The 2026 ICPC Asia Chengdu Regional Contest

Hosted by UESTC





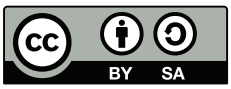
电子科技大学

— 70周年校庆 —

70th ANNIVERSARY OF UESTC

Problems

A	Average of Averages
B	Bouncing Particles
C	Counting Arrays
D	Duel of Wrath
E	Enclose the Point
F	First Word
G	Graph Reconstruction
H	Hidden Order
I	Invisible Edges
J	Joint Relocation
K	Keep Your Distance
L	Long Slides
M	Minimum Records

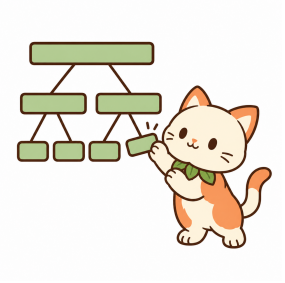


Copyright © 2026. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
https://creativecommons.org/licenses/by-sa/4.0/

A Average of Averages

Time limit: 1s

The average of averages is not always the average.



VegiCat has just learned about segment trees and wants to practice using them.

For a positive integer n, he considers the sequence $a_i = i$ $(1 \le i \le n)$. He decides to build a segment tree to answer **range average** queries on this sequence.

In VegiCat's segment tree, each node corresponds to an interval $[l, r]$ and stores a value $v_{l,r}$, which VegiCat intends to represent the average over that interval. The root corresponds to the entire interval $[1, n]$.

However, VegiCat mistakenly assumes that a parent interval's average always equals the average of the values stored in its two child nodes. Therefore, he adds the values stored in the two child nodes and divides the sum by 2.

VegiCat first constructs the segment tree by calling Build$(1, n)$. After the segment tree has been built, he answers a query for an interval $[L, R]$ by calling Query$(1, n, L, R)$. The two recursive procedures are given in Algorithm 1.

Algorithm 1 Build and Query

```
1:  procedure BUILD(l, r)
2:      if l = r then
3:          v_{l,l} ← a_l
4:          return
5:      end if
6:      m ← ⌊(l + r) / 2⌋
7:      BUILD(l, m)
8:      BUILD(m + 1, r)
9:      v_{l,r} ← (v_{l,m} + v_{m+1,r}) / 2
10: end procedure

11: procedure QUERY(l, r, L, R)
12:     if L ≤ l and r ≤ R then
13:         return v_{l,r}
14:     end if
15:     m ← ⌊(l + r) / 2⌋
16:     if R ≤ m then
17:         return QUERY(l, m, L, R)
18:     else if L > m then
19:         return QUERY(m + 1, r, L, R)
20:     end if
21:     x ← QUERY(l, m, L, R)
22:     y ← QUERY(m + 1, r, L, R)
23:     return (x + y) / 2
24: end procedure
```

Although VegiCat's algorithm is incorrect in general, it returns the correct average for some intervals. Count the number of such intervals.

For an interval $[L, R]$, its true average is

$$\frac{\sum_{i=L}^{R} a_i}{R - L + 1}.$$

More precisely, count the number of pairs (L, R) satisfying $1 \le L \le R \le n$ such that

$$\text{Query}(1, n, L, R) = \frac{\sum_{i=L}^{R} a_i}{R - L + 1}.$$

All averages are computed as exact **rational numbers**, without rounding or truncation.

Input

The first line contains an integer T ($1 \le T \le 10^4$), the number of test cases. The description of the test cases follows.

Each of the next T lines contains one integer n ($1 \le n \le 10^6$), the length of the sequence.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^6$.

Output

For each test case, output an integer — the number of intervals $[L, R]$ for which VegiCat's query procedure returns exactly the true average.

Sample Input 1	Sample Output 1
3 1 5 14	1 10 35

Note

For the second test case, $n = 5$, there are 10 valid intervals:

$$[1,1],\ [1,2],\ [2,2],\ [2,3],\ [2,5],$$
$$[3,3],\ [3,4],\ [4,4],\ [4,5],\ [5,5].$$

For example, querying $[2, 5]$ returns

$$\frac{\frac{2+3}{2} + \frac{4+5}{2}}{2} = \frac{7}{2},$$

which equals the true average of $2, 3, 4, 5$. Thus, $[2, 5]$ is counted.

In contrast, querying $[1, 5]$ returns

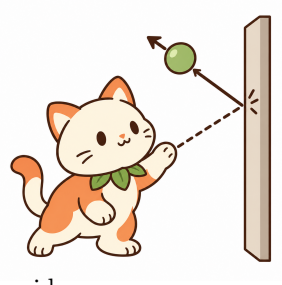
$$\frac{\frac{\frac{1+2}{2}+3}{2} + \frac{4+5}{2}}{2} = \frac{27}{8},$$

while its true average is 3. Thus, $[1, 5]$ is not counted.

B Bouncing Particles

Time limit: 2s

This problem has an attachment. You can use it to simulate and visualize the movements of the particles.



VegiCat observes particles on a grid with n rows and n columns. The lower-left cell has coordinates $(1, 1)$. The horizontal coordinate increases from left to right, and the vertical coordinate increases from bottom to top. Each particle has a horizontal direction (left or right) and a vertical direction (up or down). At the beginning of each step, each particle adjusts its directions as follows:

- Its horizontal direction is reversed if the next cell in that direction is outside the grid.
- Its vertical direction is reversed if the next cell in that direction is outside the grid.

