) with any other letter. We need to find the original letter and that position.

Outline: easy problem: handle 2 cases.

C. Colors on a Stadium

Example:

$$ABC\ BDA\ CAB$$

Letter "D" is the one we need to output (position $2, 2$).

$$ABC\ BAA\ CAB$$

We can replace C (position $2, 2$) with some presented letter!

C. Colors on a Stadium

For each letter we can calculate how many times it is present on the board. For the original board every letter is presen exactly n times.

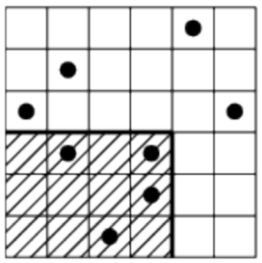
Case 1: we replaced the letter with some letter that was not on the board before. After that we will have 2 letters with count 1 and $n-1$. The second letter is the original one and the position we need is the position of the letter with count 1.

Case 2: we replaced the letter with some letter that was on the board before. We will have 2 letters with counts $n+1$ and $n-1$. The second letter is the original letter. To find the position we need to find the position of one of $n+1$ letters we replaced. We need to iterate over all of them and check if that column and row both have 2 of these letters.

D

D. Divide and Sell

Given a rectangular board with some cells marked. We need to find a subreactangle with one end in the corner of the board that this subrectangle contains exactly half of these selected cells. This subrectangle should have minimum possible area. Board side can be up to 10^9. Number of marked cells up to 10^6.



Outline: Maps and two pointers.

D. Divide and Sell

First notice that if we can solve the problem for upper left corner we can rotate our board 3 times and solve for each corner independently and choose the best answer.

Also we can say that if the number of marked cells is odd then we have no solution.

D. Divide and Sell

Let's sort our marked cells (first on row, then on column).

Let's add our cells one by one. We will store the height of the current rectangle. We can notice that the height never increases when we add points.

Basically we will have map of all X positions and count of the marked cells on these positions. We will add all cells on the row. Now we can check if current rectangle still has less than $n/2$ marked cells we do nothing. If it has exactly $n/2$ cells we can update and answer.

What if we have more than $n/2$ marked cells now? We will just move our pointer to the left until we have less or equal to $n/2$ marked cells.

E

E. Elves

Given a tree with $n \leqslant 200$ nodes. We need to find the number of ways to select some edges of that tree with exactly $k \leqslant n$ vertices.

Outline: Dynamic Programming

E. Elves

Let's calculate the following dynamic programming function with 3 parameters: v — current vertex, t — how many vertices we have taken for the subtree and the state of the root: not taken, taken with no neighbor, taken with at least one neighbor.

How to calculate it?

E. Elves

Let's iterate through the edges from the vertex, the size and the size of the subtree for the given edge.

Well, if current root has a state 0 the other vertex should be either in state 0 or 2.

If the root is in the state 1 we could not have anybody in the child, so that is just 0.

If the root is in the state 2 we could have it as the first marked child so we need to make a transition from the state 1 and the child could be in state 2 or 1, or our root is already have a pair so we can add a child of any state.

If current vertex is v and child is u and d is the current state of DP for v and t is a state for u then

$d_0[i] += d_0[i-j] * (t_0[j] + t_2[j])$

$d_1[i] += d_1[i-j] * (t_0[j])$

$d_2[i] += d_2[i-j] * (t_0[j] + t_1[j] + t_2[j]) + d_1[i-j] * (t_1[j] + t_2[j])$

F

F. Fox and Goose

Given a grid $n \times m, n, m \leqslant 1000$. Some cells a marked and there is a special cell. We want to build a wall around the marked cell so that we cannot reach the outer side of the grid from any of the marked cells. The wall can have self touches but not self intersections. The wall should have the smallest possible length.

F. Fox and Goose

Let's first find the bounding box for the given marked cells.

First we can notice that if there is a rectangle formed with the special cell and one of the corners the bounding box is the answer.

What if not? That means that the special cell is inside the bounding box.

Let's find the closest way out from our special cell and that route is what we need to extend our wall.

F. Fox and Goose

UFO came and erased the rest of this problem's analysis. =(

Hopefully, we will add it later.

Any comments from the participants?

G

G. Gregor's Vacations

Given n simple polygons. Total number of vertices of all polygons is $m \leqslant 10^3$. We need to find the line that crosses the most number of polygons (touching is not counted towards crossing).

Outline: Geometry with scan-line.

G. Gregor's Vacations

When does the line cross the polygon? This happens then there exist two points on different sides of the line which belong to that polygon.

