Problem Analysis
The 2025 ICPC Asia Yokohama Regional Contest

Note: This document was generated by machine translation from the Japanese version.

Problem A: Tatami Renovation

Proposer: Ryotaro Sato   Author: Ryotaro Sato   Analysis: Ryotaro Sato

If the number of tiles is odd, tiling is clearly impossible. On the other hand, if the number of tiles is even, tiling is always possible. In what follows, we assume that the number of tiles is even.

We give a lower bound on the number of tiles that must be moved as follows. First, we divide the long corridor as finely as possible into segments so that each segment contains an even number of tiles. For each segment whose first column is a column containing exactly one tile, we determine a lower bound on the number of tiles in that segment that must be moved, based on the following reasons.

- If there are two tiles placed in some column, then clearly at least one of them must be moved. Otherwise, if both of them are left unmoved, the entire corridor would be divided by that column into two rectangles, each having an odd number of tiles.
- If we color the grid in a black-and-white checkerboard pattern, then for each segment it can be proved that, among the tiles that were initially placed in that segment, the number of tiles that finally end up on white cells and the number of tiles that finally end up on black cells must be equal (note that in the proof we must also take into account moves of tiles that cross the boundaries between the segments).

The sum over all segments of the lower bounds obtained in this way gives a lower bound on the answer to the original problem. Moreover, since there always exists a way of moving tiles that attains this lower bound, this lower bound is in fact the answer to the original problem. The expected time complexity is $O(n \log n)$, dominated by sorting the given values.

As an alternative solution, one can consider a dynamic programming approach that processes the corridor from the first column, using as the key the arrangement of tiles in the current and next columns and as the value the minimum number of tiles moved so far, but its implementation would be somewhat complicated.

Problem B: Minimizing Wildlife Damage

Proposer: Mitsuru Kusumoto   Author: Kentaro Matsushita   Analysis: Kentaro Matsushita

We first analyze the structure of a solution. After performing d operations, each a_i belongs to exactly one of the following three types: those that are removed until $a_i = 0$, those that remain with $a_i > 0$ but are decreased at least once, and those that are never decreased. We call these Block 1, Block 2, and Block 3, respectively.

In Block 1, except for positions where $a_i = 0$ already in the initial state, there are never three consecutive zeros. Otherwise, we could avoid making the middle element zero and thereby consume more of d. Also, we may assume that there are no two or more consecutive positions with $a_i > 0$. For the same reason, we can change the solution so that one of them becomes zero without worsening the solution. Under these constraints, the amount of d that can be consumed can be computed by a simple DP.

Block 2 consists of a segment of K consecutive positive values $a_t, a_{t+1}, \ldots, a_{t+K-1} > 0$, each of which has been decreased by the same number of operations, which is strictly less than $\min(a_t, \ldots, a_{t+K-1})$.

An important observation is that, from the condition involving min, Block 1 is kept as long as possible without exceeding d. Therefore, including the parity, the number of candidates for the length of Block 1 is $O(1)$.

Finally, we discuss how to answer the queries. There is still freedom in where to cut Block 2, but this can be handled as follows. We scan the sequence from the back and maintain linear functions using the Convex Hull Trick. Then, when we reach a candidate position where Block 1 ends, we can obtain the maximum value in $O(\log n)$ time.

Problem C: Seagull Population

Proposer: Shinya Shiroshita Author: Shinya Shiroshita Analysis: Shinya Shiroshita

First, an obvious lower bound on the total number of seagulls is the maximum value of b_i, namely $D_{\max} = \max(b_1, \ldots, b_n)$. This follows from the fact that the number of seagulls observed on any single day cannot exceed the total number of seagulls. Similarly, the total number of increases,

$$S_{\text{inc}} = \sum_{i=1,\ldots,n} \max(b_i - b_{i-1}, 0) \quad (\text{where } b_0 = b_n)$$

is also a lower bound on the total number of seagulls. Indeed, if $b_i > b_{i-1}$, then on day i there must be at least $b_i - b_{i-1}$ seagulls that arrive at the island. Therefore $M_{\text{opt}} = \max(D_{\max}, S_{\text{inc}})$ is a lower bound on the answer. In the following, we show that M_{opt} is optimal by giving constructions.

