Problem A. Station Module Hierarchy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	256 mebibytes

The engineers of an orbital station are designing a hierarchy of modules of size n. The main command module is numbered 1. To keep the structure stable, they decided to calculate the number of *k-module calibration groups* in the hierarchy.

A *station module hierarchy* of size n is defined as a rooted tree with n vertices, rooted at vertex 1, where each vertex has **strictly fewer** direct children than its parent (except the root that doesn't have any parent).

A *k-module calibration group* is a pair (v, S) where:

- $v \in V$ is a vertex of the hierarchy (the *center*),
- $S \subset V \setminus \{v\}$ is a set of vertices of size $k-1$,
- there exists an integer d such that for every $u \in S$, $\text{dist}(v, u) = d$.

The distance $\text{dist}(u, v)$ is defined as the number of edges on the simple path between vertices u and v.

In other words, a k-module calibration group consists of one vertex v (the center) and $k-1$ other vertices that are all at the same distance from v.

Your task is to compute the number of k-module calibration groups in the given hierarchy.

Since the answer may be large, output it modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

Each test case begins with a line containing two integers n and k ($2 \le n \le 5 \cdot 10^5$, $1 \le k \le n$): the number of vertices in the station module hierarchy and the size of the calibration group.

The next line contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i denotes the parent of vertex i in the station module hierarchy.

The sum of n over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single integer: the number of k-module calibration groups modulo 998 244 353.

Example

standard input	*standard output*
3 3 3 1 1 5 2 1 1 2 3 6 3 1 1 1 2 2	1 20 16

Note

In the first test case, the only existing calibration group is centered at vertex 1, and the corresponding set is $S = \{2, 3\}$.

Problem B. Energy Cell Selection

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

The orbital station is preparing a new power module for calibration. The engineers have n energy cells in storage. Each cell has two associated values: a_i and b_i. You can select any subset S of exactly k cells and install them together to obtain a calibration score calculated as:

$$\left(\sum_{i \in S} a_i\right) \cdot \left(\sum_{i \in S} b_i\right)$$

Your task is to determine the maximum possible calibration score that can be achieved by installing exactly k energy cells.

It is additionally known that the cell storage is *safety-certified*, meaning no energy cell has both a_i and b_i simultaneously too high. Specifically, for every cell, $\min(a_i, b_i) \leq 100$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

Each test case begins with a line containing two integers n and k ($1 \leq n \leq 10^5$, $1 \leq k \leq \min(n, 5)$): the total number of energy cells and the number of cells to install for a single calibration.

Each of the next n lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq 10^9$), representing the values of an energy cell. For every cell, $\min(a_i, b_i) \leq 100$.

The total number of energy cells across all test cases does not exceed 10^5.

Output

For each test case, print a single integer: the maximum possible calibration score that can be obtained by selecting exactly k energy cells.

Examples

standard input	*standard output*
1 5 5 1 1 2 2 3 3 4 4 5 5	225
1 6 5 1 1 2 6 3 5 4 4 5 3 6 2	400

Note

In the first test case, we use all the energy cells. The score is $(1+2+3+4+5)\cdot(1+2+3+4+5) = 15^2 = 225$.

In the second test case, we have to leave out one energy cell. The set with cells $[2, 3, 4, 5, 6]$ has the value 400.

Station note: the two values of a cell may represent any pair of independent engineering measurements. Only the formula above and the safety-certification condition matter for choosing the cells.

Problem C. Shuttle Pilot Rating

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Shuttle pilots on an orbital station periodically fly timed training routes between docking beacons. After joining the station flight program, Kostya wanted to understand the local shuttle-pilot rating called the "Crew Navigation" score (CN). Here is a moderately simplified model of this rating system.

In our model of the shuttle pilot rating system, there are the following elements:

- A total of n pilots, numbered 1 through n. Kostya is pilot number 1.
- A list of m shuttle trials in the order of their occurrence. Trial j occurs on day d_j, and has a flag $s_j \in \{0, 1\}$ indicating "special" status (for example, orbital qualification finals).
- A list of k_j pilots who took part in trial j. The i-th listed pilot is pilot number a_i who completed the route with time (result) $R_{a_i,j}$. Some pilots may be disqualified (DSQ): they appear in the list with $R_{a_i,j} = 0$. Except for the DSQ case, the smaller the $R_{a_i,j}$ value is, the better the performance of a pilot.
- Each pilot i has a base rating $CN_{i,\text{base}}$ (their rating before any trial they flew in).
- Each day is attributed to a year, each year being 365 days long. So, days $[0; 365)$ form the first year, days $[365; 730)$ form the second year, and so on.
- During each year, there exists a normalization coefficient K_{norm}. During the first year, $K_{\text{norm}} = 10\,000$.

At each moment of time t, pilot i has a defined rating $CN_{i,t}$. Ratings evolve over time according to the following process.

