Problem A. Rain Diary

Author: Mikhail Ivanov, developer: Rita Sablina

There are 14 days between today and the last record. Record for last Sunday should be made after 7 days from the two weeks ago record and 7 days before today's record. If $n \geq 8$, the last Sunday was in the current month, Petya should sign the missed record with $n - 7$. Otherwise, if $n \leq 7$, the last Sunday was last month, Petya should sign the missed record with $m + 7$.

Problem B. Magnetic Games

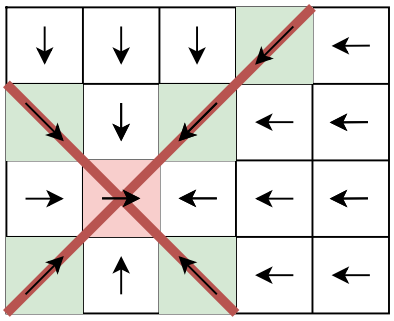
Author: Semen Chebykin, разработчик: Pavel Ralnikov

$O(n^2m^2)$ solution:

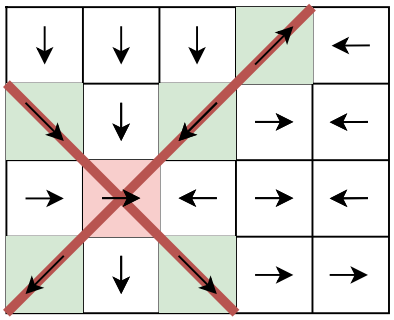
Let's go through positions of the magnet. If the magnet's position is fixed, we can make the table of the directions of the arrows uniquely. Let's check, if directions of the arrows in the constructed table are equal to the appropriate directions of the arrows in the original table. There are only one column and only one row, in which directions are different, moreover, in this row and column directions have to be inverted to the original. If it is true, we found the solution.

Fast solution:

Let's consider cells of the table, in which arrows of the compasses are directed diagonally. Note, that if an anomaly changed the direction of diagonal arrow, its direction is still diagonally. Also note, that all the cells form two diagonals, one goes from top to bottom and from left to right, other — from top to bottom and from right to left. The magnet lies on the cross of these diagonals (an arrow in the cell with the magnet may be directed not diagonally).



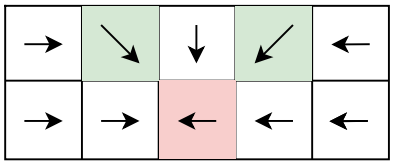
Here and further cell with a magnet is red and cells with a diagonal direction of the arrow are green. Also in this illustration diagonals mark with red lines.



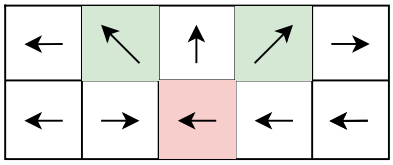
Same example, but with anomaly which inverts readings of the compasses on the fourth row and in the fourth column.

Let's cross these diagonals and find a number of the row and column, in which readings of the compasses are inverted in the $O(nm)$ time.

This solution has some problem cases. If $n = 2$ or $m = 2$ diagonals may not be detected:



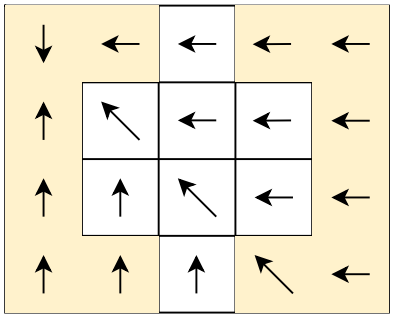
All arrows point to the magnet.



Same example but with anomaly which inverts readings of the compasses in the first row and in the first column.

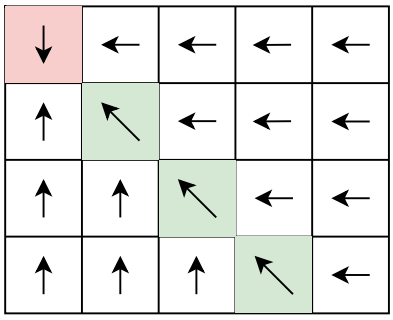
In this case, let's run $O(n^2m^2)$ solution, it will work fast for $n = 2$ or $m = 2$.

If the magnet is located in one of the four corner cells or one of the adjacent to them, one of the diagonals could not be detected.