First, consider the case where there is a day with no seagulls. By symmetry, we may assume $b_n = 0$. In this case, starting from day 1, we greedily determine the stay duration of each seagull so that earlier-arriving seagulls stay as long as possible, as illustrated in the figure. In the figures below, each vertical column represents a day, each letter represents the stay duration of a seagull, and each number represents the total number of seagulls observed on that day. In this case, $S_{\text{inc}} \geq D_{\max}$ holds, and thus M_{opt} is optimal.

```
AAAAAAAAAA.
..BBBBBB...
...CC.DD...
-----------
11233233110
```

Next, consider the case where there is at least one seagull on every day. Let $D_{\min} = \min(b_1, \ldots, b_n)$. The number of seagulls that stay throughout the whole year is not necessarily

$D_{\min}$, and we can reduce this number by overlapping the stay intervals of seagulls as illustrated in the figure.

```
AAAAA....AAAAAA
..BBBBBBBBBB...
---------------
112221111222111
```

Now subtract $D_{\min}$ from all b_i, and consider the same greedy construction as in the first case. In the first figure, seagulls that end up at the same height (that is, groups for which the number of seagulls already staying when they arrive is the same) form groups whose stay intervals do not overlap with each other. For each such group, by cyclically shifting the end points of their intervals as shown below, we can reduce the number of year-round seagulls by at most (the size of the group -1).

```
..AAA................    ..AAAAAAAAA..........    ..AAAAAAAAAAAAAAAAA..
.......BBBB..........    .......BBBBBBBBBBBB..    BBBBB..BBBBBBBBBBBBBB
...............CCCC.. -> CCCCC..........CCCCCC -> CCCCCCCCCCC....CCCCCC
---------------------    ---------------------    ---------------------
001110011110000111100    112221122221111222211    223332233332222333322
```

If we apply this procedure at every height, we can in total reduce the number of year-round seagulls by exactly $S_{\text{inc}} - (D_{\max} - D_{\min})$. If $S_{\text{inc}} \geq D_{\max} \Leftrightarrow S_{\text{inc}} - (D_{\max} - D_{\min}) \geq D_{\min}$ holds, then by reducing the number of year-round seagulls exactly $D_{\min}$ times, we obtain a construction with S_{inc} seagulls in total. On the other hand, if $S_{\text{inc}} < D_{\max} \Leftrightarrow S_{\text{inc}} - (D_{\max} - D_{\min}) < D_{\min}$ holds, then we need

$$D_{\min} - (S_{\text{inc}} - (D_{\max} - D_{\min})) = D_{\max} - S_{\text{inc}}$$

year-round seagulls, and thus the total number of seagulls is

$$S_{\text{inc}} + (D_{\max} - S_{\text{inc}}) = D_{\max}.$$

In either case, we obtain a construction with exactly M_{opt} seagulls, and hence M_{opt} is the optimal answer.

Problem D: Decompose and Concatenate

Proposer: Kazuhiro Inaba Author: Fumihiko Yamaguchi Analysis: Fumihiko Yamaguchi

One positive integer is given; let this be s. Among all numbers obtained as follows,

- choose two positive integers a and b such that $s = a + b$, and
- concatenate the decimal representations of a and b as strings,

we are asked to find the maximum possible resulting integer. In what follows, we denote string concatenation by "." and write the representation of the number we seek as $a.b$.

Since we want to maximize the value of the positive integer, we want to make the number of digits of $a.b$ as large as possible. If we can make the number of digits of both a and b equal to that of s, then this is optimal. This is possible when the most significant digit of s is at least 2. In this case, we want to make a as large as possible (and hence $b = s - a$ as small as possible). The smallest positive integer with n digits is 10^{n-1}, so we can take b to be the largest power of 10 that does not exceed s.