1. For each trial on day d_j:
   (a) Fetch each listed pilot's rating from 15 days ago: $CN_{i,(d_j-15)}$.
   (b) Calculate $F_j = |\{i : R_{i,j} \neq 0\}|$, the number of pilots who finished the route without being disqualified.
   (c) Compute the *RaceScore*:

$$N_{\text{infl}} = \max\bigl(3, \lfloor 2F_j/3 \rfloor\bigr),$$

$$\text{and if } F_j \geq N_{\text{infl}}, \text{ then } \mathit{RaceScore}_j = \left\lfloor \frac{1}{N_{\text{infl}}} \sum_{p=1}^{N_{\text{infl}}} \bigl(R_{Pos_p,j} \cdot CN_{Pos_p,(d_j-15)}\bigr) \cdot \frac{10\,000}{K_{\text{norm}}} \right\rfloor$$

   where $Pos_1, \ldots, Pos_{F_j}$ lists non-disqualified pilots in increasing order of $R_{i,j}$ (ties are broken in a way that a pilot with a higher CN value is earlier in rankings). If $F_j < N_{\text{infl}}$, the trial is unrated and no performances are recorded.
   (d) In case the trial is rated, each of the k_j pilots who took part gets a record of participation. Pilot a_i has the following performance value:

$$\mathit{Performance}_{a_i,j} = \begin{cases} 0 & \text{if } R_{a_i,j} = 0 \text{ (DSQ)}, \\ \left\lfloor \frac{\mathit{RaceScore}_j}{R_{a_i,j}} \right\rfloor & \text{otherwise.} \end{cases}$$

   This performance (along with trial date d_j and special flag s_j) is added to this pilot's history.

2. At any day t, a pilot i's current $CN_{i,t}$ is defined by their recorded performances. Let H_i be the multiset of all performances in days $(t-365, t]$.

   (a) If $H_i = \varnothing$, then:
   $$CN_{i,t} = \begin{cases} CN_{i,\text{base}} & \text{if pilot } i \text{ has no recorded performances for all times in } (-\infty, t], \\ 0 & \text{otherwise.} \end{cases}$$

   (b) Otherwise:

   - Sort H_i in non-increasing order.
   - Let $h = \min(\max(\lfloor 0.4 \cdot |H_i| \rfloor, 4), |H_i|)$. Then:
     i. Take the h largest **non-zero** scores into multiset S.
     ii. Construct a multiset $S' = S$. If $|S| > 1$, drop the very best score from S' unless there exists a special trial ($s_j = 1$) with the very best score. If $|S| \le 1$, S' is not changed under any circumstances.
     iii. Compute *avg*: the average of the scores in S'. In case $|S'| = 0$, let $avg = 0$.
     iv. If $|S| < 4$, there will be a penalty of 2% per missing trial (for example, 3 trials $\to$ 2%, 2 trials $\to$ 4%, 1 trial $\to$ 6%). If there were DSQ trials in H_i, this penalty should be reduced by 2% regardless of the number of DSQ trials. We define the effect of x% penalty on number y as $\frac{100-x}{100} \cdot y$.
   - The floored *avg* after applying the penalty is $CN_{i,t}$. Please note that this value is an integer, and was floored from the corresponding rational number exactly once in the end.

3. Additionally, whenever $t \equiv 0 \pmod{365}$ for all years but the first, a recalibration is performed (before any of the trials on that day, if any):
   - Let $M = \max_i CN_{i,t-1}$.
   - If $M = 0$, $K_{\text{norm}} = 10\,000$.
   - Otherwise, $K_{\text{norm}} = M$.

The Station Flight Board wants as many pilots joining the shuttle trials as possible. Kostya has a hunch that this complicated system has the opposite effect.

So Kostya wants to show the Station Flight Board that by omitting some trials he actually flew from his official log, he could end up with different ratings after the last trial of the season. In other words, he is interested in different possible values of CN_{1,d_m} after all updates have been processed if he was not listed in the logs of some trials. As each shuttle route is evaluated independently, all $R_{i,j}$ do not influence each other.

Given the full data of n pilots, m shuttle trials, and Kostya's personal flight log, compute every distinct final rating CN_{1,d_m} achievable by choosing a subset of the trials Kostya flew (including an empty subset). Output the ratings in strictly decreasing order.

Input

The first line contains two integers n and m ($1 \le n, m \le 1000$): the number of pilots and the number of shuttle trials.

The second line contains n integers $CN_{i,\text{base}}$ ($1 \le CN_{i,\text{base}} \le 12\,000$): base ratings for the pilots.

The third line contains m integers $d_1, \ldots, d_m$ ($1 \le d_i \le 2000$, $d_i \le d_{i+1}$): days of the shuttle trials.

The fourth line contains m integers $s_1, \ldots, s_m$ ($0 \le s_i \le 1$): special flags for the trials.

Then follow n blocks, one per pilot $i = 1, \ldots, n$:

- A line with an integer n_i ($0 \le n_i \le 10$), the number of trials that pilot i flew.
- Each of the next n_i lines contains two integers b_j and R_{i,b_j} ($1 \le b_j \le m$, $600 \le R_{i,b_j} \le 9000$ or $R_{i,b_j} = 0$): trial number and result in seconds. For each pilot, all b_j are distinct.

Output

On the first line, print a single integer c: the number of distinct achievable ratings. On the next c lines, print the achievable values of CN_{1,d_m} in decreasing order, one per line.