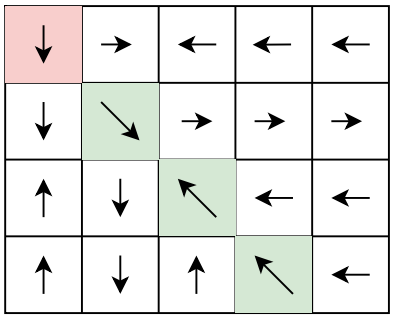


Cells which are described above are yellow.

An example of the case magnet is located in a corner cell:



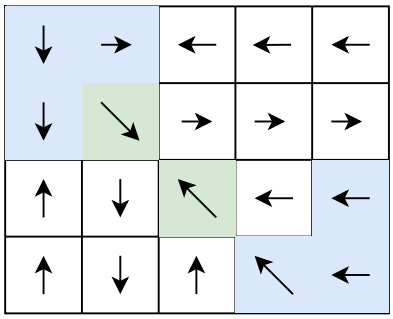
All the arrows point to the magnet.



The Same example, but with the anomaly which inverts reading of the compasses on the second row and in the second column.

In this case, let's look at the formed diagonal.

If it goes from top to bottom and from left to right (like in the case above) magnet might be in only six positions: in the leftmost top cell and two adjacent to it or in the rightmost bottom and two adjacent to it. Let's check these cells, the answer is one of them.



Same example with the anomaly. Cells that need to be checked are blue.

Else if the diagonal goes from top to bottom and from right to left, the magnet also might be in only six positions: in the rightmost top and two adjacent to it, and leftmost bottom and two adjacent to it. By analogy let's check these cells and find the answer.

Problem C. Campfire Riddle

Author: Dmitriy Klepov, developer: Mikhail Perveev

Main solution.

Let's formalize the problem statement: we have to construct a graph on n vertices. The graph has an edge between two vertices if and only if the degrees of these vertices are equal, with a minimum possible number of edges. Let's notice that all vertices with equal degrees will be pairwise connected with edges, so they will form a complete subgraph. Also, there are no two vertices with different degrees connected with an edge, so each connected component of the graph is a clique, and degrees of vertices in different components are pairwise different.

So we have to choose such sizes of connected components $s_1, s_2, \ldots, s_k$ that the following conditions are satisfied:

1. All sizes of connected components are pairwise distinct: $s_1 < s_2 < \ldots < s_k$.
2. The total number of vertices equals to n: $s_1 + s_2 + \ldots + s_k = n$.
3. The total number of edges $\frac{s_1(s_1-1)}{2} + \frac{s_2(s_2-1)}{2} + \ldots + \frac{s_k(s_k-1)}{2}$ is minimum possible.

Let's solve this problem using dynamic programming. Let $dp[n][max]$ is the minimum possible number of edges in the graph on n vertices if the maximum size of the connected component is not greater than max. The initial values: $dp[0][i] = 0$ for each $i = 0 \ldots n$. The transition: $dp[n][max] = \min\left(dp[i][max-1],\ dp[i-max][max-1] + \frac{max(max-1)}{2}\right)$. The answer is $dp[n][n]$.

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

Alternative solution.

Let's analyze what the answer is for small values of n and detected the pattern. Let max be the minimum integer such that $1 + 2 + \ldots + max \geq n$. Consider two cases:

1. If $1 + 2 + \ldots + max = n$, there will be exactly $\sum_{i=1}^{max} \frac{i(i-1)}{2}$ edges in the optimal answer.

2. Otherwise, let $t = (1 + 2 + \ldots + max) - n$. Then there will be exactly $\left(\sum\limits_{i=1}^{max} \frac{i(i-1)}{2}\right) - \frac{t(t-1)}{2}$ edges in the optimal answer.

Time complexity: $\mathcal{O}(\sqrt{n})$, $\mathcal{O}(\log n)$ or $\mathcal{O}(1)$ depending on the method of calculating the value of max and the sum of squares of consecutive integers.

Bonus solution.

Let $f(x) = \left\lfloor \sqrt{2x + \frac{1}{4}} + \frac{1}{2} \right\rfloor$ and $s = f(n)$. It turns out that the answer can be found using the formula:

$$\frac{s(s-1)(s-2)}{6} + \frac{\left(n - \frac{s(s-1)}{2}\right) \cdot \left(s^2 + 3s - 2(n+1)\right)}{4}$$

You can refer to the following sequence in OEIS for details: https://oeis.org/A121924.

Time complexity: $\mathcal{O}(\log n)$ or $\mathcal{O}(1)$ depending on the method of calculating the square root.