However, when (and only when) the most significant digit of s is 1, we cannot decompose s into a sum of integers that all have the same number of digits as s.

If the second most significant digit of s is nonzero (for example, $s = 110$), then we can take b to be a number with the same number of digits as s, namely the largest power of 10 that does not exceed s (which is 100 in this example), and this gives $110 = 10 + 100$.

Finally, consider the case where the two most significant digits of s are 10 (for example, $s = 102$). If we took b to be the largest power of 10 that does not exceed s (which is 100 in this example), the number of digits of a would be at least 2 less than that of s, which is undesirable. In this case, we should instead take b to be a power of 10 with one fewer digit than s (which is 10 in this example), giving $102 = 92 + 10$.

Since s can be as large as 10^{17}, care must be taken when using fixed-width integer representations.

Problem E: Cutting Tofu

Proposer: Naoki Marumo Author: Masatoshi Kitagawa Analysis: Masatoshi Kitagawa

Let x be the side length of a cube to be cut out. From a rectangular prism with side lengths a, b, and c, we can cut out $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor$ cubes of this size. Here, $\lfloor \cdot \rfloor$ denotes the floor function, which returns the integer part. Therefore, this problem can be rephrased as finding the maximum $x > 0$ that satisfies $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor \geq k$. The maximum value of x is a rational number, and the problem asks us to output it as a reduced fraction.

If there were no requirement to output a reduced fraction, and some numerical error were allowed, we could find x by binary search. This is because $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor$ is a monotonically decreasing function in x. However, performing binary search on x as a floating-point number does not necessarily yield the exact answer, so we need some additional ideas.

Next, we consider how to obtain the exact value as a fraction. If a/x is not an integer, then increasing x by a sufficiently small amount does not change $\lfloor a/x \rfloor$. The same holds for $\lfloor b/x \rfloor$ and $\lfloor c/x \rfloor$. Therefore, $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor$ decreases when we increase x by a sufficiently small amount only if at least one of a/x, b/x, or c/x is an integer. From this, if x is the desired maximum value, then at least one of a/x, b/x, or c/x must be an integer. In other words, the maximum x must be of the form a/i, b/i, or c/i for some positive integer i.

For the case $x = a/i$, we have

$$\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor = i \cdot \left\lfloor \frac{b}{a} i \right\rfloor \cdot \left\lfloor \frac{c}{a} i \right\rfloor .$$

To find the minimum i such that this is at least k, we can perform a binary search on i. The cases $x = b/i$ and $x = c/i$ can be handled in the same way, so we perform three binary searches corresponding to these three forms of x, and the maximum of the resulting values is the answer.

Although this is not an issue in Python, in other languages the check $i \cdot \lfloor \frac{b}{a} i \rfloor \cdot \lfloor \frac{c}{a} i \rfloor \geq k$ may cause overflow, so care must be taken. Also, the search range of i is prone to errors. In the case $x = a/i$, the optimal i lies in the range greater than 0 and at most $\max(a, k)$.

Alternative solutions

Rational numbers of the form a/i, b/i, and c/i can all be expressed in the unified form $\mathrm{lcm}(a, b, c)/i$ using the least common multiple $\mathrm{lcm}(a, b, c)$. Therefore, by restricting x to the form $x = \mathrm{lcm}(a, b, c)/i$ and performing a binary search on i, we can obtain the answer.

It is also possible to solve the problem by directly performing a binary search on x. In that case, we must compute with sufficiently high precision so that the fraction can be recovered from the result. After finding the maximum value of x in the form $x = n/2^{64}$ (where n is a positive integer), we can compute the continued fraction expansion of $n/2^{64}$ to obtain the optimal value. In any of these alternative solutions, arbitrary-precision integers are required.