Examples

standard input	*standard output*
3 4 6200 7800 10200 90 100 300 464 1 1 1 1 4 1 1950 2 1940 3 900 4 1800 4 1 1920 2 1800 3 844 4 1800 4 1 1860 2 1670 3 769 4 1800	15 9156 8120 8016 7981 7920 7828 7719 7433 6983 6965 6960 6899 6600 6200 0
6 2 10118 10485 10427 9158 9307 10044 356 365 0 0 2 1 1105 2 791 1 2 960 1 1 929 1 1 841 0 2 1 1043 2 1195	4 11360 10118 8068 7900

Note

We will explain two of the achievable CN values for the first example. Omitting all 4 trials would result in $CN_{1,d_m} = CN_{1,\text{base}} = 6200$ for Kostya. If Kostya keeps only the first trial on day 90 in his official log, by the day 464 it would be gone from the performances being considered, thus his $CN_{1,d_m} = 0$. Other possible results are given in the output.

Problem D. Gateway Code

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The orbital station has a large system of service gateways, docking corridors, and maintenance airlocks. To make the access system easy to audit, every gateway is assigned a numeric code.

A new gateway is being connected to the station network, but all simple codes have already been used. The chief engineer decides to generate its code from today's access key k and wants to choose some number n as the new gateway code.

The gateway controller will accept a code n if it has the following properties:

- the number n is divisible by k,
- the sum of all digits in n is k.

You already found valid gateway codes for small values of k, so now you need a program to work with larger keys.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

Each test case is given on a single line containing one integer k ($1 \le k \le 10^5$).

The sum of k over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n ($1 \le n \le 10^{20k}$). In case there are multiple valid gateway codes, output any one of them.

Example

standard input	*standard output*
4 2 8 18 84	2 8 198 84848484848484

Problem E. Station Navigation with Closed Sectors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

A lockdown drill has just started on the orbital station. Some maintenance sectors are sealed, but service corridors between them remain open. The navigation system must route a shuttle technician from the north-west airlock to the south-east docking gate as efficiently as possible.

The station layout is represented as a grid formed by the Cartesian product of two layout axes, A and B. The positive direction along axis A goes **right**, and along axis B goes **down**.

Each axis is partitioned into segments that alternate between:

- Closed sector spans: these are sealed areas that cannot be traversed,
- Service corridors: gaps between the closed sector spans, where movement is allowed.

The segments are numbered starting from 1, and the pattern alternates: segments with odd indices are closed sector spans, and even indices are corridors. The number of segments is always odd.

Additionally, no service corridor is wider than any dimension of any closed sector span. Formally, among the segments on both axes, the smallest segment with an odd index is no less than the largest segment with an even index.

We define a point (x, y) to be *inside a closed sector* if it lies within a horizontal closed sector span (from axis A) **and** within a vertical closed sector span (from axis B). These closed sectors are impassable. You may move freely elsewhere, as long as your path does not pass through the interior of any closed sector. Touching the edges or corners is allowed.

The technician wants to get from the top-left corner of the grid to the bottom-right corner, moving through open areas only, and using Euclidean distance. Help the navigation system find the shortest possible path!

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

Each test case is described as follows:

- A line containing two integers n and m ($1 \le n, m \le 2000$): the number of closed sector spans along the horizontal and vertical axes, respectively.
- A line with n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$): the widths of the horizontal closed sector spans.
- A line with $n-1$ integers $b_1, b_2, \ldots, b_{n-1}$ ($1 \le b_i \le 10^6$): the widths of the horizontal service corridors between closed sector spans. For input consistency, we add an extra zero at the end of this line. It should be read and ignored.
- A line with m integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le 10^6$): the heights of the vertical closed sector spans.
- A line with $m-1$ integers $d_1, d_2, \ldots, d_{m-1}$ ($1 \le d_i \le 10^6$): the heights of the vertical service corridors between closed sector spans. For input consistency, we add an extra zero at the end of this line. It should be read and ignored.

In each test case, $\max(b_1, \ldots, b_{n-1}, d_1, \ldots, d_{m-1}) \le \min(a_1, \ldots, a_n, c_1, \ldots, c_m)$.

The sum of $n \cdot m$ over all test cases does not exceed $4 \cdot 10^6$.

Output

For each test case print a single real number: the minimum Euclidean distance needed to travel from the top-left corner to the bottom-right corner, without passing through the interior of any closed sector.

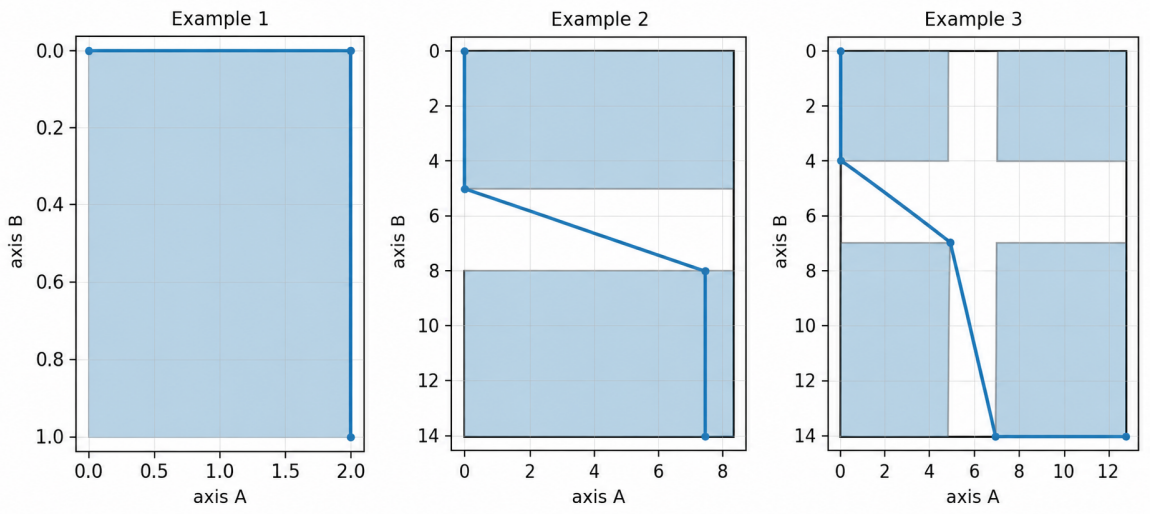
Your answer must have an absolute or relative error of at most 10^{-9}.

