The 17th Zhejiang Provincial Collegiate Programming Contest

October 17

Problems

Do not open before the contest has started.

Prepared by Zhejiang University

Problem A. AD 2020

2020 is the current year and is a leap year starting on Wednesday of the Gregorian calendar, the 2020th year of the Common Era (CE) and Anno Domini (AD) designations, the 20th year of the 3rd millennium, the 20th year of the 21st century, and the 1st year of the 2020s decade.

2020 has been designated as Year of the Nurse and Midwife by the World Health Organization. The United Nations has declared 2020 as the International Year of Plant Health. 2020 has been designated as International Year of Sound by the International Commission for Acoustics.

Because there are so many unforgettable things happening in 2020, somebody has now proposed that all dates with a substring containing "202" to be designated as the Day for Disaster Reduction. We represent a date in the format YYYYMMDD (for example, 21110202), then if 202 is a substring of this date, this day is the Day for Disaster Reduction.

Please write a program to compute how many Days for Disaster Reduction are there in all the dates between $Y_1M_1D_1$ and $Y_2M_2D_2$ (both inclusive)? Note that you should take leap years into consideration. A leap year is a year that can be divided by 400 or can be divided by 4 but can't be divided by 100. A leap year has 29 days in February, instead of the normal 28 days.

Input

The input contains multiple cases. The first line of the input contains a single positive integer T ($1 \le T \le 10^5$), the number of cases.

The first and only line of each case contains six integers $Y_1, M_1, D_1, Y_2, M_2, D_2$, which are described in the problem statement above.

It's guaranteed that $Y_1M_1D_1$ is not larger than $Y_2M_2D_2$. Both $Y_1M_1D_1$ and $Y_2M_2D_2$ are between 20000101 and 99991231, and both dates are valid.

We kindly remind you that the size of the input and output of this problem can be large, so it's recommended to use a faster I/O method. For example, in C++, you can use `scanf/printf` instead of `cin/cout`.

Output

For each case, print a single line containing a single integer, the answer.

Example

standard input	standard output
3 2111 02 01 2111 02 03 2202 01 01 2202 12 31 2000 01 01 9999 12 31	1 365 44294

Note

In the first sample case, 21110202 is the only Day for Disaster Reduction.

In the second sample case, 202 is a substring of 2202, so every day in the year 2202 is the Day for Disaster Reduction!

Problem B. Bin Packing Problem

Lzw is having the Operations Research class now. Today the teacher is talking about the bin packing problem and some approximation algorithms to solve it.

In the bin packing problem, items of different volumes must be packed into a finite number of bins with a fixed capacity C in a way that minimizes the number of bins used. In computational complexity theory, it is a combinatorial NP-hard problem.

There are two classical approximation algorithms.

- **First Fit Algorithm**. Consider the items in input order and maintain a list of bins, initially empty. In each step, it attempts to place the current item in the first bin in the list that can accommodate the item. If no suitable bin is found, it appends a new bin at the end of the list and puts the item into the new bin.
- **Best Fit Algorithm**. Consider the items in input order and maintain a list of bins, initially empty. In each step, it attempts to place the current item in the best bin that can accommodate the item. "Best" means that if there are more than one bins that can accommodate the item, the bin with the least remaining space is chosen. If no suitable bin is found, a new bin is added and the item is put into the new bin.

Please help Lzw implement these two algorithms and find out how many bins will each algorithm use.

Input

The input contains multiple cases. The first line of the input contains a single positive integer T, the number of cases.

For each case, the first line of the input contains two integers n, C ($1 \le n \le 10^6$, $1 \le C \le 10^9$), the number of items and the capacity of each bin. The second line contains n integers, where the i-th ($1 \le i \le n$) integer a_i ($1 \le a_i \le C$) denotes the volume of the i-th item.

It is guaranteed that the sum of n over all cases doesn't exceed 10^6.

Output

For each case, print a single line containing two integers, denoting the number of bins used by the First Fit and the Best Fit algorithm, respectively.