Problem F: Astral Geometry

Proposer: Naoki Marumo Author: Mitsuru Kusumoto Analysis: Mitsuru Kusumoto

Let the coordinates of star i be (x_i, y_i, z_i), and write $\vec{r_i} = [x_i, y_i, z_i]^\top$. At the beginning, we are given the distances between the Earth and each star, which is equivalent to knowing $\|\vec{r_i}\|^2$. Moreover, measuring the distance between star i and star j is equivalent to determining $\|\vec{r_i} - \vec{r_j}\|^2$, which, when expanded, is $\|\vec{r_i}\|^2 + \|\vec{r_j}\|^2 - 2\vec{r_i} \cdot \vec{r_j}$, and hence is also equivalent to determining the inner product $\vec{r_i} \cdot \vec{r_j}$.

Let the $3 \times n$ matrix M be $M = [\vec{r_1} \dots \vec{r_n}]$. What we want to obtain are the values of all entries of the matrix product $S = M^\top M$. From the above discussion, at the initial stage of the interaction only the diagonal entries of S are known, and each single measurement reveals one entry of S. The objective of the problem can thus be rephrased as determining all entries of S using at most $3n$ measurements. Note that S is a symmetric matrix.

To solve the problem, we use the fact that $\operatorname{rank}(S) \le 3$ (since $\operatorname{rank}(M) \le 3$). The approach is to take a set of linearly independent basis vectors from the columns of M (that is, $\operatorname{rank}(M)$ vectors among $\vec{r_1}, \dots, \vec{r_n}$) and, based on them, determine all entries of S.

First, by the assumption in the problem statement that no star is located at the origin, we may include $\vec{r_1}$ in the basis. Next, we want to check whether there exists a vector that is linearly independent of $\vec{r_1}$. If some $\vec{r_a}$ is linearly independent of $\vec{r_1}$, then the determinant of the submatrix

$$\begin{bmatrix} S_{11} & S_{1a} \\ S_{a1} & S_{aa} \end{bmatrix}$$

is nonzero. Conversely, if the determinant is zero, then these two vectors are linearly dependent. In this way, we can find linearly independent vectors by performing $n-1$ measurements between star 1 and each of the other stars.

If we find a linearly independent vector $\vec{r_a}$, we then look for yet another linearly independent vector. Similarly, if some $\vec{r_b}$ is linearly independent of both $\vec{r_1}$ and $\vec{r_a}$, then the determinant of

$$\begin{bmatrix} S_{11} & S_{1a} & S_{1b} \\ S_{a1} & S_{aa} & S_{ab} \\ S_{b1} & S_{ba} & S_{bb} \end{bmatrix}$$

must be nonzero. By performing $n-2$ measurements between star a and each of the other stars, we can determine whether such a vector $\vec{r_b}$ exists.

Suppose we have obtained three basis vectors in this way. We then perform an additional $n-3$ measurements, determining the distances between star b and each of the remaining stars. For any $i \ne j$, consider the submatrix formed by rows $1, a, b, i$ and columns $1, a, b, j$:

$$\begin{bmatrix} S_{11} & S_{1a} & S_{1b} & S_{1j} \\ S_{a1} & S_{aa} & S_{ab} & S_{aj} \\ S_{b1} & S_{ba} & S_{bb} & S_{bj} \\ S_{i1} & S_{ia} & S_{ib} & S_{ij} \end{bmatrix}$$

The determinant of this matrix must be zero; otherwise, the rank of S would be at least 4. If we regard S_{ij} as the unknown, this determinant becomes a linear equation in S_{ij}. All other entries are already known, so by solving this linear equation we can determine the value of S_{ij}. By doing this for every pair i, j, we obtain all entries of S, and we can then convert inner products back to distances to output the answer.

Even when the number of basis vectors is only 1 or 2, we can similarly consider determinants of appropriate submatrices and solve the resulting linear equations to determine each S_{ij}. The time complexity is $O(n^2)$, and the number of required measurements is $3n - 6$.

In implementation, note that each entry of S can be as large as about $3 \times 4000^2 \simeq 5 \times 10^7$. In particular, when computing determinants of 3×3 or 4×4 submatrices of S, using 64-bit integers can cause overflow. To address this, it is effective either to use an integer type with at least 128 bits of precision or to perform computations over finite fields using several different prime moduli.