Example

standard input	*standard output*
3 1 1 2 0 1 0 1 2 7 0 5 6 3 0 2 2 5 6 2 0 4 7 3 0	3.0000000000 18.6157731059 23.1110617841

Note

You can see the illustrations for the two example test cases below.



Problem F. Communication Antennas

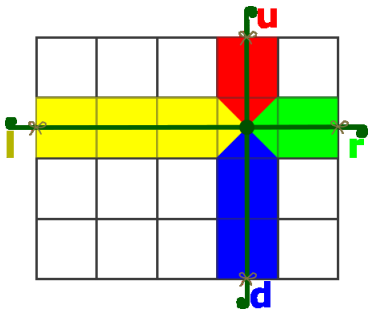
Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The communication deck of the orbital station is a rectangular area enclosed by the outer hull along its perimeter. The deck is divided into an $n \times m$ grid of square sectors of size 1×1.

In the center of some square sectors, there are four identical directional antennas with range $\ell_{r,c}$. You want to activate as many antennas as possible and switch the remaining antennas off.

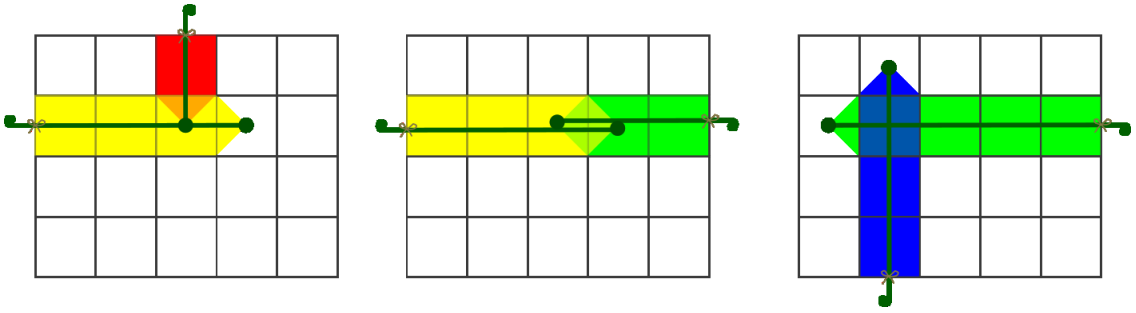
To activate an antenna, you can point it in a straight line along the field toward the perimeter receiver rail. The antenna must have enough range to reach the corresponding segment of the receiver rail. You may only point antennas in directions parallel to the sides of the field (up, right, down, or left).

Each activated antenna reserves all the sectors along its signal corridor to the perimeter, as well as a quarter of its own starting sector, as shown in the figure below (each antenna's reserved area is shown in a different color):



No two activated antennas can share the same reserved space. In particular, for each sector with antennas, you can potentially activate any number of antennas from 0 to 4, but no two of them can be pointed in the same direction.

Below are three examples where the signal corridors intersect. These activation configurations are invalid:



Your task is to activate as many antennas as possible and output which antennas are activated and in which direction.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

Each test case starts with a line containing three integers n, m, and s ($1 \le n, m \le 10^6$, $n \cdot m \le 10^6$, $1 \le s \le \min(10^5, n \cdot m)$): the size of the field and the number of sectors with antennas.

Each of the next s lines contains three integers r_i, c_i, and ℓ_i ($1 \le r_i \le n$, $1 \le c_i \le m$, $1 \le \ell_i \le 10^6$): the row, column, and range of the antennas in the i-th sector. Each pair (r_i, c_i) appears at most once per test case.

The sum of $n \cdot m$ over all test cases does not exceed 10^6, and the sum of s over all test cases does not exceed 10^5.

Output

For each test case, on the first line, print the answer k: the maximum number of antennas that can be activated.

Then print k lines, each containing two integers r_j and c_j, the coordinates of the sector with the antenna, and a character $d_j \in \{$'`u`','`r`','`d`','`l`'$\}$ indicating the direction of the antenna (up, right, down, or left, respectively).

If multiple solutions exist, print any one of them.