Problem D. The Tree

Author: Andrey Zavarin, developer: Victor Romanenko

Denote by S - the sum of the lengths of the paths to all the vertices of the query. We will store the tree lazily, maintaining only those vertices that participated in the queries. It can be done, for example, using a structure on pointers, where we create a new vertex only when we enter it during a request.

Now let's learn how to respond to queries.

For the first type of request, let's go down to the vertex of the request and put two values: *timer* — the number of the current request and *color* — the color in which the vertex was painted. Initially, in every vertex both of these values are -1.

For the second type request, let's locally maintain two values *timer* and *color* — the number of the last encountered color change request on the path from the root and the color in which the current vertex should be colored. If at the current vertex v its value *timer* is greater, then we change the current values of *timer* and *color* to the values recorded at this vertex. When descending, we change the values of *color* according to the rules from the problem condition. When we get to the vertex of the query, the answer is the *color* variable.

The solution works for $O(S)$ of memory and time.

Problem E. Just Like Pickle

Author: Alexey Luchinin, developer: Nikolay Budin

You should construct x in form of $\sum\limits_{i=1}^{k} s_i \cdot 2^{p_i}$, where $s_i \in \{-1, 1\}$, $0 \le p_i$, and k is the minimum possible.

Let's make some notes:

- The optimal answer doesn't have two equal p_i.
  - If $p_i = p_j$ and $s_i \neq s_j$, then these two summands gives 0 together, and we can remove them.
  - If $p_i = p_j$ and $s_i = s_j$, then these two summands can be replaced by a single $s_i \cdot 2^{p_i+1}$.
- There's an optimal answer that doesn't have two summands satisfying $p_i = q$, $p_j = q + 1$.
  - If $p_i = q$, $p_j = q + 1$ and $s_i \neq s_j$, then they can be replaced by a single $s_j \cdot 2^{p_i}$.

  - If $p_i = q$, $p_j = q + 1$ and $s_i = s_j$. Let's choose such a pair that the corresponding q is the minimum possible. And choose such an optimal answer, that this minimum q is the maximum possible. Then let's replace a pair $s_i \cdot 2^{p_i}$ and $s_j \cdot 2^{p_j}$ by a pair $-s_i \cdot 2^{p_i}$ and $s_j \cdot 2^{p_j+1}$. Sum won't change, a number of summands won't change, but the minimum q for which exist a pair $p_{i'} = q$ and $p_{j'} = q + 1$ will increase. But we've chosen such an answer, that this q is the maximum possible. Contradiction.

- An answer for the x equals to an answer for the $|x|$. So we will solve the problem only for non-negative numbers.
- Answer for the $x = 0$ equals to 0.
- Answer for the $x \bmod 2 = 0$ equals to the answer for the $\frac{x}{2}$. You can build an answer for x from the answer for $\frac{x}{2}$ and vice versa.
- Answer for the $x \bmod 4 = 1$ ($x = \overline{\ldots 01}_2$) contains summand $+2^0$. It should contain either $+2^0$ or -2^0. If it contains -2^0, it will also contain 2^1 or -2^1. That means that an answer has two summands with p_i differ by 1.
- Answer for the $x \bmod 4 = 3$ ($x = \overline{\ldots 11}_2$) contains summand -2^0. Similar to the previous.

Eventually, we will calculate an answer using the following recursive formula:

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(\frac{x}{2}) & \text{if } x \bmod 2 = 0 \\ 1 + f(\frac{x-1}{4}) & \text{if } x \bmod 4 = 1 \\ 1 + f(\frac{x+1}{4}) & \text{if } x \bmod 4 = 3 \end{cases}$$

It is easy to see the time complexity is $O(\log x)$.

Problem F. Lazy to Win

Author: Alexey Luchinin, developer: Grigory Shovkoplyas

Let's assume that Alexey's optimal strategy is to choose problems with numbers from l to r inclusive and skip the problem with the number k.

Note that if among the numbers $a_l, a_{l+1}, \ldots a_r$ there is a number less than a_k, then Alexey can skip this problem instead of k-th, and the number of points scored will increase. This means that with fixed l and r, in order to receive the maximum number of points, we must skip the minimum cost problem on the segment.

Then we reformulate the original problem. It is required to find in the array a segment of numbers $a_l, a_{l+1}, \ldots a_r$ such that $\sum_{i=l}^{r} a_i - \min(a_l, a_{l+1}, \ldots a_r) \geq \lceil \sum_{i=1}^{n} a_i/2 \rceil$.