Example

standard input	standard output
2 2 2 1 1 5 10 5 8 2 5 9	1 1 4 3

Problem C. Crossword Validation

A crossword is a word puzzle that usually takes the form of a square or a rectangular grid of white- and black-shaded cells. The game's goal is to fill the white cells with letters, forming words or phrases, by solving clues, which lead to the answers. The answer words and phrases are typically placed in the grid from left to right and from top to bottom. The shaded cells are used to separate the words or phrases. Crossword grids such as those appearing in most North American newspapers and magazines feature solid areas of white cells. Every letter is checked (i.e. is part of both an "across" word and a "down" word) and usually each answer must contain at least three letters. In such puzzles, shaded cells are typically limited to about one-sixth of the total. Crossword grids elsewhere, such as in Britain, South Africa, India and Australia, have a lattice-like structure, with a higher percentage of shaded cells, leaving about half the letters in an answer unchecked. For example, if the top row has an answer running all the way across, there will often be no across answers in the second row.

Crossword puzzles have a long history. The title for the world's first crossword puzzle is disputed, but most people believe the crossword puzzle is invented in the 19th century. With the emergence of personal computers, more and more crossword puzzles are distributed online instead of on newspapers and books. Software that aids in creating and solving crossword puzzles has been written since at least 1976. In this problem, you are to write a program that checks if a solution for a crossword puzzle is valid. Note that the rules of the puzzle in this problem, which is described in the next paragraph, may differ from the real-world ones.

You are given a completed crossword puzzle on a $N \times N$ grid. Each cell is either a white cell filled with a letter or a black cell. You are also given a dictionary of M distinct words, where each word has a score associated with it. A horizontal candidate word in the grid is a continuous string of letters on the same row of the grid that can't be extended. Formally, this means that the cell to the left of the leftmost cell of the candidate word either doesn't exist or is a black cell, and the same goes for the cell to the right of the rightmost cell. Vertical candidate words are defined similarly, except that the letters are on the same column. Candidate words are read from left to right for horizontal words or from top to bottom for vertical words. The crossword puzzle is valid if and only if each candidate word is present in the given dictionary. The score of a valid puzzle is the sum of the scores in the dictionary associated with each candidate word. Note that two or more candidate words may be the same. In this case, the score of that word is added multiple times accordingly. Your program must determine the score of the solution, or report that it is not valid.

Input

The input contains multiple cases. The first line of the input contains a single positive integer T, the number of cases.

For each case, the first line of the input contains two integers N, M ($1 \le N \le 1\,000, 1 \le M \le 4 \times 10^6$), the size of the grid and the number of words in the dictionary. The following N lines each contain a string of length N, where each character is either a lowercase English letter or '`#`'. If the j-th character of the i-th string is '`#`', then the cell located in the intersection of the i-th row and the j-th column is a black cell. Otherwise, this character denotes the letter filled in that cell. The following M lines each contain a non-empty string consisting of only lowercase English letters, and a positive integer, denoting a word in the dictionary and its score. It is guaranteed that those M words are pairwise distinct, the length of each word doesn't exceed N, and the score of each word doesn't exceed 10^9.

It is guaranteed that the sum of N^2 over all cases doesn't exceed 4×10^6, and the sum of the lengths of all words in the dictionary over all cases doesn't exceed 4×10^6.

Output

For each case, print a single integer in a single line, the score of the crossword solution given in the input if it is valid. If the solution is not valid, print -1 instead.

Example

standard input	standard output
2 2 4 ab #d ab 1 a 2 d 3 bd 4 2 4 ab c# ab 5 ca 2 b 6 c 7	10 -1

Problem D. Dividing the Points

In Spectral Clustering, data points are represented as vertices of an edge-weighted undirected graph G. Let n and m be the number of vertices and edges in G respectively. The vertices are numbered from 1 to n. Large edge weights mean that the adjacent vertices are very similar; small weights imply dissimilarity.

Clustering can be viewed as partitioning vertices into two non-empty disjoint groups A and B such that

$$Ncut(A,B) = \frac{cut(A,B)}{\sum_{x\in A} d_x} + \frac{cut(A,B)}{\sum_{x\in B} d_x}$$

is minimized, where $cut(A,B)$ is the total edge weight of edges connecting vertices from different groups, and d_x is the total edge weight of edges adjacent to the x-th vertex.