Also we can notice that if we have an answer we can always move the line so that it contains some vertex of the polygon. We just move that line to some small *eps* and include that polygon to the answer.

So first step: let's fix some point and make it the origin (by translating all other points). We can just rotate the line now and process the events of one type: point for polygon changed the location from one side to other. We also keep count of number of polygons which have points on both sides.

Like most of similar problems, the implementation is a little tricky. So you need to code it very carefully.

H

H. Hydrology

Given array of $n \leqslant 10^5$ integers. Every integer has a time moment assosiated to that. Times are given in ascending order.

Also you are given $c \leqslant 10$ queries. Query has some aggregating function (average, min or max), time range, value and relation (greater or smaller). We need to find the number of consecutive integers in our array, such that we include maximum number of integers within given time range and if we apply aggregation function to these integers the result value has the given relation to given value.

Confusing...

H. Hydrology

Example!

Let's have an array $1, 1, 5, 7, 9$ and times: $1, 2, 10, 20, 50$. Let's have one query: $min, 10, 4, smaller$.

Which segments we need to check?

Segment 1 (fits into last 10 minutes starting from 1). Minimum is $1 \leqslant 4$. $res++$

Segment $1, 1$ (fits into last 10 minutes starting from 2) Minimum is $1 \leqslant 4$. $res++$

Segment $1, 1, 5$ (fits into last 10 minutes starting from 3) Minimum is $1 \leqslant 4$. $res++$

Segment $5, 7$ (fits into last 10 minutes starting from 4) Minimum is $5 > 4$.

Segment 9 (fits into last 10 minutes starting from 5) Minimum is $9 > 4$.

$res == 3$

H. Hydrology

The solution is easier than the problem statement.

First we iterate over all right ends. We can find the left end with binary search or just keeping second pointer.

Now for given segment we need to caclulate the function.

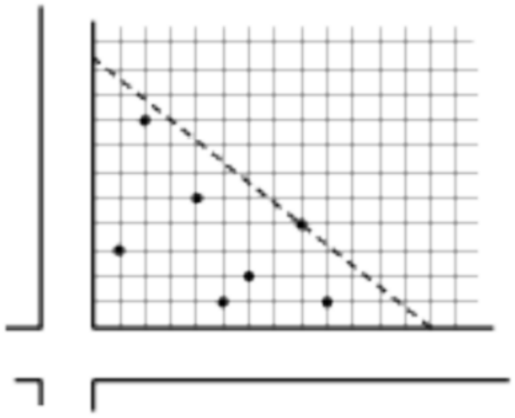
For average we can keep the partial sum array ($s_i = suma_j, j \leqslant i$), then average is $(s_i - s_{j-1})/(i - j)$.

For min and max we can use segement tree, Fenwick tree or RMQ table.

I

I. Industrial Buildings

Given set of n points with positive coordinates ($n \leqslant 10^6$). You need to find the segment that will form a triangle with coordinate lines and this triangle contains all of these points. The length of that segemnt should be as small as possible.



Outline: Convex hull.

I. Industrial Buildings

First let's find the point with maximum coordinate x. Let's add point $(x, 0)$ to the set. That should not change the answer. Let's do that same with maximum y coordinate.

Now let's find the convex hull of these point. We need only the convex part of points starting and ending in the points we have added.

It is easy to see that our answer can be one of the following lines: line passing through two consecutive points of that convex hull or passing through one point of that convex hull.

The first case is easy: we just iterate over these consecutive points and check all of these lines. But what to do with the second case?

I. Industrial Buildings

We can just run the ternary search and find the min of that function or we can find the formula for the minimum of that function.

If our current point has coordinates (x, y), then the formula is $a = arctan(\sqrt[3]{y/x})$. Then the Y coordinate of our intersection is $y + x * tan(a)$, X coordinate is $x + y * tan(a)$.

J

J. Jumps

You have $n \leqslant 1000$ numbers where each number less or equal than 10^9. Two numbers are called similar if they are represented with the same set of digits. Count number of different sets.

Example of similar numbers: $123, 332211, 333221$, or $5, 55, 555, 5555$

J. Jumps

For each number we can calculate the bit mask with 10 bits. Each bit represents the digit. If bit is on, the corresponded digit is presented.

For example for number 1401 we need to have bits $0, 1, 4$ to be on, other bits should be off: 0000010011_2 (with leading zeros) $= 19_{10}$

We can put all of these representations into a set and in the end output its size.