Problem G: Charity Raffle

Proposer: Tatsuya Sumiya Author: Tatsuya Sumiya Analysis: Tatsuya Sumiya

Let a_i denote the number of prizes of type i. We want to count the number of nonnegative integer sequences $A = (a_1, \dots, a_n)$ whose maximum value is at most m and which are realizable. Clearly, we must have $\sum_i a_i = k$, so we instead count the number of sequences A with this total that are not realizable.

From the rule for choosing prizes, it is impossible that only a single type of prize is obtained in a large quantity. More precisely, let r $(\leq m)$ be the maximum value of A, and suppose that the following holds.

- There is exactly one index i such that $a_i = r$ (we denote it by i_0, so $a_{i_0} = r$).
- For all i with $1 \leq i < i_0$ we have $a_i \leq r-1$, and for all i with $i_0 < i \leq n$ we have $a_i \leq r-2$.

In this situation, there is no way to increase the number of type-i_0 prizes from $r-1$ to r, so such a sequence A is not realizable. Otherwise, there exist indices i, j $(1 \leq i < j \leq n)$ such that $a_i = r$ and $r - 1 \leq a_j \leq r$. Then we can realize the sequence A by the following scenario: first the pair of types i and j is picked up $a_i + a_j$ times, and then for each l $(\neq i, j)$, the pair of types i and l is picked up a_l times.

We now describe two methods to count the sequences A that satisfy the above condition (i.e., that are not realizable).

Solution 1

Construct a nonnegative integer sequence $A' = (a'_1, \dots, a'_n)$ by setting $a'_{i_0} = r - 1$ and $a'_i = a_i$ for $i \neq i_0$. Then the following hold:

- $\sum_i a'_i = k - 1$,
- $\max A' = r - 1$.

Conversely, for any sequence A' satisfying these conditions, consider the operation that increases by 1 the rightmost element a'_i with $a'_i = r - 1$. This operation produces a sequence A that satisfies the impossibility condition. One can show that these transformations are bijective, so the problem reduces to counting such sequences A'.

Since we take the sum over r, it suffices to count the number of sequences whose total is $k - 1$ and whose maximum value is at most $m - 1$. This can be done in $O(n + k)$ time.

Solution 2

For fixed (r, i_0), the number of sequences A that satisfy the above condition can be computed by inclusion-exclusion as

$$\sum_{S \subset \{1,2,\ldots,n\} \setminus \{i_0\}} (-1)^{|S|} f(i_0, S, r), \tag{1}$$

where $f(i_0, S, r)$ denotes the number of nonnegative integer sequences $A' = (a'_1, \ldots, a'_n)$ satisfying the following conditions:

- $\sum_i a'_i = k$,
- $a'_{i_0} = r$,
- For each $i \in S$, if $i < i_0$ then $a'_i \geq r$, and if $i > i_0$ then $a'_i \geq r - 1$.

Let $c(i_0, S)$ be the number of elements of S that are less than i_0. Then $f(i_0, S, r)$ can be computed as the number of nonnegative integer sequences of length $n - 1$ whose total is $k - r - (r-1)|S| - c(i_0, S)$.

To obtain the number of sequences A that are not realizable, we sum the above expression over all r and i_0:

$$\sum_{r=1}^{m} \sum_{i_0=1}^{n} \sum_{S \subset \{1,2,\ldots,n\} \setminus \{i_0\}} (-1)^{|S|} f(i_0, S, r). \tag{2}$$

However, evaluating this expression directly would be too slow.

Here, by letting $S' = S \cup \{i_0\}$, we can rewrite it as

$$\sum_{r=1}^{m} \sum_{S' \subset \{1,2,\ldots,n\}} (-1)^{|S'|-1} \sum_{i_0 \in S'} f(i_0, S' \setminus \{i_0\}, r). \tag{3}$$

In the inner sum over $i_0 \in S'$, the value $c(i_0, S' \setminus \{i_0\})$ takes each integer from 0 to $|S'| - 1$ exactly once. Therefore, $\sum_{i_0 \in S'} f(i_0, S' \setminus \{i_0\}, r)$ is equal to the number of nonnegative integer sequences of length $n - 1$ whose total is between $k - r|S'|$ and $k - (r-1)|S'| - 1$, inclusive. This value depends only on r and $|S'|$, and it can be computed in $O(1)$ time using binomial coefficients.