In this problem, a graph G with n vertices and m edges will be given, where each vertex corresponds to a data point on the 2D plane. You need to find a straight line to divide these n points into two non-empty groups A and B such that $Ncut(A,B)$ is minimized. Note that the straight line shouldn't pass through any data points.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 200$), the number of cases.

For each case, the first line of the input contains two integers n and m ($2 \le n \le 1\,500$, $n-1 \le m \le 5\,000$), denoting the number of vertices and the number of edges.

For the next n lines, the i-th line contains two integers x_i and y_i ($1 \le i \le n,\ 0 \le x_i, y_i \le 10\,000$), denoting a data point at (x_i, y_i). It is guaranteed that no two points coincide.

For the next m lines, the i-th line contains three integers u_i, v_i and w_i ($1 \le i \le m,\ 1 \le u_i, v_i \le n,\ u_i \ne v_i$, $1 \le w_i \le 100$), denoting a bidirectional edge connecting the u_i-th vertex and the v_i-th vertex. It is guaranteed that there won't be multiple edges between the same pair of vertices, and it is also guaranteed that the graph is connected.

It is guaranteed that the sum of n over all cases does not exceed 1 500, and the sum of m over all cases does not exceed 5 000.

Output

For each case, print a single line containing a single irreducible fraction of the form `p/q`, denoting the minimum value of $Ncut(A,B)$. Formally, your answer must satisfy $p, q > 0$ and $\gcd(p,q) = 1$, where $\gcd(a,b)$ denotes the greatest common divisor of a and b.

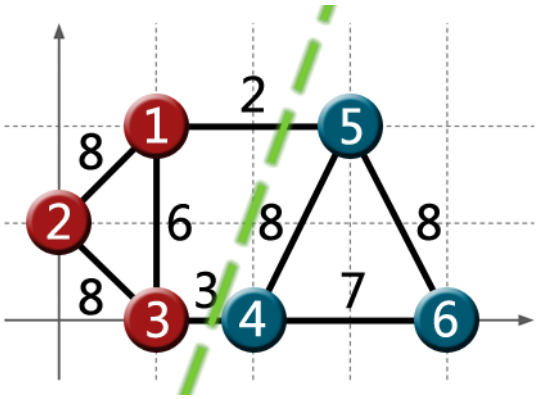
Example

standard input	standard output
1 6 8 1 2 0 1 1 0 2 0 3 2 4 0 1 2 8 1 3 6 2 3 8 1 5 2 3 4 3 5 4 8 5 6 8 4 6 7	500/2499

Note

The following is an optimal solution for the sample case, which is illustrated below.

- $A = \{1, 2, 3\}$, $B = \{4, 5, 6\}$.
- $cut(A, B) = 2 + 3 = 5$.
- $\sum_{x \in A} d_x = d_1 + d_2 + d_3 = 16 + 16 + 17 = 49$.
- $\sum_{x \in B} d_x = d_4 + d_5 + d_6 = 18 + 18 + 15 = 51$.
- $Ncut(A, B) = \frac{5}{49} + \frac{5}{51} = \frac{500}{2\,499}$.



An illustration of the sample case.

Problem E. Easy DP Problem

lqybzx is weak in dynamic programming. One day, his roommate gives him a sequence $b_1, b_2, \ldots, b_m$ and he asks lqybzx to do a dynamic programming. There are $(m+1)^2$ states $dp[i][j]$ where $0 \le i, j \le m$, and the transition equation is:

$$dp[i][j] = \begin{cases} 0 & (i = 0) \\ i^2 + dp[i-1][j] & (i > 0, j = 0) \\ i^2 + \max(dp[i-1][j], dp[i-1][j-1] + b[i]) & (i > 0, j > 0) \end{cases}$$

lqybzx can't solve this easy DP problem, so he turns to you for help. You are given a sequence $a_1, a_2, \ldots, a_n$ and q queries. In the i-th query, you are given three integers l_i, r_i and k_i. Please write a program to figure out the value of $dp[m_i][k_i]$ when lqybzx's roommate chooses $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ as the initial sequence b. Here, $m_i = r_i - l_i + 1$ and $b_j = a_{l_i+j-1}$ for every j satisfying $1 \le j \le m_i$.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 100$), the number of cases.