After these adjustments, a particle at (x, y) produces two child particles:

- One at $(x-1, y)$ if its horizontal direction is left, or at $(x+1, y)$ if it is right;
- One at $(x, y-1)$ if its vertical direction is down, or at $(x, y+1)$ if it is up.

Both child particles inherit the parent's horizontal and vertical directions. The parent then disappears.

Multiple particles may occupy the same cell at the same time, regardless of their directions. They do not interact or change each other's directions.

At time 0, there are four particles at (x_0, y_0), with the following direction pairs:

- Right and up;
- Left and up;
- Right and down;
- Left and down.

At each step, all particles simultaneously adjust their directions and then split as described above.

You are given q queries. Each query gives integers A, B, C and a time t. For each query, find the **number of grid cells** (x, y) that satisfy $Ax + By + C = 0$ and contain at least one particle at time t.

Input

The first line contains an integer T ($1 \le T \le 10^5$), the number of test cases. The description of the test cases follows.

The first line of each test case contains four integers n, x_0, y_0, q ($2 \le n \le 10^9$, $1 \le x_0, y_0 \le n$, $1 \le q \le 10^5$). The grid has n rows and n columns. The initial cell is (x_0, y_0), and there are q queries.

Each of the next q lines contains four integers A, B, C and t ($|A|, |B|, |C| \le 10^9$, $0 \le t \le 10^9$). This describes a query for the line $Ax + By + C = 0$ at time t. It is guaranteed that A and B are not both 0 simultaneously.

It is guaranteed that the sum of q over all test cases does not exceed 10^5.

Output

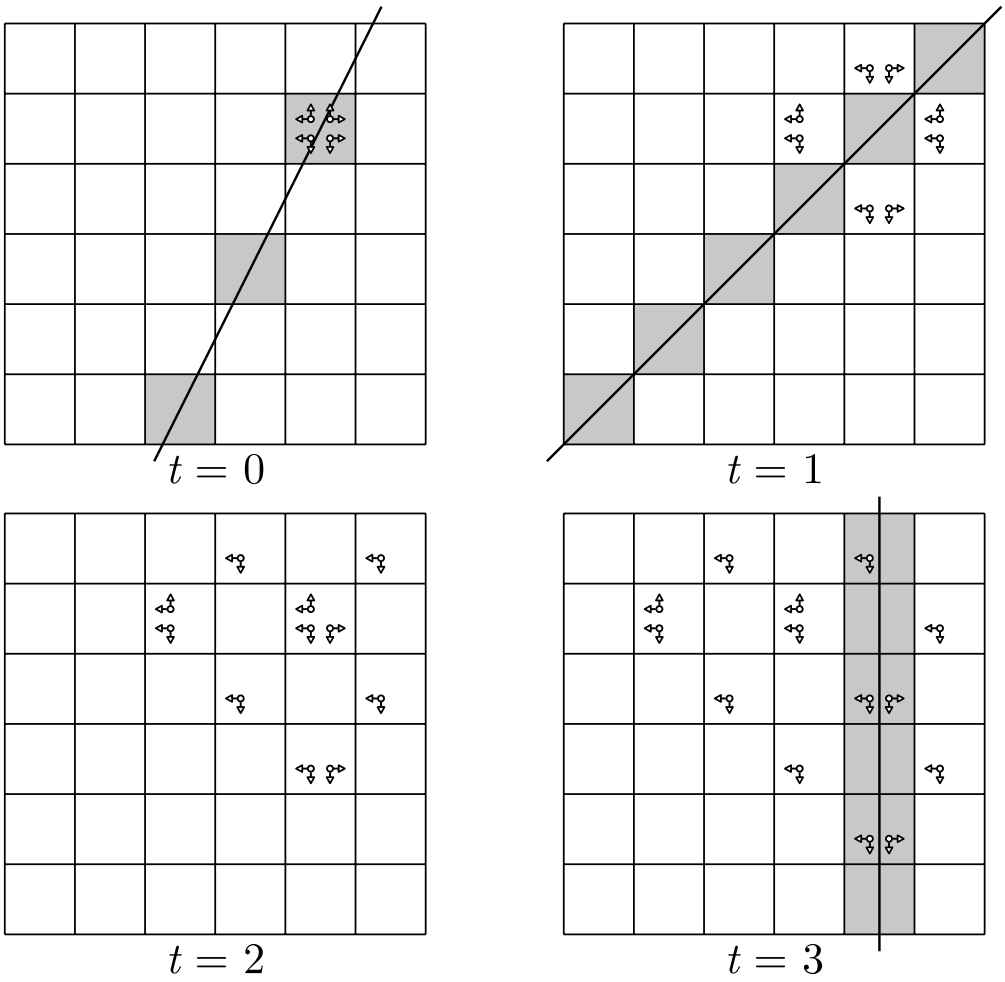
For each query, output one integer on one line: the number of grid cells (x, y) that satisfy $Ax + By + C = 0$ and contain at least one particle at time t.

Sample Input 1	Sample Output 1
1 6 5 5 4 2 -1 -5 0 1 -1 0 1 11 45 14 2 1 0 -5 3	1 0 0 3

Note

The figure below shows the particle evolution in the sample, as well as the queried straight lines, where the gray cells represent the cells containing integer points through which the straight line passes, and all splitting directions have undergone legality adjustments. Since the third query is too far away from the grid under consideration, the line of the third query is omitted here.