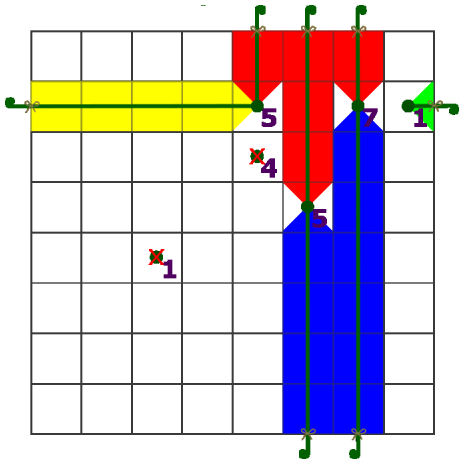
Example

standard input	*standard output*
2 4 5 1 2 4 9 8 8 6 2 5 5 2 7 7 2 8 1 3 5 4 4 6 5 5 3 1	4 2 4 u 2 4 d 2 4 r 2 4 l 7 2 5 u 2 7 u 4 6 d 2 8 r 2 5 l 4 6 u 2 7 d

Note

The first test case is illustrated in the statement.

A valid activation plan for the second test case is shown below:



Note that switched-off antennas do not block the signal corridors of activated antennas. For example, antenna "`2 8 r`" could be replaced with "`2 7 r`", and the solution would remain valid. Other valid solutions are also possible.

Problem G. Signal Messages

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

The orbital station stores old communication logs from its service modules. Most messages are ordinary maintenance notes, but some of them are generated by an ancient beacon protocol used for repeated calibration signals.

The protocol chooses a non-empty lowercase word $\mathbb{M}$ as a signal key. One valid signal block consists of two consecutive lines. The first line repeats the key in the old compact form, and the second line confirms the same key by adding the fixed prefix `i said`. Formally, given a non-empty word $\mathbb{M}$ consisting of lowercase English letters, the $\mathbb{M}$-signal message is composed of a **block of two lines** that can be repeated multiple times using the same word $\mathbb{M}$. Each block looks like:

$\mathbb{M}$`, `$\mathbb{M}$` `$\mathbb{M}$`ity `$\mathbb{M}$
`i said `$\mathbb{M}$`, `$\mathbb{M}$` `$\mathbb{M}$`ity `$\mathbb{M}$

Given a large multiline station log, determine the maximum possible amount of characters in some sequence of consecutive blocks that forms a valid signal message.

Input

The input consists of multiple lines, consisting of lowercase English letters, commas, and spaces. Each line ends with a newline character. The total number of characters including endlines does not exceed 10^5. There are no empty lines, and lines do not start or end with a space.

Output

Print a single integer: the maximum possible amount of characters in some consecutive sequence of blocks that matches the signal-message pattern. This includes newline characters at the end of each line of the sequence, including the last line of the signal message. Each newline is considered to be 1 character: pay attention to this if you have a Windows setup.

If there are no blocks matching the signal-message pattern, output -1.

Examples

standard input	*standard output*
pulse, pulse pulseity pulse i said pulse, pulse pulseity pulse	63
orbit, orbit orbitity orbit i said orbit, orbit orbitity orbit module, module moduleity module i said module, module moduleity module module, module moduleity module i said module, module moduleity module	142
control room is quiet, signal deck is blue, checksum reports are calm, but no beacon is true	-1

Note

In the first example, the whole log is one valid signal block with key `pulse`, so the answer is 63.

In the second example, the last four lines form two consecutive signal blocks with the same key `module`. This sequence is longer than the single `orbit` block, so the answer is 142.

The third example contains no valid signal blocks.

Problem H. Capsule Passenger Distribution

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are managing a row of n transport capsules on an orbital station. There are $n \cdot k$ passengers unevenly distributed over the capsules; more precisely, there are h_i passengers in capsule i. You want to balance the load and have exactly k passengers in every capsule.

At every transfer cycle, some passengers (probably none) can leave their capsule and move to another one, but they cannot move more than d capsules in one transfer.

It's up to you to decide who moves where and when. How many transfer cycles do you need to achieve your goal?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and d ($1 \le d \le n \le 10^6$): the number of capsules and the distance one passenger can move at once.

The second line of each test case contains n integers $h_1, h_2, \ldots, h_n$ ($0 \le h_i \le 10^9$): the distribution of passengers over the capsules. The sum of h_i is divisible by n.

The sum of n over all test cases does not exceed 10^6.

Output

For each test case, output a line with a single integer: how many transfer cycles you need.

Example

standard input	*standard output*
3 1 1 5 3 2 3 0 0 10 1 0 0 10 0 0 0 0 10 0 0	0 1 2

Note

In the first test case, there is a single capsule, so passengers are evenly distributed from the start.

In the second test case, before the first transfer cycle, there are 3 passengers in the first capsule: Aleksei, Dima, and Kostya. During the first cycle, Dima moves to the second capsule, and Kostya moves to the third one, so one cycle is enough to make every capsule have one passenger.

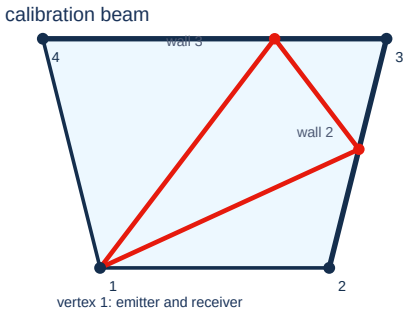
Problem I. Optical Calibrator

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The orbital station's engineers are installing a new optical calibrator in one of its shielded service modules. The module's cross-section is shaped like a convex polygon with n vertices, and thus has n reflective walls. There is a service hatch located at vertex 1. This hatch already contains a fixed calibration unit that protects walls 1 (between vertices 1 and 2) and n (between vertices n and 1).