For each case, the first line of the input contains a single integer n ($1 \le n \le 10^5$), the length of the sequence. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le i \le n,\ 1 \le a_i \le 10^6$). The third line contains a single integer q ($1 \le q \le 10^5$), the number of queries. Each of the next q lines contains three integers l_i, r_i and k_i ($1 \le i \le q,\ 1 \le l_i \le r_i \le n, 1 \le k_i \le r_i - l_i + 1$), denoting a query.

It is guaranteed that the sum of all n over all cases does not exceed 5×10^5, and the sum of q over all cases does not exceed 5×10^5.

Output

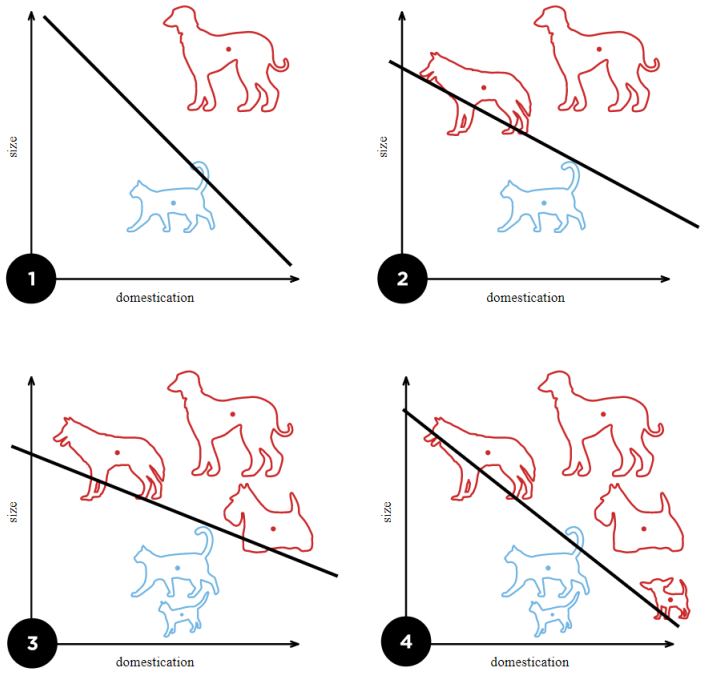
For each case, print q lines, where the i-th ($1 \le i \le q$) line contains a single integer, the value of $dp[m_i][k_i]$.

Example

standard input	standard output
1 5 1 2 3 4 5 3 1 3 2 1 5 5 3 3 1	19 70 4

Problem F. Finding a Sample

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function that can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector.



A diagram showing the training process of a perceptron. Source: Wikipedia, CC BY-SA 4.0.

In the modern sense, the perceptron is an algorithm for learning a binary classifier called a threshold function: a function that maps its input $\mathbf{x}$ (a real-valued vector) to an output value $f(\mathbf{x})$:

$$f(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

$$sign(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

where $\mathbf{w}$ is a vector of real-valued weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product $\sum_{i=1}^{m} w_i x_i$, where m is the number of inputs to the perceptron, and b is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value.

Now, you have two perceptrons f_1 and f_2. You want to find a sample $\mathbf{x}$ that makes the classification results of the two perceptrons different. Formally, this means $f_1(\mathbf{x}) \cdot f_2(\mathbf{x}) < 0$.

Input

The input contains multiple cases. The first line of the input contains a single positive integer T ($1 \le T \le 10$), the number of cases.

For each case, the first line of the input contains a single integer N ($1 \le N \le 200$), the number of inputs to the perceptron. The second line contains $N+1$ integers , where the i-th ($1 \le i \le n$) integer denotes w_i, and the last integer denotes b, of the first perceptron. The third line contains $N+1$ integers describing the second perception in the same format as the first.

It is guaranteed that the parameters (w_i and b) of the perceptrons are all integers in the range $[-100, 100]$.