Particles in the same cell with the same direction pair are drawn as a single symbol; multiplicities are omitted.



C Counting Arrays

Time limit: 2s

VegiCat gives you two integers n and m. Consider integer arrays $a_1, a_2, \ldots, a_n$ such that

$$a_1 = 0, \qquad 0 \le a_{i+1} - a_i \le 1 \quad (1 \le i < n).$$



Let $s_i = a_1 + a_2 + \ldots + a_i$ be the prefix sums of a. VegiCat also tells you the prefix sums at m positions; that is, you are given m constraints of the form $s_{p_j} = v_j$ $(1 \le j \le m)$.

Count the number of arrays satisfying all constraints, modulo 998244353. If no array satisfies all the constraints, the answer is 0.

Input

The first line contains an integer T $(1 \le T \le 10^4)$, the number of test cases. The description of the test cases follows.

Each test case is given as follows:

The first line contains two integers n and m $(1 \le n \le 100, 0 \le m \le n)$, the length of the array and the number of prefix sum constraints, respectively.

The j-th of the next m lines contains two integers p_j and v_j $(1 \le p_j \le n, 0 \le v_j \le 10^9)$. This describes a constraint: the sum of the first p_j elements of the array must equal v_j.

The positions p_j are given in strictly increasing order.

Output

For each test case, output an integer — the number of valid arrays modulo 998244353.

Sample Input 1	Sample Output 1
6 1 0 3 1 3 1 3 1 2 0 4 1 4 3 5 2 1 0 5 4 5 2 2 0 5 10	1 1 2 2 2 0

Note

Without any prefix constraints there are exactly 2^{n-1} such arrays, because each of the $n-1$ adjacent differences may independently be 0 or 1.

In the second test case, the only array with $s_3 = 1$ is $(0, 0, 1)$.

In the third test case, $s_2 = 0$ forces $a_2 = 0$, after which a_3 may be 0 or 1.

In the fourth test case, the arrays are $(0, 1, 1, 1)$ and $(0, 0, 1, 2)$.

In the fifth test case, the arrays with $s_5 = 4$ are $(0, 0, 1, 1, 2)$ and $(0, 1, 1, 1, 1)$.

In the sixth test case, $s_2 = 0$ forces $a_1 = a_2 = 0$, so $s_5 \le 6$. The constraint $s_5 = 10$ is impossible.

This page is intentionally left blank.

D Duel of Wrath

Time limit: 1s

VegiCat and VegiDog are playing a special duel in Slay the Spire. VegiCat plays as Watcher A, and VegiDog plays as Watcher B. A and B have base attack powers a and b and initial health values h_A and h_B, respectively.



At the start of each round, the game determines each Watcher's stance for that round. Each Watcher can be in one of the following two stances:

- Normal: damage dealt and damage received are unchanged.
- Wrath: damage dealt and damage received are both doubled.

For each attack, the damage initially equals the attacker's base attack power. It is doubled if the attacker is in Wrath and doubled again if the defender is in Wrath. Thus, the damage in each round is as follows:

- If both characters are in Normal, A deals a damage to B, and B deals b damage to A.
- If exactly one character is in Wrath, A deals $2a$ damage to B, and B deals $2b$ damage to A.
- If both characters are in Wrath, A deals $4a$ damage to B, and B deals $4b$ damage to A.

Each round, both characters attack once simultaneously, and both health values decrease simultaneously. Both attacks take effect even if either character's health becomes 0 or less during the round. At the end of the round, a character with health less than or equal to 0 dies. If either character dies, the battle ends immediately; otherwise, the next round begins. Base attack powers never change, and health is never restored.

The stances for each round are determined by the game, not by the players. Any combination of the two characters' stances is possible in each round.

VegiCat wins only if B dies while A remains alive. If both characters die in the same round, VegiCat does not win. Determine whether VegiCat will win **regardless of how both characters' stances change**.

Input

The first line contains an integer T ($1 \le T \le 10^5$), the number of test cases. The description of the test cases follows.

Each test case contains one line with four integers a, b, h_A, and h_B ($1 \le a, b, h_A, h_B \le 10^{18}$), giving the characters' base attack powers and initial health values, respectively.

Output

For each test case, output `YES` if VegiCat will win regardless of how both characters' stances change, and `NO` otherwise. Output each answer on a separate line.

You can output the answer in any case. For example, `yEs`, `yes`, `Yes`, and `YES` are all accepted as positive responses.

Sample Input 1

```
5
1 1 5 1
1 1 4 1
1 1 1 1
1 1 1 5
2 3 13 2
```

Sample Output 1

```
YES
NO
NO
NO
YES
```

Sample Input 2

```
4
3 2 18 13
3 2 16 13
3 2 9 13
3 2 17 13
```

Sample Output 2

```
YES
NO
NO
YES
```

Note

In the first test case of the first sample, B always dies in the first round. A receives at most 4 damage in that round and remains alive.

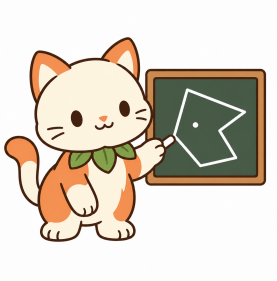
In the second test case of the first sample, A wins if both characters are normal in the first round. If both characters are in Wrath in the first round, both die. Therefore, the answer is `NO`.

In the second test case of the second sample, if both characters are normal for the first 4 rounds, their remaining health values are 8 and 1. If both enter Wrath in round 5, both die, so victory is not guaranteed.