However, the remaining reflective walls 2 through $n-1$ still need to be calibrated. Fortunately, the hatch also has a steerable optical emitter and receiver installed at vertex 1. You want to aim the calibration beam in such a way that:

- it starts at vertex 1,
- reflects in this exact order off wall 2, then wall 3, and so on until it reflects off wall $n-1$,
- and finally returns precisely to vertex 1.



All reflections follow the laws of ideal optics (angle of incidence equals angle of reflection), and the polygon is convex. You need to determine if it is possible to align the calibrator in the described way. If so, compute the required firing angle.

The base direction (angle 0°) is the positive x-axis, and angles are measured in degrees counter-clockwise.

Important: To keep the calibration stable, you decided to have all reflection points strictly inside their respective walls: more precisely, to be at least 10^{-5} **relative** units away from both endpoints. Formally, the valid points of reflection p for the wall $[a, b]$ are of the form $p = \alpha a + (1-\alpha)b$, where $\min(\alpha, 1-\alpha) \geq 10^{-5}$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

For each test case, the first line contains a single integer n ($3 \leq n \leq 20$): the number of vertices of the convex polygon.

Each of the next n lines contains two integers x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$): the coordinates of the vertices. The vertices are given in counter-clockwise order and form a strictly convex polygon.

Output

For each test case, output a single line "`NO`" if there is no answer.

Otherwise output two lines. The first line must contain "`YES`". On the second line, print a real number d ($-180 \leq d \leq 180$): the angle (in degrees) relative to the positive x-axis (counter-clockwise), at which the beam should be fired towards wall 2.

The trajectory of the ray should hit all the walls in required order at points inside the walls, and end in the 10^{-7}-neighborhood of vertex 1. The 10^{-7}-neighborhood of vertex 1 is defined as any point (x, y) on the walls 1 or n that follows $(x_1 - x)^2 + (y_1 - y)^2 < (10^{-7} \cdot L)^2$, where L is the total length of polygon sides.

If multiple angles satisfy the condition, output any one suitable angle.

Examples

standard input	*standard output*
2 3 2 3 0 0 6 0 4 0 0 2 0 2 3 0 3	YES -90 NO
3 4 0 0 6 0 8 5 -2 5 5 1 0 7 0 10 4 5 9 -1 5 6 -6 0 -3 -5 3 -5 6 0 3 5 -3 5	YES 10.20397372173168 YES 8.28061631487776 NO

Note

Illustration for the first test case of the second example is provided in the statement.

It is guaranteed that, for each test with answer "YES", there exists a shooting angle that has all its reflections at points $p = \alpha a + (1 - \alpha)b$, where $\min(\alpha, 1 - \alpha) \geq 10^{-5} + 10^{-9}$). It is also guaranteed that, for each test with answer "NO", there are no shooting angles that are *almost* valid: this means that at some point there is either no intersection, or $\min(\alpha, 1 - \alpha) < 10^{-5} - 10^{-9}$.

Problem J. Broken Navigation Console

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

It is not enough to plot the route, you still have to make the console obey it.

Put the cursor at the end of a row on the station console. Press Up and Right again and again. You learned a convenient emergency navigation trick.

The orbital station has a route table on its navigation console. During a calibration drill, several control keys stopped responding. The station technician still needs to move the cursor through the table and bring it to the top command row; fortunately, the Up and Right arrows still work.

The route table consists of n rows numbered from 1 to n, where row i contains s_i possible cursor positions from $(i, 1)$ to (i, s_i), denoting the j-th position on the i-th row as (i, j).

Navigation on a damaged console is a bit tricky. Fortunately, the technician still has Up and Right arrows working:

- Up ($\uparrow$): if the cursor is at position (i, j), then if $i = 1$, the cursor stays in place; otherwise, it moves to position $(i - 1, \min(j, s_{i-1}))$, where s_{i-1} is the length of row $i - 1$.
- Right ($\rightarrow$): if the cursor is at position (i, j), then if $j < s_i$, the cursor moves to $(i, j + 1)$; otherwise, it moves to $(i + 1, 1)$; except when at (n, s_n), then it stays in place.

The technician has a fallback routine. They pick an initial cursor position and start to perform an infinite sequence of key presses: u times Up, followed by r times Right, then again u times Up, followed by r times Right, and so on.

For example, let $n = 5$, $u = 2$, $r = 5$, and $s = \{5, 2, 4, 3, 1\}$. Starting from position $(4, 3)$, the first few key presses will move the cursor as follows:

$$(4, 3) \uparrow (3, 3) \uparrow (2, 2) \rightarrow (3, 1) \rightarrow (3, 2) \rightarrow (3, 3) \rightarrow (3, 4) \rightarrow (4, 1) \ldots$$

A position is called *successful* if, starting from this position and performing the infinite routine, the technician will sooner or later have the cursor on the first row.

Your task is to help the station crew count how many successful starting positions exist in the route table.

The table is still being recalibrated, so you also need to handle q updates. In each update, you get two values, *pos* and x, and you have to assign s_{pos} a new value x. You need to give the answer to the problem before any updates and after processing each update.

Input

The first line contains three integers: n, u, and r ($1 \le n \le 10^5$, $1 \le u, r \le 10^{18}$).

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 10^{12}$): the number of positions in each row of the route table.

The third line contains a single integer q ($1 \le q \le 10^5$), the number of updates.