This problem can be solved using the method of two pointers and a minimum in a sliding window. We store the segment in such a way that the difference between its sum and minimum is at least $\lceil \sum_{i=1}^{n} a_i/2 \rceil$. The minimum on the segment can be calculated using a sparse table or queue with a minimum or a multiset.

The final complexity of the solution is $O(n)$ or $O(n \log n)$, depending on the method of calculating the minimum on the segment.

Problem G. The Length of the Sequence

Author and developer: Maria Zhogova

First of all let's notice an obvious fact: if we could construct two correct segments that start at l_1 and l_2 and $l_1 < l_2$, then the length of the second segment is not greater than the length of the first segment.

Let's find a segment $[0...l]$ with minimum possible length such that the length (sum) of the corresponding string in decimal notation is at least S. We can do it using a simple precalculation.

Now we are going to consider integers with equal length together and claim them as a block of integers of length x.

Consider the number of digit of integer l (denote it as len). Let's calculate $r = sum - S$. It is easy to see that $r < len$.

Now we will use the two pointers technique: we will increase the left pointer while the length of the string is greater than we need, and increase the right pointer otherwise. We will be able to find the leftmost correct segment using this method.

If $r \leq 10$, we can remove some prefix of one-digit integers from the answer and get the correct length. It is obvious that this answer is the leftmost among all answers.

It is easy to precalculate that $len \leq 17$. According to the fact described above, we are interested only in cases when $len \geq 12$.

Now let's estimate the length of the left integer of the answer when we will stop our process.

We can notice that if we did not achieve the blocks with lengths 15 and 5 simultaneously (using different pointers), we would be able to collect the required sum using integers with length 5 (or may be less) according to the fact that len and 5 are co-prime. The last case is when our pointer will move from the block with length 14 to the block with length 15 and the block of length 15 itself.

The block of integers with length 15 will be handled after at most 100 operations. If we moved our pointer out of block of length 15 during the process, then the claim above is satisfied because we have not achieved the block 5 yet.

The last case is the transition between blocks 14 and 15. We can understand that if transition will be after the block 5, it contradicts the claim described above. The transition also could not be after the block 3 because 3 and 14 are co-prime. So the transition was before that block so we will be able to collect sum in the block 15 before we achieve block 4.

Problem H. Pines

Author and developer: Alexander Gordeev

It is obvious that we only need indexes of "A"-type lamps. We have the optimal answer only if one half of "A"-type lamps shines red, and the other half shines blue.

We want to get any optimal permutation, so let's find the one where in the beginning all lamps are blue or white and then, starting at some point, all lamps are red or white.

Let m be the number of "A"-type lamps. Let's make an array of indexes a, where a_i is the index of i-th "A"-type lamp in the initial array ($1 \leq i \leq m, 1 \leq a_i \leq n$). Then all lamps with indexes $a_1, a_2, \ldots, a_{\frac{m+1}{2}}$ should shine blue and all lamps $a_{\frac{m+1}{2}+1}, a_{\frac{m+1}{2}+2}, \ldots, a_m$ should shine red.

To achieve said result it's enough for the permutation to ascend until the element indexed $a_{\frac{m+1}{2}+1}$ and descend after it. Indeed, if the color of the lamp depends on the difference between the current element and the next one, then the first lamps will be shining blue and the lamps after them will be shining red.

Then one of the optimal answers is a permutation $1, 2, 3, \ldots, a_{\frac{m+1}{2}}, n, n-1, n-2, \ldots, a_{\frac{m+1}{2}}+1$. To get such a permutation it's enough to reverse the $[a_{\frac{m+1}{2}}+1; n]$ segment of a monotonously increasing permutation.

Problem I. Circus Performance

Author and developer: Daniil Oreshnikov

Let's take a look at the condition for some triple of acrobats to be good. It's just the following inequality:

$$a_i b_j + a_j b_k + a_k b_i \geq a_i b_k + a_j b_j.$$

This inequality by itself does not make it clear in what order the acrobats should be lined up. But if we move everything to the left side and group specific items together, we get

$$((a_j - a_i)b_k + a_i b_j) + ((a_k - a_j)b_i - a_j b_i) \geq 0.$$

Then let us add $-a_j b_j$ to the first part and $a_j b_j$ to the second. The inequality will remain true, but the expressions in parentheses can then be represented as $(a_j - a_i)(b_k - b_j)$ and $(a_k - a_j)(b_i - b_j)$. For the points $P(a_i, b_i)$, $Q(a_j, b_j)$, and $R(a_k, b_k)$ on the plane the inequality that we've obtained corresponds to the left-hand rotation predicate from vector $\overrightarrow{PQ}$ to vector $\overrightarrow{QR}$.