E Enclose the Point

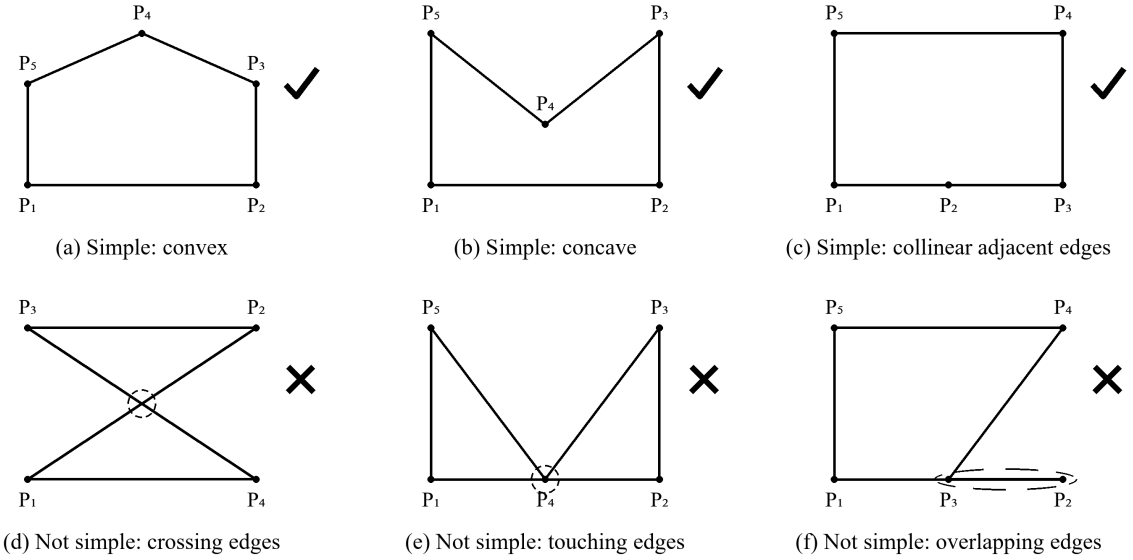
Time limit: 1s

There are n points $P_1, P_2, \ldots, P_n$ in the plane. VegiCat has q queries for you. Each query gives a point P.



For each query, construct a simple polygon using some of the given points as its vertices. The polygon must contain P **strictly** inside it and have the **minimum number of vertices**. If no such polygon exists, output -1.

A simple polygon is a closed polygonal chain with at least three distinct vertices. Adjacent edges intersect only at their common endpoint, and nonadjacent edges do not intersect.



(a) Simple: convex (b) Simple: concave (c) Simple: collinear adjacent edges

(d) Not simple: crossing edges (e) Not simple: touching edges (f) Not simple: overlapping edges

Examples of simple and non-simple polygons

Input

The first line contains an integer T ($1 \le T \le 2 \cdot 10^4$), the number of test cases. The description of the test cases follows.

The first line of each test case contains two positive integers n and q ($3 \le n \le 10^5$, $1 \le q \le 10^5$), representing the number of given points and the number of queries.

The i-th of the next n lines contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$), the coordinates of P_i. It is guaranteed that these n points are pairwise distinct.

Each of the next q lines contains two integers x and y ($-10^9 \le x, y \le 10^9$), the coordinates of the queried point.

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

It is guaranteed that the sum of q over all test cases does not exceed 10^5.

Output

For each query, if no polygon satisfying the requirements exists, output one integer -1 on a line.

Otherwise, output a polygon with the minimum number of vertices. On the first line, output one integer m, the number of vertices.

On the second line, output m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$), the indices of the vertices in order. The polygon is $P_{a_1} P_{a_2} \ldots P_{a_m}$. The vertices may be output in either clockwise or counterclockwise order.

The indices must be pairwise distinct, and the output polygon must be simple and contain the queried point strictly inside it. If there are multiple solutions, output any of them.

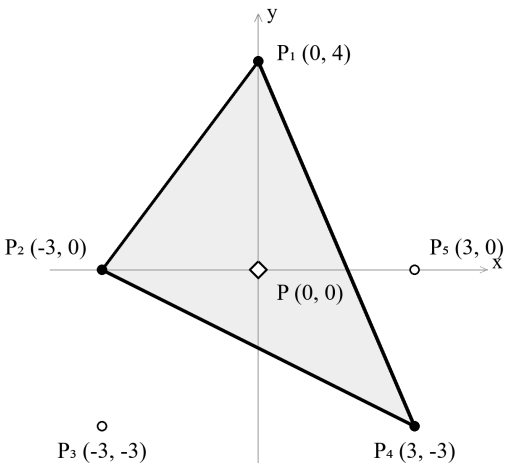
Sample Input 1	Sample Output 1
2 5 1 0 4 -3 0 -3 -3 3 -3 3 0 0 0 3 1 0 4 -3 0 3 0 0 0	3 4 1 2 -1

Note

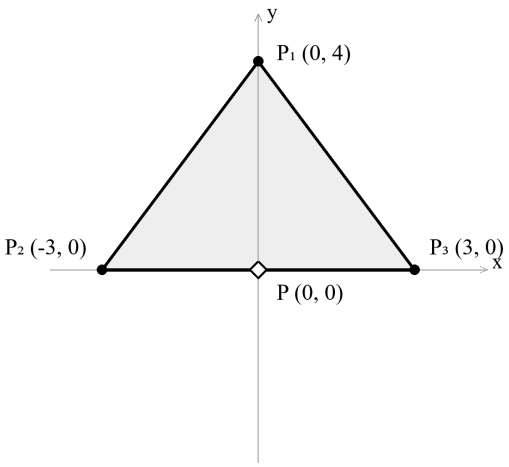
In the first test case, the triangle $P_4P_1P_2$ contains the queried point $P = (0, 0)$ strictly inside it. Since a polygon has at least three vertices, this triangle has the minimum number of vertices.