Moreover, the values of $|S'|$ that need to be considered are at most $\min\left(n, \frac{k-1}{r-1}\right)$. Combining these observations, the entire problem can be solved in $O(n + k \log k)$ time.

Problem H: U-Shaped Panels

Proposer: Mitsuru Kusumoto Author: Tomoharu Ugawa Analysis: Tomoharu Ugawa

Let us closely look at the shape of a U-shape. As stated in the problem statement, a U-shape has an opening on exactly one side. When the constraint $k \geq 5$ is satisfied, no matter how we place other U-shapes, we can never completely fill this opening. Once you notice this fact, the problem is almost solved.

When $k \geq 5$ holds, the width of the opening is at least three cells. However, as long as other U-shapes do not overlap each other, no matter how we place them, they can fill at most two cells of the opening. Therefore, at least one cell of the opening always remains empty. Incidentally, in order to fill two cells, we must insert the vertical sides of two other U-shapes into the opening as in Figure 1. Thus, if there exists a U-shape whose bounding box is a $k \times k$ square region, then the orientation of that U-shape is uniquely determined. In other words, if such a U-shape

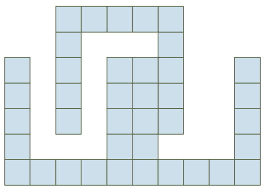
Figure 1: A placement that fills two cells of the opening

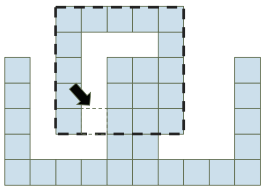
Figure 2: Determining the orientation of a U-shape

exists, then as in Figure 2, among the four sides of the square, there must be exactly one side that has some unfilled cells, and that side is the opening.

If the given configuration is feasible, then the configuration obtained by removing one U-shape is also feasible. Moreover, the configuration with no boards at all is feasible. Therefore, if we can remove U-shapes one by one until no boards remain, we can conclude that the configuration is feasible.

We just have to simulate this process of repeatedly removing U-shapes. Each of the four corners of the square bounding box of a U-shape contains a board (`#`), so when we scan the pond from the end in order, the first board we encounter must be a corner of the bounding box of some U-shape. We remove the U-shape inside that bounding box. Since the orientation of a U-shape is uniquely determined, no backtracking is necessary. During the simulation, if we ever encounter a situation where there is no U-shape inside a bounding box (this corresponds to the opening being on at least two sides, or at least one of the remaining three corners not having a board), then the configuration is not feasible.

If, after removing a U-shape, we always restart the search for the next board from the beginning, the time complexity becomes $O((nm)^2)$, which is too slow. Since we know that there are no boards before the position of the U-shape we have just removed, it is sufficient to continue searching from there for the next U-shape.

Problem I: Game of Names

Proposer: Tatsuya Sumiya Author: Soh Kumabe Analysis: Tatsuya Sumiya

Let m_A and m_B be the numbers of cells that initially contain the names of Alice and Bob, respectively, and let n_A and n_B be their final numbers of cells. Alice's winning condition is $n_A - m_A > n_B - m_B$, that is, $n_A - n_B > m_A - m_B$.

Suppose that the winner continues to write their name on the board as many as possible even after the game is decided. Considering the final state of the board, we can see that the names of Alice and Bob must alternate if we skip over empty cells (otherwise, one of them could still write their name on the board). Therefore, $n_A - n_B$ is determined by the cells at both ends as follows (both end cells are always filled).

- If Alice takes both end cells, then it is 1.
- If Bob takes both end cells, then it is -1.
- Otherwise, it is 0.

Thus, the winner of the game is determined solely by which player takes the two end cells, and the problem can be solved by an appropriate case analysis.