Output

For each case, if there is at least one sample satisfying the requirements, output one such sample. Print a single line containing N real numbers, where the i-th ($1 \le i \le N$) number denotes x_i of the sample. If not, print a single string "`No`" (without quotes).

Please note that each number you output must be in the range $[-10000, 10000]$. Also, it should have no more than 11 decimal places, otherwise your solution may not be accepted even if it is correct, due to the limited precision of the checker program (Generally, you should try to print as few decimal places as possible to avoid precision issues. The checker program is written in C++ and uses the `double` data type). It is guaranteed that if a solution exists, there is a solution that meets the conditions above.

Example

standard input	standard output
2 2 1 -1 1 1 -1 -1 2 1 -1 0 2 -2 0	0 0 No

Problem G. Gliding

Link loves cool challenges, such as shooting arrows through small holes, sliding down a 3000-metre-high snow mountain and playing music with a huge piece of leaf. Now he is trying a new challenge of gliding through a lake without sinking into it.

Link's world is a three-dimensional space, let's define a Cartesian coordinate system (x, y, z), where z is the height. The water surface is the plane $z = 0$. Link starts at position $(s_x, s_y, 0)$ and the destination is at $(t_x, t_y, 0)$. He fails the challenge if, at any time, his height z is strictly less than 0 ($z < 0$), which means he has fallen into the water.

In Link's world, unlike Earth, things in the air fall freely with a constant speed of v_f. Under such a fast falling speed Link cannot control his body and will fall down vertically with zero horizontal speed. Luckily, Link has a paraglider to slow down the falling. Once he opens the paraglider, the falling speed becomes v_p ($v_p < v_f$) and Link is able to move horizontally in any direction with a speed of at most v_h. Formally, Link's speed in the x direction v_x and in the y direction v_y should satisfy $\sqrt{v_x^2 + v_y^2} \le v_h$. Link can close and open the paraglider any number of times, and he can do so at any time.

Even better for Link, there are $n + 1$ strange wind caves in the lake, producing wind blowing upwards that helps Link to go higher. Those wind caves are numbered from 0 to n, and are described by the tuples $(x_i, y_i, v_i),\ i = 0, 1, \ldots, n$. The position of the i-th wind cave is $(x_i, y_i, 0)$, and the speed of its upwards wind is v_i. Link must be on the vertical line $[x = x_i, y = y_i]$ and his paraglider must be open to be affected by the i-th cave. If Link's paraglider is open while on top of the i-th wind cave, he will rise with speed $v_i - v_p$ if $v_i > v_p$, or fall with speed $v_p - v_i$ if $v_p > v_i$, or have a zero vertical speed if $v_p = v_i$. The location where Link starts is the 0-th wind cave, which means $s_x = x_0, s_y = y_0$. Link carefully chooses this location such that $v_0 > v_p$, which means he can rise as high as he wants initially. Therefore, it can be shown that Link can always reach the destination without falling into the lake, given sufficient time.

The time that it takes for Link to open and close the paraglider, and the time it takes to change his speed can be neglected. Link wants to know the minimum time needed to arrive at the destination.

Input

The input contains multiple cases. The first line of the input contains a single positive integer T, the number of cases.

For each case, the first line contains four integers s_x, s_y, t_x, t_y ($-10^4 \le s_x, s_y, t_x, t_y \le 10^4$), describing the coordinates of the starting location and the destination. The second line contains three integers $v_f, v_p, v_h (0 < v_p < v_f \le 10^4,\ 0 < v_h \le 10^4)$ denoting the free falling speed, the falling speed with the paraglider open and the maximum horizontal speeed respectively. The third line contains a single positive integer n ($n \le 4000$), the number of wind caves excluding the one at the start location. The following $n + 1$ lines each contains three integers x_i, y_i, v_i ($-10^4 \le x_i, y_i \le 10^4,\ 0 < v_i \le 10^4,\ i = 0, 1, \ldots, n$), denoting the position of each cave and the speed of its wind. It is guaranteed that no two caves have the same position, and there are no wind caves at the destination location.

It is guaranteed that $x_0 = s_x,\ y_0 = s_y,\ v_0 > v_p$ and that the sum of n over all cases doesn't exceed 10^4.