In the second test case, the only possible polygon is the triangle $P_1P_2P_3$. The queried point $P = (0, 0)$ lies on the edge P_2P_3, so the answer is -1.

(a) First test case



(b) Second test case



Illustrations of the sample test cases

F First Word

Time limit: 5s



VegiCat has a string S of n lowercase English letters. He wants to build a dictionary that contains k distinct nonempty substrings of S as its words. A substring is a contiguous part of a string. Substrings with the same content are treated as the same word, even if they occur at different positions in S.

Let π be a mapping from the letters $\texttt{a}, \texttt{b}, \ldots, \texttt{z}$ to the corresponding letters in a permutation of the 26 lowercase English letters. For example, the permutation `qwertyuiopasdfghjklzxcvbnm` defines the following mapping:

c	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
$\pi(c)$	q	w	e	r	t	y	u	i	o	p	a	s	d	f	g	h	j	k	l	z	x	c	v	b	n	m

For a string $u = u_1 u_2 \cdots u_m$, define

$$\pi(u) = \pi(u_1)\pi(u_2)\cdots\pi(u_m).$$

For example, with the mapping above, $\pi(\texttt{apple}) = \texttt{qhhst}$.

VegiCat wants the dictionary to contain a word w such that, after applying any such mapping π to every word in the dictionary, $\pi(w)$ is always the lexicographically smallest word in the resulting dictionary.

For each $k = 1, 2, \ldots, n$, find the number of dictionaries that contain k distinct words and satisfy the requirement, modulo 998244353. Two dictionaries are considered different if and only if they contain different sets of words before any mapping is applied.

Recall that string a is lexicographically smaller than string b if a is a prefix of b (and $a \neq b$), or there exists an index i $(1 \le i \le \min(|a|, |b|))$ such that $a_i < b_i$, and for any index j $(1 \le j < i)$ $a_j = b_j$.

Input

The first line contains an integer T $(1 \le T \le 10^4)$, the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n $(1 \le n \le 10^5)$.

The second line contains a string S of length n, consisting only of lowercase English letters.

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

Output

For each test case, output n integers on one line. The k-th integer represents the number of dictionaries that contain k distinct words and satisfy the requirement, modulo 998244353.

Sample Input 1	Sample Output 1
3 3 aab 4 ppqq 7 abcaabc	5 4 3 8 10 11 10 21 52 87 119 139 128 84

Note

Each dictionary below is written as a set of words.

For the first test case, $S = aab$, the dictionaries that satisfy the requirement for $k = 1, 2, 3$ are:

k	Word sets that satisfy the requirement	Number of dictionaries
1	{a}, {b}, {aa}, {ab}, {aab}	5
2	{a, aa}, {a, ab}, {a, aab}, {aa, aab}	4
3	{a, aa, ab}, {a, aa, aab}, {a, ab, aab}	3

This page is intentionally left blank.

G Graph Reconstruction

Time limit: 2s



VegiCat has an $n \times n$ integer matrix $A = (a_{ij})$. For each $1 \le i, j \le n$:

- If $i = j$, $i = 1$, or $j = 1$, then $a_{ij} = -1$.
- Otherwise, a_{ij} is a nonnegative integer.

Help VegiCat construct an undirected simple graph G with n vertices, numbered from 1 to n.

For every pair of distinct vertices i, j with $i > 1$ and $j > 1$, the graph must satisfy the following condition: after deleting vertices i and j, together with all edges that have an endpoint at either vertex, exactly a_{ij} edges must remain.

Each pair of vertices i, j is considered separately. That is, for each pair, start with G before any vertices or edges have been deleted.

It is guaranteed that at least one such graph exists. If there are multiple solutions, output any of them.

Input

The first line contains an integer T ($1 \le T \le 6 \cdot 10^4$), the number of test cases. The description of the test cases follows.

For each test case, the first line contains an integer n ($4 \le n \le 10^3$), denoting the number of vertices in the graph.

Then each of the next n lines contains n space-separated integers a_{ij} ($-1 \le a_{ij} \le \frac{(n-3)(n-2)}{2}$), representing the matrix A described in the statement.

It is guaranteed that the total sum of n^2 over all test cases does not exceed $1.2 \cdot 10^6$, and that at least one graph corresponding to matrix A exists.

Output

For each test case, output the adjacency matrix of the graph you construct. Output n lines, each containing n space-separated integers. Each integer must be either 0 or 1. The entry in the i-th row and j-th column is 1 if there is an edge between vertices i and j, and 0 otherwise. If there are multiple solutions, output any of them.