Problem J: ICPC Board

Proposer: Naoki Marumo Author: Naoki Marumo Analysis: Naoki Marumo

We call a board that satisfies the hypothesis in the statement a *good* board. A necessary and sufficient condition for a board to be good is that it has one of the following forms:

- Rows in which `C` and `I` alternate and rows in which `C` and `P` alternate are arranged vertically in an alternating fashion.
- Rows consisting only of `C` and rows in which `I` and `P` alternate are arranged vertically in an alternating fashion.
- The transpose of either of the above.

Once we have obtained this necessary condition, both checking and constructing can be done in time $O(nm)$. Below, we verify that this is indeed a necessary and sufficient condition. We write `I` or `P` as `X`.

If the first row consists of alternating `C` and `X`, then for the board to be good, the second row must also consist of alternating `C` and `X`. By induction, every row consists of alternating `C` and `X`.

If the first row contains `CC` or `XX`, then in those columns `CC` and `XX` appear alternately in the vertical direction. We can also see that in every other column `C` and `X` alternate.

From the above, we see that for a board to be good, it is necessary that either every row or every column consists of alternating `C` and `X`. In the following, we only consider the case where each row consists of alternating `C` and `X`. Each row is one of the following:

(a) `CXCXCX...`

(b) `XCXCXC...`

Case where both (a) and (b) appear in the board There is a place where (a) and (b) are vertically adjacent as follows:

```
CXCXCX...
XCXCXC...
```

Focusing on this part, we see that rows in which `C` and `I` alternate and rows in which `C` and `P` alternate must themselves alternate vertically. For example:

```
CICICI...
CPCPCP...
ICICIC...
CPCPCP...
```

Case where the board consists only of (a) The board looks as follows:

```
CXCXCX...
CXCXCX...
CXCXCX...
CXCXCX...
```

That is, columns consisting only of `C` and columns consisting only of `X` alternate. In columns consisting only of `X`, `I` and `P` must alternate. For example:

```
CICICP...
CPCPCI...
CICICP...
CPCPCI...
```

The case where the board consists only of (b) is similar.

Thus we have confirmed that the condition at the beginning is a necessary and sufficient condition for a board to be good.

Problem K: Membership Structure of a Secret Society

Proposer: Kazuhiro Inaba Author: Kazuhiro Inaba Analysis: Kazuhiro Inaba

Interpreting the problem in mathematical terms, we are given several formulas of the following three types:

- $x_i \in x_j$
- $x_i \notin x_j$
- $x_i = x_j \cap x_k$

and we are asked to determine whether it is possible to assign sets to the variables so that all of them are satisfied. Since in the problem statement two persons are identified if their sets of recommenders coincide, in this analysis we do not distinguish between a set of recommenders and the person it recommends.

What makes the problem difficult are the formulas of type $x_i = x_j \cap x_k$. From these formulas, it may follow that the sets denoted by some variables are actually equal. For example, from $x_1 = x_2 \cap x_3$ and $x_4 = x_3 \cap x_2$ we can derive $x_1 = x_4$. If in addition we have formulas $x_1 \in x_5$ and $x_4 \notin x_5$, then although they may look syntactically harmless at first glance, they are semantically inconsistent, and no set assignment can satisfy all formulas.

We first resolve this issue. We process the formulas of type $x_i = x_j \cap x_k$ and compute all equality relations $=$ between variables that follow from them as logical consequences.

To this end, assume for the moment that each set denoted by a variable contains some element that is different from those of all other variables:

$$c_1 \in x_1, \qquad c_2 \in x_2, \qquad \ldots, \qquad c_m \in x_m.$$

Under this assumption, together with the given formulas, we can derive further membership relations inductively as follows:

- If $c \in x_i$ and $x_i = x_j \cap x_k$, then $c \in x_j$ and $c \in x_k$ hold.
- If $c \in x_j$ and $c \in x_k$ and $x_i = x_j \cap x_k$, then $c \in x_i$ holds.