Output

For each case, print a single number in a single line, the minimum time needed to arrive at $(t_x, t_y, 0)$ without falling into the lake. Your answer will be accepted if and only if its absolute or relative error does not exceed 10^{-9}.

Example

standard input	standard output
2 0 0 3 4 5 4 1 1 0 0 5 3 0 6 0 0 3 4 5 4 1 1 0 0 5 3 0 7	25.000000000000000 24.333333333333332

Note

In the first example, Link opens the paraglider at $(0, 0, 0)$ and waits 20 seconds so that he rises to 20 units high. Then he moves straight to $(3, 4, 0)$ and it costs 5 seconds to arrive there.

In the second example, Link rises to 12 units high at first and moves straight to (3,0,0), which costs him 15 seconds. Then he rises to 16 units high and moves straight to the destination which costs $\frac{28}{3}$ seconds.

Problem H. Huge Clouds

It's a starry night and DreamGrid is counting stars. Unfortunately, it becomes cloudy and many stars are hidden from view. DreamGrid wants to go out of the shadow and find a place where at least one star can be seen. He wants to know the total area of the shadow first.

There are n stars and m clouds in the sky. Formally, each star is represented by a point (x_i, y_i) $(1 \le i \le n)$ in a two-dimensional plane. The i-th $(1 \le i \le m)$ cloud is represented by a segment $(u_{i1}, v_{i1}) - (u_{i2}, v_{i2})$, where (u_{i1}, v_{i1}) and (u_{i2}, v_{i2}) are the two endpoints of the segment.

For a viewpoint $(x, 0)$ on the x-axis, DreamGrid considers it to be in shadow if none of the stars can be seen there. A star can be seen at a viewpoint if the segment connecting the viewpoint and the star doesn't intersect with any cloud (including the endpoints of a cloud). Note that if a cloud (including its endpoints) goes through a star, the star can't be seen anywhere.

It's tiring to go through the whole x-axis and calculate the total shadow area. Can you tell DreamGrid the total area (length) covered by shadow on the x-axis?

Input

The input contains multiple cases. The first line of the input contains a single integer T $(1 \le T \le 500)$, the number of cases.

For each case, the first line of the input contains two integers n and m $(1 \le n, m \le 500)$, the number of stars and clouds. The following n lines describe the positions of the stars. The i-th $(1 \le i \le n)$ line contains two integers x_i and y_i $(-10^4 \le x_i \le 10^4,\ 1 \le y_i \le 10^4)$, the position of the i-th star. The following m lines describe the positions of the clouds. The i-th $(1 \le i \le m)$ line contains four integers $u_{i1}, v_{i1}, u_{i2}, v_{i2}$ $(-10^4 \le u_{i1}, u_{i2} \le 10^4,\ 1 \le v_{i1}, v_{i2} \le 10^4, (u_{i1}, v_{i1}) \ne (u_{i2}, v_{i2}))$, the positions of the endpoints of the i-th cloud.

It's guaranteed that the number of cases where $\max(n, m) \ge 100$ doesn't exceed 5. Please note that multiple stars can be in the same position, and the clouds can intersect and overlap with each other.

Output

For each case, print a single line containing a single real number, denoting the total area (length) of shadow on the x-axis. If the answer is larger than 10^9, print -1 instead.

Your answer will be accepted if and only if the absolute or relative error does not exceed 10^{-4}.

Example

standard input	standard output
8 1 2 0 3 -2 1 -1 1 2 1 1 1 1 2 0 3 -2 1 -1 1 1 2 2 1 2 2 10000 100 -10000 100 -10000 50 -3000 50 10000 50 3000 50 2 2 0 3 -1 10 3 2 0 1 0 1 -1 10 1 1 0 2 1 1 3 2 1 1 0 10000 -10000 9999 10000 9999 1 1 0 10000 -9999 9999 9999 9999 1 1 0 10000 -10000 1 9999 2	3.0000000000 1.5000000000 8000.0000000000 -1 -1 200000000.0000000000 199980000.0000000000 20002.0003000500

Note

For the first sample case, the range of the shadow is $[-3, -1.5] \cup [1.5, 3]$.