Sample Input 1

```
2
5
-1 -1 -1 -1 -1
-1 -1  3  1  2
-1  3 -1  2  2
-1  1  2 -1  1
-1  2  2  1 -1
7
-1 -1 -1 -1 -1 -1 -1
-1 -1  4  5  3  6  5
-1  4 -1  5  4  5  5
-1  5  5 -1  5  7  5
-1  3  4  5 -1  4  4
-1  6  5  7  4 -1  5
-1  5  5  5  4  5 -1
```

Sample Output 1

```
0 1 0 1 1
1 0 0 0 0
0 0 0 1 0
1 0 1 0 1
1 0 0 1 0
0 1 1 0 1 0 0
1 0 0 0 0 1 1
1 0 0 0 1 0 1
0 0 0 0 1 1 0
1 0 1 1 0 0 1
0 1 0 1 0 0 0
0 1 1 0 1 0 0
```

Note

The actual input matrix will only have a single space between two integers. The extra spaces in the sample are designed to enhance the readability of the layout.

This page is intentionally left blank.

H Hidden Order

Time limit: 1s

This is an interactive problem.



VegiCat knows an array $a_1, a_2, \ldots, a_n$, where each element is 0, 1, or 2. He does not want to tell you its values, but you can ask him questions. In each query, you choose two indices i and j, and VegiCat tells you whether $a_i > a_j$.

VegiCat asks you to output the indices in nondecreasing order of the corresponding array values. That is, find a permutation $p_1, p_2, \ldots, p_n$ of $1, 2, \ldots, n$ such that

$$a_{p_1} \le a_{p_2} \le \cdots \le a_{p_n}.$$

You may make at most $2n - 3$ queries in each test case.

The interactor is adaptive. This means that the array is not fixed in advance. The interactor guarantees that there always exists at least one array with elements in $\{0, 1, 2\}$ consistent with all responses given so far. Your final permutation must be valid for **every array consistent with all responses**.

Input

The first line contains an integer T ($1 \le T \le 250$), the number of test cases. The description of the test cases follows.

Each test case begins with an integer n ($2 \le n \le 500$), the length of the array.

The sum of n over all test cases does not exceed 500.

Interaction

To make a query, output a line in the following format:

```
? i j
```

Here, $1 \le i, j \le n$. After each query, read one integer:

- 1, if $a_i > a_j$;
- 0, if $a_i \le a_j$.

Queries with $i = j$ and repeated queries are allowed and are included in the query count.

When you have determined the order, output a line in the following format:

```
! p_1 p_2 ... p_n
```

Each index from 1 to n must appear exactly once. Printing the answer does not count as a query.

If your answer is correct, proceed to the next test case by reading its n, or terminate your program if this was the last test case. The interactor does not send a separate confirmation after an answer.

After outputting each query or answer, output a newline and flush the output buffer. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

If you make an invalid query, exceed the query limit, or print an incorrect answer, you will receive a **Wrong Answer** verdict. Output that does not follow the specified format is also judged **Wrong Answer**. In these cases, the interactor terminates without sending an error response.

Read	Sample Interaction 1	Write
2 3		
	? 1 2	
1		
	? 2 3	
1		
	! 3 2 1	
2		
	? 1 2	
0		
	! 1 2	

Note

In the first test case, the queries `? 1 2` and `? 2 3` both return 1, so $a_1 > a_2 > a_3$. Therefore, $(3, 2, 1)$ is a correct answer.

In the second test case, the query `? 1 2` returns 0, so $a_1 \leq a_2$. Therefore, $(1, 2)$ is a correct answer.

I Invisible Edges

Time limit: 1s

This is a multi-pass problem.



VegiCat and VegiDog play a game with a forest of n vertices, numbered from 1 to n, and m edges. A forest is an undirected graph with no cycles. It is **not necessarily connected**.

In the first stage, VegiCat assigns an integer a_u ($1 \le a_u \le 2n$) to each vertex u. The values may repeat.

After VegiCat finishes assigning the values, the vertices are renumbered. Let $p_1, p_2, \ldots, p_n$ be a permutation of $1, 2, \ldots, n$, in which each integer from 1 to n appears exactly once. For each u, the vertex originally numbered u is now numbered p_u.

Each edge (u, v) becomes (p_u, p_v). Each vertex keeps its assigned value. Let b_i be the value at vertex i after renumbering. Then $b_{p_u} = a_u$.

In the second stage, VegiDog receives only n and $b_1, b_2, \ldots, b_n$, without the original forest or p. He must recover all edges using the new vertex numbers.

Design strategies for both players that work for every valid forest and every permutation p.

Input

Your program will run twice: first as VegiCat, then as VegiDog. The test cases appear in the same order in both runs.

First Run. The first line contains the string `cat`, indicating that you play VegiCat.

The second line contains an integer T ($1 \le T \le 10^4$), the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m < n$), the number of vertices and edges.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), describing an undirected edge between vertices u and v.

It is guaranteed that the graph has no cycles or multiple edges.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Second Run. The first line contains the string `dog`, indicating that you play VegiDog.

The second line contains an integer T ($1 \le T \le 10^4$), the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$), the number of vertices.

The second line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 2n$), the assigned values in the new vertex order.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

First Run. For each test case, output n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2n$) on one line. The integer a_i is the value assigned to vertex i.

Second Run. For each test case, output an integer m on the first line, the number of edges. Each of the next m lines must contain two integers u and v ($1 \le u, v \le n$, $u \ne v$), describing an edge using the new vertex numbers.

Output each edge once, in any order. Either endpoint may come first. If $m = 0$, output only 0.