Each of the next q lines contains two integers, *pos* and x ($1 \le pos \le n$, $1 \le x \le 10^{12}$): the updates.

The updates are consecutive, so each update is applied to the current state of the route table, not the initial one.

Output

Output $q + 1$ integers, each on a separate line: the number of successful starting positions before all the updates and after each one of them.

Examples

standard input	*standard output*
5 2 5 5 2 4 3 1 1 1 1	15 11
5 2 10 5 2 4 3 1 1 2 3	11 12
10 1 42 169 42 42 42 42 42 42 42 42 42 1 2 43	211 254

Note

In the first example, before any updates, starting from position $(5, 1)$ and pressing "2 times Up followed by 5 times Right" in a loop:

Round 1: $(5, 1) \uparrow (4, 1) \uparrow (3, 1) \rightarrow (3, 2) \rightarrow (3, 3) \rightarrow (3, 4) \rightarrow (4, 1) \rightarrow (4, 2)$

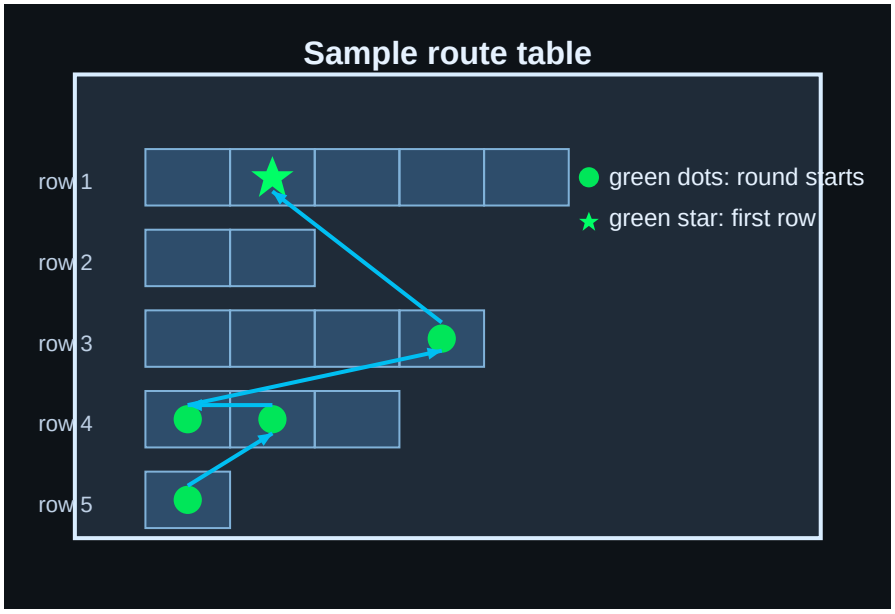
Round 2: $(4, 2) \uparrow (3, 2) \uparrow (2, 2) \rightarrow (3, 1) \rightarrow (3, 2) \rightarrow (3, 3) \rightarrow (3, 4) \rightarrow (4, 1)$

Round 3: $(4, 1) \uparrow (3, 1) \uparrow (2, 1) \rightarrow (2, 2) \rightarrow (3, 1) \rightarrow (3, 2) \rightarrow (3, 3) \rightarrow (3, 4)$

Round 4: $(3, 4) \uparrow (2, 2) \uparrow (\mathbf{1}, \mathbf{2}) \rightarrow \ldots$

Thus, $(5, 1)$ is a successful position.

On the image below you can see the starting positions of each round marked with the green dots and the final position marked with the green star.



Problem K. Power Consumption Optimization

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	1024 mebibytes

The orbital station has a row of n power modules. Module i currently consumes a_i units of power. During a power-saving calibration, the station controller may throttle selected modules one unit at a time.

You are given an array $a = [a_1, a_2, \ldots, a_n]$ of n positive integers.

We define $f_k(b_1, b_2, \ldots, b_m)$ to be the minimum possible product of the array $[b_1, \ldots, b_m]$ that can be obtained by applying the following operation at most k times:

- Select any index i such that $b_i > 1$ and assign $b_i := b_i - 1$.

Your task is to process q maintenance queries. Each query is given as three integers ℓ, r, and k, and asks you to compute the value of $f_k(a_\ell, a_{\ell+1}, \ldots, a_r)$: the minimum possible product of the consumption levels of modules ℓ through r after applying at most k throttling operations, as described above.

Since the answer may be large, output it modulo 998 244 353.

Input

The first line contains two integers n and q ($1 \le n \le 2 \cdot 10^5$, $1 \le q \le 5 \cdot 10^5$): the number of power modules and the number of queries.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 998\,244\,352$): the current power consumption levels of the modules.

Each of the next q lines contains three integers ℓ, r, and k ($1 \le \ell \le r \le n$, $0 \le k \le 10^{18}$) describing a query on the station sector from module ℓ to module r, with at most k allowed throttling operations.

Output

For each query, print one integer: the value of $f_k(a_\ell, \ldots, a_r)$ modulo 998 244 353.

Example

standard input	*standard output*
5 2 1 2 3 4 5 1 2 3 4 5 1	1 15

Note

The product is computed after all throttling operations are applied, and only the final result should be taken modulo 998 244 353.