For the fourth sample case, the second star is covered by the second cloud, so it can't be seen anywhere.

Problem I. Invoking the Magic

BaoBao is a lazy boy. He has n pairs of socks in different colours and he washes them once a month. In the washing machine, these socks will be mixed.

Because there are too many socks that need to be paired, BaoBao divides the whole sock pairing process into two stages.

In the first stage, BaoBao randomly distributes the socks in pairs. Then in the second stage, BaoBao repeats the following operation until all the socks are paired: BaoBao chooses some of the pairs of socks, puts them into his magic washing basin, and uses his magic. If the socks in the magic washing basin can be paired perfectly when he uses his magic, the magic washing basin will pair them for BaoBao automatically. However, if they can't (which means there is at least one sock whose colour is unique in the basin), the magic basin will explode and BaoBao must not let this happen.

BaoBao's magic is limited, after the first stage, he needs to know the minimum capacity of the magic washing basin that he needs to pair all the socks successfully. The capacity of the basin is the maximum number of pairs of socks BaoBao can put in the magic washing basin at one time.

Input

The input contains multiple cases. The first line of the input contains a single positive integer T ($1 \le T \le 10$), the number of cases.

For each case, the first line of the input contains a single integer n ($1 \le n \le 10^5$), the number of pairs of socks.

For the following n lines, the i-th ($1 \le i \le n$) line contains two integers a_i and b_i ($1 \le a_i, b_i \le 2^{30}$), denoting the colours of the two socks in the i-th pair after the first stage. It is guaranteed that for each sock, there is exactly one other sock of the same colour.

Output

For each case, print a single line containing a single integer, the minimum capacity of the magic washing basin.

Example

standard input	standard output
1 5 1 2 2 3 1 3 4 5 4 5	3

Note

In the sample, BaoBao can first put these three pairs of socks: {1,2}{2,3}{1,3} in the magic washing basin and then they will be paired into {1,1}{2,2}{3,3}. Then, he can put {4,5}{4,5} in the magic washing basin and they will be paired into {4,4}{5,5}. Therefore, with a capacity of 3 it is possible to pair all the socks successfully. It can be shown by brute-force that this is impossible with a capacity of 2.

Problem J. Just an Old Problem

Chenjb came across a tricky problem in the 2018 ICPC Asia Xuzhou Regional Contest, which was a problem about the minimum spanning tree (MST).

A minimum spanning tree, or minimum weight spanning tree, is a subset of edges from an edge-weighted undirected graph, which forms a tree with the minimum possible total edge weight that connects all the vertices together.

In that problem, Chenjb was required to calculate the summation of total edge weights of all MSTs for a given graph, which obviously equals to the product of the total edge weight in an MST and the total number of different MSTs. Note that two spanning trees are different if and only if their edge sets are different.

It took Chenjb about an hour to solve that problem using Kruskal and Matrix-tree Theorem. Now Chenjb believes that you can solve it faster than him. You will be given a connected undirected graph G with n vertices and m edges, the vertices of which are labelled by $1, 2, \ldots, n$. Interestingly, there are no simple paths passing through 8 vertices in G. You need to calculate the summation of total edge weights of all MSTs for the given graph. To decrease the size of the output, you only need to output the answer modulo 2^{30}.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \leq T \leq 5\,000$), the number of cases.

For each case, the first line of the input contains two integers n and m ($2 \leq n \leq 50\,000$, $1 \leq m \leq 100\,000$), denoting the number of vertices and the number of edges in the graph.

For the next m lines, the i-th line contains three integers u_i, v_i and w_i ($1 \leq i \leq m, 1 \leq u_i, v_i \leq n$, $u_i \neq v_i$, $1 \leq w_i \leq 10\,000$), denoting a bidirectional edge connecting the u_i-th vertex and the v_i-th vertex with weight w_i. It is guaranteed that there won't be multiple edges between the same pair of vertices, and it is also guaranteed that it is impossible to find any simple path passing through 8 vertices.