If we take care not to repeat the same derivation, as in breadth-first search on a graph, then for each c_i we can compute all such consequences in $O(n)$ time, and in total $O(n^2)$ time.

If, in this way, both $c_a \in x_b$ and $c_b \in x_a$ are derived, then every element of x_a belongs to x_b, and conversely, so we obtain $x_a = x_b$. Conversely, for any pair for which this equality condition is not derived, by assigning

$$x_i = \{\, c_a \mid c_a \in x_i \text{ has been derived}\}$$

we can explicitly construct a set assignment in which all input formulas involving $\cap$ are satisfied and all sets assigned to different variables are distinct.

Using the equivalence classes obtained by the above computation, we rename variables to the representatives of their equivalence classes. After this renaming, we may assume that different variable names always denote different sets, and checking for contradictions becomes easy. We can solve the entire problem in $O(n^2)$ time by the following steps.

1. First, process the formulas of type $x_i = x_j \cap x_k$ and derive all equality relations $=$ between variables.
2. Using the equivalence classes obtained above, rename variables to the representatives of their equivalence classes.
3. Using the formulas of type $x_i \in x_j$ and the formulas of type $x_i = x_j \cap x_k$, compute all membership relations $\in$ that can be derived. (This is done by a computation analogous to the breadth-first-search style argument used for deriving the $=$ relations.)
4. Check that there is no pair of variables for which both $\in$ and $\notin$ formulas exist.
5. Check that there is no loop formed by the $\in$ relation.

Problem L: Common Tangent Lines

Proposer: Mitsuru Kusumoto Author: Tomohiro Oka Analysis: Tomohiro Oka

We try all ways to divide the four given lines into two internal tangents and two external tangents. For each division, we translate these lines and check whether there exist two circles that have these four lines as common tangents. Among such cases, we are to find the minimum translation distance.

As a necessary condition for the existence of two circles having these four lines as common tangents, the two external tangents may be parallel, but the two internal tangents must not be parallel. Moreover, no internal tangent is parallel to any external tangent. In the following, we only consider cases that satisfy this necessary condition.

The two internal tangents must be located symmetrically with respect to the line joining the centers of the two circles. The axes of symmetry are the angle bisectors at the intersection point of the two internal tangents, and there are two such bisectors.

Similarly, the two external tangents are also located symmetrically with respect to the line joining the centers of the two circles. If the two external tangents intersect at a point, there are two angle bisectors, as in the case of the internal tangents. If the two external tangents are parallel, the two circles are sandwiched between these lines, and the axis of symmetry is a line parallel to them, at equal distance from both.

The axis of symmetry of two lines (a_i, d_i) and (a_j, d_j) can be obtained as follows, using

$$\alpha := \frac{a_i + a_j}{2}$$
$$\beta := \frac{a_i - a_j}{2}$$

as

$$x \cos\left(\frac{\pi\alpha}{180}\right) + y \sin\left(\frac{\pi\alpha}{180}\right) = \frac{d_i + d_j}{2\cos\left(\frac{\pi\beta}{180}\right)}.$$

When the two lines intersect, the other axis of symmetry is given by

$$x \cos\left(\frac{\pi(\alpha + 90)}{180}\right) + y \sin\left(\frac{\pi(\alpha + 90)}{180}\right) = \frac{d_i - d_j}{2 \sin\left(\frac{\pi\beta}{180}\right)}.$$

When the given lines are translated, these axes of symmetry are also translated. Therefore, if there exists a pair consisting of an axis of the internal tangents and an axis of the external tangents that have the same angle, then it is possible to translate the four lines so that they become common tangents.

When one internal tangent is translated, its axis of symmetry is translated by a distance proportional to that translation distance. Similarly, the translation distance of an external tangent and that of its axis of symmetry are proportional. Therefore, to make the axes of symmetry coincide with minimum total translation distance, it suffices to translate only one of the four lines.

For each of $d_1, \ldots, d_4$, we compute the translation distance (increase or decrease) required to make the axes of symmetry coincide. The minimum of these values is the required translation distance.