Thus, one possible way to solve the task is to depict the acrobats by points on the plane with coordinates equal to their height and weight and then arrange the points so that the transition to each following pair of points corresponds to a left turn.

To do this, we could use the Jarvis algorithm for finding a convex hull. First, we find a convex hull — on this convex hull each following side is rotated counterclockwise relative to the previous one. But there could still be points that are not included in the convex hull. To include them in the sequence, we will let the algorithms continue working even after it reaches the starting point and completes the convex hull. On each step we'll be choosing the vertex with the minimum angle from all the vertices not yet included in the sequence. Eventually such an algorithm will construct a "spiral" wrapping inward counterclockwise, on which all the points will be ordered in the right way.

Problem J. Square Running

Assume the photographer takes a photo when all runners are in R_p. There are two possible positions for each runner in the photo: to the left and to the right from the photographer. Let's consider all 2^n variants.

We know the runner's i coordinates for each variant: runner i is on the lane i, row R_p, a column number depends on the position of the runner from the photographer. It is possible to calculate d_i — the distance that runner i should cover to get the proper position for the first time. Also, let's calculate p_i — the length of the lane i. Runner i will get R_p and proper position from the photographer for all times t, equal d_i, $d_i + p_i$, $d_i + 2 \cdot p_i, \ldots$ In other words, the number t should have the remainder d_i when divided by p_i. We get a system of equations type $t \equiv d_i \pmod{p_i}$.

A system of such equations can be solved by the Chinese remainder theorem. If a system of equations does not have a solution, it is not possible to take a beautiful photo under given conditions.

There are similar assumptions for a possible photograph with all runners in C_p above or below the photographer. The answer is a minimum of all obtained times t or -1 if all systems of equations have no solution.

Let's estimate the size of the answer. The resulting solution will not exceed the least common multiple of all p_i. The limitation that $R_R - R_L + C_R - C_L$ is divisible by 4 ensures that all numbers p_i are divisible by 8, then the LCM of these numbers does not exceed $9 \cdot 10^{18}$. Note that intermediate numbers, depending on the implementation of the Chinese remainder theorem, may exceed $9 \cdot 10^{18}$. The `__int128` type could be used to avoid this.

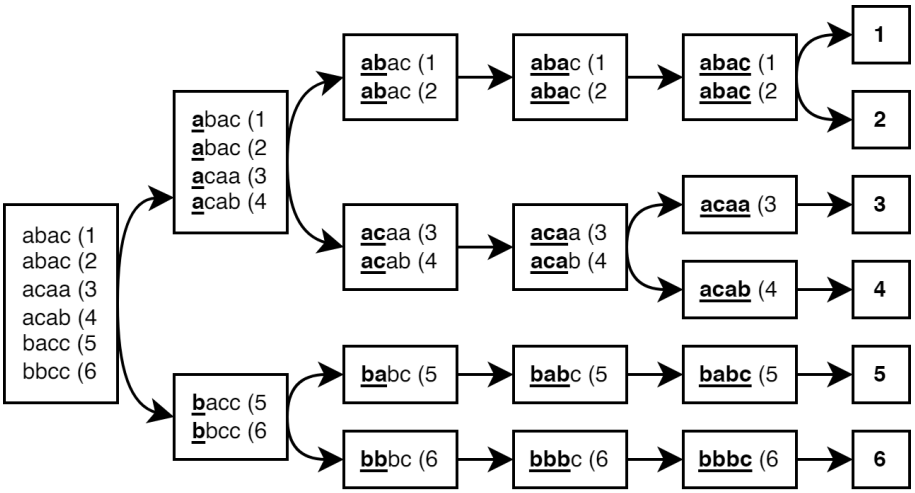
Problem K. Pick a Pair

Author and developer: Konstantin Bats

Suppose the answer is k. Consider a prefix of length k in each of the words. Let's split the selected prefixes into blocks so that there are identical prefixes in one block. Let's these blocks will have a level of k.

The number of words in each of the blocks is an even number. If it is not, then some word doesn't have a pair, which contradicts that k is the answer to the problem.

Consider the blocks for the prefixes $k-1$. This block is either a block that exists at the k level, or a union of several blocks from the k level. It follows from this that there are an even number of words in all blocks of the level less than k.