Sample 1

Input	Pass 1	Output
cat 4 1 0 4 3 1 2 4 3 2 3 7 2 6 4 7 1 10 7 10 9 7 9 10 5 4 8 3 7 4 10 1 2		1 2 5 8 5 11 11 4 11 3 7 2 10 13 12 5 3 2 16 6 19 3

Input	Pass 2	Output
dog 4 1 1 4 5 2 5 8 7 11 7 2 11 11 3 4 10 16 3 13 5 10 12 19 6 2 3		0 3 1 4 2 1 4 3 2 4 3 5 2 7 1 7 4 8 5 3 6 1 7 10 10 2 10 4

Note

In the first test case, the forest has one vertex and no edges. VegiCat assigns the value 1 to the vertex, and VegiDog outputs 0.

In the last test case, the permutation is $p = [5, 3, 6, 4, 2, 9, 1, 8, 7, 10]$.

VegiCat assigns $a_7 = 16$. Since $p_7 = 1$, VegiDog receives $b_1 = 16$.

The original edge $(3, 7)$ becomes $(p_3, p_7) = (6, 1)$.

The original edge $(4, 10)$ becomes $(p_4, p_{10}) = (4, 10)$. The sample outputs this edge as $(10, 4)$, which represents the same undirected edge.

The sample illustrates the input and output formats. Its outputs are correct for the graphs and permutations shown. However, the sample assignments are not guaranteed to be part of a strategy that works for every valid forest and every permutation.

J Joint Relocation

Time limit: 4s



VegiCat owns a company with n employees in k nonempty teams. The employees live along a straight road, represented by a number line. VegiCat wants to arrange two meeting points for them.

VegiCat divides the teams into two sets, A and B, with each team in exactly one set. He chooses a meeting point for each set at any real coordinate. **All employees on the same team must go to the same meeting point.** Each team in set A goes to the meeting point for A, and each team in set B goes to the meeting point for B. Either set may be empty.

Let S_A list the home positions of all employees on teams in A, and define S_B similarly for B. Each employee is counted separately, even when home positions repeat.

For a list $S = (x_1, x_2, \ldots, x_m)$ of home positions, the minimum total distance for these employees to travel to one meeting point is

$$M(S) = \min_{p \in \mathbb{R}} \sum_{j=1}^{m} |x_j - p|.$$

Define $M(S) = 0$ when S is empty. Help VegiCat minimize $M(S_A) + M(S_B)$ over all ways to divide the teams into the two sets.

Input

The first line contains an integer T ($1 \le T \le 3 \cdot 10^5$), the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 3 \cdot 10^5$), denoting the number of employees and the number of teams, respectively.

Each of the next k lines describes one team. The i-th line contains an integer m_i ($1 \le m_i \le n$), the number of employees on team i, followed by m_i integers $x_{i,1}, \ldots, x_{i,m_i}$ ($1 \le x_{i,j} \le 10^7$), their home positions. The sum of all m_i in a test case is n.

Home positions may repeat and may be given in any order.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output an integer — the minimum possible total travel distance.

Sample Input 1	Sample Output 1
2 4 2 2 1 3 2 8 10 8 3 3 4 12 1 2 3 2 3 9 8 10	4 15

Note

In the first test case, VegiCat sends team 1 to a meeting point at position 1 and team 2 to a meeting point at position 8. The employees in each team travel a total distance of 2, giving the answer 4.

In the second test case, VegiCat chooses meeting points at 3 and 9. He sends teams 1 and 2 to the first meeting point, and team 3 to the second. The employees in teams 1, 2, and 3 travel total distances of 12, 1, and 2, respectively, giving the answer 15.

This page is intentionally left blank.

K Keep Your Distance

Time limit: 1s



VegiCat is on a mine-sweeping mission. There are n mines on a number line. Mine i is at the integer coordinate a_i. The mines are numbered in order of their coordinates, so $a_1 \le a_2 \le \cdots \le a_n$. Mine coordinates may repeat.

VegiCat tests each mine once, in the order $1, 2, \ldots, n$. For mine i, he chooses an integer coordinate b_i as the test location, which may be negative or zero. The error of this test is $|a_i - b_i|$.

For each $2 \le i \le n$, an integer l_i is given, specifying the minimum required distance between the test locations for mines $i-1$ and i. These locations must satisfy

$$|b_i - b_{i-1}| \ge l_i.$$

Help VegiCat minimize the largest test error; that is, find

$$\min_{\substack{b_1,\ldots,b_n \in \mathbb{Z} \\ |b_i - b_{i-1}| \ge l_i \quad (2 \le i \le n)}} \left(\max_{1 \le i \le n} |a_i - b_i| \right).$$

Input

The first line contains an integer T ($1 \le T \le 10^4$), the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($2 \le n \le 10^5$), the number of mines.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the coordinate of mine i. It is guaranteed that $a_1 \le a_2 \le \cdots \le a_n$.

The third line contains $n-1$ integers $l_2, l_3, \ldots, l_n$ ($1 \le l_i \le 10^9$), where l_i is the minimum distance between the test locations for mines $i-1$ and i.

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

Output

For each test case, output an integer — the minimum possible value of the largest test error.

Sample Input 1	Sample Output 1
2 2 1 1 1 7 2 6 12 14 14 16 19 11 8 4 2 10 9	1 6

Note

In the first test case, choosing $b_1 = 1$ and $b_2 = 2$ gives a maximum error of 1. An error of 0 would require $b_1 = b_2 = 1$, which violates the distance requirement.