It is guaranteed that the sum of n over all cases does not exceed 50 000, and the sum of m over all cases does not exceed 100 000.

Output

For each case, print a single line containing a single integer, denoting the answer modulo 2^{30}.

Example

standard input	standard output
2 3 3 1 2 4 1 3 5 2 3 6 3 3 1 2 4 1 3 4 2 3 4	9 24

Problem K. Killing the Brute-force

Chenjb is the task author of the 71-st Zhejiang Provincial Collegiate Programming Contest. He created an NP-Hard problem on a graph with n vertices, the intended solution of which is DFS with branching and pruning. Chenjb is an experienced task author, he knows that the constraints can't be too tight, so he will set the time limit t to be $3x$, where x is the running time of the intended solution on the slowest test case.

Chenjb heard that the contest would be unfortunately held on Potato OJ, whose hardware is not so good. To protect Potato OJ from being overloaded and the contest being a failure, Chenjb plans to cut down the input size of his task. He has tested the running time of the intended solution and the brute-force solution for various values of n. Note that for simplicity, we assume the running time of the solutions on a test case only depends on the value of n.

Please help Chenjb find the smallest value of n such that $1 \le n \le m$, and the brute-force solution won't pass the problem. Note that if the time limit is t, we assume that a solution will pass if and only if its running time on each test case is smaller than or equal to t.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 50$), the number of cases.

For each case, the first line of the input contains a single integer m ($1 \le m \le 50$), denoting the upper bound of n.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le 100, a_i \le a_{i+1}$), the i-th of which denotes the running time of the intended solution when $n = i$.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 100, b_i \le b_{i+1}$), the i-th of which denotes the running time of the brute-force solution when $n = i$.

Output

For each case, print a single line containing a single integer denoting the minimum possible value of n. If there is no solution, print -1 instead.

Example

standard input	standard output
2 4 1 2 7 20 2 5 30 40 3 2 3 3 5 7 8	3 -1

Problem L. List of Products

In number theory, the fundamental theorem of arithmetic, also called the unique factorization theorem or the unique-prime-factorization theorem, states that every integer greater than 1 either is a prime number itself or can be represented as a product of prime numbers and that this representation is unique. For example, $720 = 2^4 \times 3^2 \times 5^1$.

According to the unique factorization theorem, we can represent any integer x as

$$x = 2^{e(x,2)} \times 3^{e(x,3)} \times 5^{e(x,5)} \times \cdots \times p_i^{e(x,p_i)} \times \ldots$$

where p_i is the i-th smallest prime number. Now let's define a different way to compare two integers x and y ($x, y \geq 2$):

- When $x = y$, we consider x to be equal to y.
- When $x \neq y$, let's find the smallest positive integer i such that $e(x, p_j) = e(y, p_j)$ holds for $j = 1, 2, \ldots, i-1$ and $e(x, p_i) \neq e(y, p_i)$, then we consider x to be smaller than y if and only if $e(x, p_i) < e(y, p_i)$, otherwise we consider x to be greater than y.

You will be given n positive integers $a_1, a_2, \ldots, a_n$ and m positive integers $b_1, b_2, \ldots, b_m$. Let's consider all $n \times m$ possible pairwise products of those integers $a_i \times b_j$ ($1 \leq i \leq n$, $1 \leq j \leq m$) and print them in non-decreasing order according to the comparison rules described above. Your task is to find the k-th number in the printed list. Note that the elements in the list are numbered from 1 to $n \times m$.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \leq T \leq 2\,000$), the number of cases.

For each case, the first line of the input contains three integers n, m and k ($1 \leq n, m \leq 10^5$, $1 \leq k \leq n \times m$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq i \leq n,\ 2 \leq a_i \leq 10^6$).

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq i \leq m,\ 2 \leq b_i \leq 10^6$).

It is guaranteed that the sum of n over all cases does not exceed 10^5, and the sum of m over all cases does not exceed 10^5.

Output

For each case, print a single line containing a single integer, denoting the k-th number in the list.

Example

standard input	standard output
1 3 3 6 7 5 7 3 2 7	15

Note

The list of the sample test is `[49,49,35,21,21,15,14,14,10]`.