So, the problem can be formulated as to find such a maximum k that the blocks of the k level contain an even number of elements.

It is easy to do if the words are sorted lexicographically. Let's consistently consider each level from 1 to the length of the word. Suppose that before this the words were divided into blocks of the $k-1$ level with an even number of elements. Consider the k level. Let's select groups of consecutive words whose characters at position k match. Blocks of the k level contain an even number of elements if and only if each of the groups has an even number of elements. It's true because, if the blocks were previously divided into an even number of elements, now some blocks have been divided into several, so each still has an even number of elements, and some have remained the same.

Thus, to solve the problem we have to sort the words lexicographically and to check the levels sequentially until we find the maximum. One level checking spends linear time from the number of words, the desired level can be found in linear time from the total length of the words.

In total, the time complexity is $\mathcal{O}(nm \cdot \log(nm))$, where m — the length of one line.

Problem L. Shifting Roads

Author: Fedor Tsarev, developer: Grigory Khlytin

Consider three roads i, j, k.

Implement the function `check(`i`, `j`, `k`)`, which will check whether it is possible to select roads $\{i, j, k\}$ for the successful completion of the modernization program. Note that for a positive answer to this question, it is necessary and sufficient to fulfill at least one of the following conditions:

- `dist(`i`, `j`)` $\le$ `length(`k`)` — the distance between the roads i and j is not greater than the length of the road k
- `dist(`i`, `k`)` $\le$ `length(`j`)` — the distance between the roads i and k is not greater than the length of the road j
- `dist(`j`, `k`)` $\le$ `length(`i`)` — the distance between the roads j and k is not greater than the length of the road i

If at least one of these conditions is met, we will always be able to shift exactly one road so that the three resulting roads form a coherent asphalt-covered section.

To check these conditions, we implement the function `dist(`s`, `t`)`, which will find the distance between the roads s and t. Since each road is a segment, we proceed to the problem of finding the distance between the segments on the plane.

Most methods of computational geometry are implemented by finding the dot and cross product of vectors:

- `dot_prod(`$\vec{v}_1$`, `$\vec{v}_2$`)` $= x(\vec{v}_1) \cdot x(\vec{v}_2) + y(\vec{v}_1) \cdot y(\vec{v}_2)$
- `cross_prod(`$\vec{v}_1$`, `$\vec{v}_2$`)` $= x(\vec{v}_1) \cdot y(\vec{v}_2) - y(\vec{v}_1) \cdot x(\vec{v}_2)$
- `length(`$\vec{v}$`)` $= \sqrt{x^2(\vec{v}) + y^2(\vec{v})}$

Recall that by the condition each of the segments is given by two different points. Let the segment s be given by two points A_s and B_s, and the segment t is given by two points A_t and B_t.

Note that the distance `dist(`s`, `t`)` between the segments s and t is zero if and only if the segments intersect, otherwise the distance between the segments is equal to the minimum of four distances:

- `dist_point_to_segment(`A_s`, `t`)`
- `dist_point_to_segment(`B_s`, `t`)`
- `dist_point_to_segment(`A_t`, `s`)`
- `dist_point_to_segment(`B_t`, `s`)`

We implement the function `dist_point_to_segment(`P`, `u`)` to calculate the distance on the plane between the point P and the segment u (which is also given by two points A_u and B_u). Note that if the condition is true:

$$\texttt{dot\_prod}(\vec{v}(A_u, B_u),\ \vec{v}(A_u, P)) \geq 0 \texttt{ and dot\_prod}(\vec{v}(B_u, A_u),\ \vec{v}(B_u, P)) \geq 0$$

then the perpendicular from the point P falls on the segment u, which means that the answer — is the distance between the point and the line:

- `dist_point_to_line(`P`, `u`)` $= |$ `cross_prod(`$\vec{v}(A_u, B_u)$`, `$\vec{v}(A_u, P)$`)` / `length(`$\vec{v}(A_u, B_u)$`)` $|$

If this condition is incorrect, then the perpendicular from the point P does not fall on the segment u, which means that the answer — is the minimum of the distances:

- `dist_point_to_point(`P`, `A_u`)` $=$ `length(`$\vec{v}(P, A_u)$`)`
- `dist_point_to_point(`P`, `B_u`)` $=$ `length(`$\vec{v}(P, B_u)$`)`

Thus, we will iterate over all the triples $\{i, j, k\}$ and for each triple we will run the check function `check(`i`, `j`, `k`)`. The time complexity of the solution: $O(m^3)$.