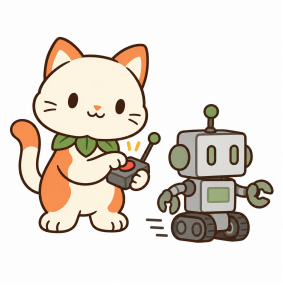
In the second test case, one optimal choice is $b = [-4, 7, 15, 10, 8, 22, 13]$. All required distances are satisfied, and the maximum error is 6.

This page is intentionally left blank.

L Long Slides

Time limit: 2s

This problem has an attachment. You can use it to simulate and visualize the movements of the robot.



VegiCat has a robot on an infinite two-dimensional grid. He wants it to move from cell $s = (s_x, s_y)$ to cell $t = (t_x, t_y)$. Each cell has integer coordinates. The cells s and t may be the same.

The robot follows a string p of instructions. Each character is `L`, `R`, `U`, or `D`. Let $|p|$ denote the length of p.

The robot starts at s. Before it executes any instructions, you may place at most $|p|$ obstacles in distinct cells. The coordinates (x, y) of each obstacle must satisfy $-2 \cdot 10^9 \le x, y \le 2 \cdot 10^9$. Neither s nor t may contain an obstacle. Each obstacle occupies one cell and remains fixed throughout the process; all other cells are free of obstacles.

The robot executes the instructions $p_1, p_2, \ldots, p_{|p|}$ in order. For each instruction, starting from its current cell (x, y), it moves along the following sequence of cells:

- `L`: $(x, y) \to (x-1, y) \to (x-2, y) \to \cdots$;
- `R`: $(x, y) \to (x+1, y) \to (x+2, y) \to \cdots$;
- `U`: $(x, y) \to (x, y+1) \to (x, y+2) \to \cdots$;
- `D`: $(x, y) \to (x, y-1) \to (x, y-2) \to \cdots$.

The robot repeatedly takes one step in the specified direction while the next cell contains no obstacle. It stops in the cell immediately before the first obstacle in that direction. If the next cell in the specified direction already contains an obstacle before the instruction starts, the robot stays in place, but the instruction is still completed. If there is no obstacle ahead in that direction, the robot would move forever, and the placement is invalid.

The robot may reach t before all instructions are completed. It **must still execute all remaining instructions**. Construct any obstacle placement satisfying the requirements above such that the robot finishes at t after the last instruction, or report that no such placement exists.

Input

The first line contains an integer T ($1 \le T \le 10^5$), the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers s_x and s_y ($-10^9 \le s_x, s_y \le 10^9$), the coordinates of the starting cell.

The second line contains two integers t_x and t_y ($-10^9 \le t_x, t_y \le 10^9$), the coordinates of the target cell.

The third line contains the instruction string p ($1 \le |p| \le 10^5$), consisting only of the characters `L`, `R`, `U`, and `D`.

It is guaranteed that the sum of $|p|$ over all test cases does not exceed 10^5.

Output

For each test case, output an answer in input order. Obstacle placements for different test cases are independent.

If no valid placement exists, output `NO` on a line by itself.

Otherwise, output `YES` on the first line. On the second line, output an integer k ($0 \le k \le |p|$), the number of obstacles.

Then output k lines. The i-th of these lines must contain two integers x_i and y_i ($-2 \cdot 10^9 \le x_i, y_i \le 2 \cdot 10^9$), specifying an obstacle in cell (x_i, y_i).

All k coordinate pairs must be distinct, and none may equal s or t. The placement must allow every instruction to finish and leave the robot at t after the last instruction. If there are multiple solutions, output any of them.

You can output the yes or no verdict in any case. For example, `yEs`, `yes`, `Yes`, and `YES` are all accepted as positive responses.

Sample Input 1	Sample Output 1
6 0 0 1 0 RL 0 0 -1 0 URD 0 0 8 0 RLR 0 0 0 0 UDUD 0 0 2 5 RDULR 0 0 1 1 RURURU	NO NO YES 2 -1 0 9 0 YES 2 0 -1 0 1 YES 5 -1 5 2 -1 2 6 3 0 3 5 YES 3 1 2 2 0 2 1

M Minimum Records

Time limit: 1s

VegiCat has n closed integer intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$.



He wants to construct a sequence $a_1, a_2, \ldots, a_n$ by choosing an integer a_i from each interval $[l_i, r_i]$. That is, $l_i \le a_i \le r_i$ for every $1 \le i \le n$.

A position i is called a **record position** if $a_i > a_j$ for every $1 \le j < i$. In particular, position 1 is always a record position.

Help VegiCat minimize the number of record positions. Output this minimum number.

Input

The first line contains an integer T ($1 \le T \le 10^4$), the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 10^6$), the number of intervals.

The i-th of the next n lines contains two integers l_i, r_i ($-10^9 \le l_i \le r_i \le 10^9$), describing the interval available at position i.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, output an integer — the minimum possible number of record positions.

Sample Input 1	Sample Output 1
2 4 1 1 1 100 20 20 30 30 8 -3 -1 -5 4 0 1 2 2 1 8 5 6 10 12 12 12	2 4

Note

In the first test case, one optimal sequence is $a = [1, 100, 20, 30]$. Its record positions are 1 and 2.

In the second test case, one optimal sequence is $a = [-1, 4, 1, 2, 8, 6, 12, 12]$. Its record positions are 1, 2, 5, and 7. Position 8 is not a record because its value equals the maximum seen so far.

This page is intentionally left blank.