Problem L. Service Drone Route

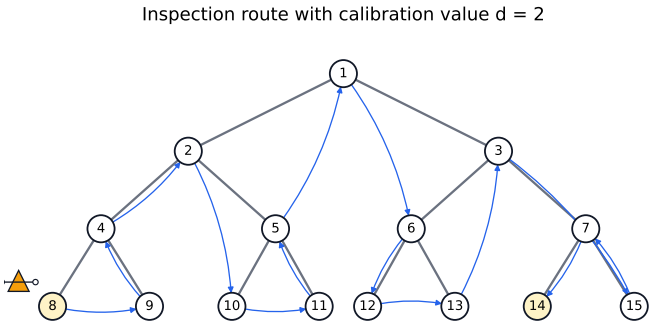
Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

A maintenance crew on the orbital station uses a small service drone to inspect remote service modules.

The part of the station that must be inspected can be represented as a tree (connected graph without cycles) of size n, where each service module is a vertex, and an edge exists if the two modules are directly connected by a short service corridor of length 1.

The drone has to visit every service module in this part of the station. To avoid wasting battery on repeated checks, each module must be visited only once. Before starting the route, engineers may spend some number of days calibrating the drone's navigation system. If they spent d days on calibration, the drone is able to make one programmed flight of distance no more than d in the tree. Here, the distance $\text{dist}(u, v)$ is defined as the number of edges on the simple path between vertices u and v. A module is considered visited either if it was chosen as the starting module or if the drone has already landed on it. Please note that modules passed over between the start and end modules of a single flight are not marked as visited.

What is the minimum number of calibration days needed to be able to visit all modules? You can choose any starting service module you want.



Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains one integer n ($2 \le n \le 10^6$), the size of the tree.

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$): the vertices connected by an edge of the tree.

The sum of n over all test cases doesn't exceed 10^6.

Output

For each test case, output a line with a single integer: the minimum number of calibration days needed to be able to fly over the given tree.

Example

standard input	*standard output*
4 3 1 2 1 3 5 1 2 1 3 1 4 1 5 7 1 2 1 3 2 4 2 5 3 6 3 7 15 1 2 1 3 2 4 2 5 3 6 3 7 4 8 4 9 5 10 5 11 6 12 6 13 7 14 7 15	1 2 2 2

Note

For the first test case, the drone can perform the following inspection route: $3 \to 1 \to 2$. For each flight, the distance is 1.

The fourth test case describes the route map from the statement. With $d = 2$, a valid inspection route is: $8 \to 9 \to 4 \to 2 \to 10 \to 11 \to 5 \to 1 \to 6 \to 12 \to 13 \to 3 \to 15 \to 7 \to 14$. The distances are: $2, 1, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 1$. So 2 calibration days are enough.

Problem M. Station Blueprint on a Coordinate Grid

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	1024 mebibytes

The engineering crew is preparing the blueprints for an orbital station. For one of the subsystems, they were given an undirected connected graph describing junctions and direct corridor segments. The issue is, it is not easy to draw a reliable station blueprint.

The crew thinks that the best place to draw an undirected connected graph with n vertices and m edges is a coordinate grid on the planning board. Each vertex of the graph will be an integer point of the grid (x, y), and each edge will be a line connecting either points $(x, y) \leftrightarrow (x + 1, y)$ or $(x, y) \leftrightarrow (x, y + 1)$ for some integers x and y. They can just highlight some corridor segments of the grid, and the station blueprint will be ready!

However, it would be nice if the drawing preserved the distances in the graph. Formally, consider points with integer coordinates on a two-dimensional plane. A *drawing* is an array of n such points $p_1, \ldots, p_n$. A drawing is *nice* if $\text{dist}(i, j) = \text{Manhattan}(p_i, p_j)$ for all possible i and j. Here, $\text{dist}(u, v)$ is defined as the number of edges on the shortest path between vertices u and v in the graph, while $\text{Manhattan}(p_i, p_j)$ is defined as the Manhattan distance between two integer points p_i and p_j on the plane.

Find a nice drawing for the given graph or indicate that it does not exist.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 50\,000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 500\,000$, $n - 1 \le m \le 500\,000$): the number of vertices and edges in the graph.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$): the two vertices connected by an edge of the graph. The graph is connected.

Both the sum of n and the sum of m over all test cases don't exceed $500\,000$.

For the convenience of visual perception, there is an empty line before every test case. It can be ignored.

Output

For each test case, print "`NO`" on a single line if no nice drawing exists.

Otherwise, print "`YES`" on the first line. After that, print an example of a nice drawing. Each of the following n lines should contain two integers x_i and y_i: the coordinates of the i-th point

($-998\,244\,353 \le x_i, y_i \le 998\,244\,353$). If there are several possible answers, print any one of them.

Examples

standard input	*standard output*
3 4 4 1 2 2 3 3 4 4 1 5 4 1 2 1 3 1 4 1 5 7 8 1 2 1 3 2 4 3 4 4 5 4 6 5 7 6 7	YES 0 0 1 0 1 1 0 1 YES 3 1 2 1 3 0 4 1 3 2 YES 1 2 2 2 1 3 2 3 3 3 2 4 3 4
2 6 6 1 2 2 3 3 4 4 5 5 6 6 1 12 16 1 2 1 3 2 4 3 4 3 5 4 6 5 6 6 7 4 8 7 8 7 9 8 10 9 10 8 11 10 12 11 12	NO NO