The 2026 ICPC Asia Wuhan Regional Contest

Official Contest





Problem List

A	It Is Late at Night, Remember to Rest
B	When Does BPE Stop Doubling?
C	Congruence Equation of Puro
D	Travel Plan
E	Coffee Record
F	Skill Activation Order
G	Shuffle Playlist
H	Galaxy War
I	Karma Gate
J	Number Game
K	Salary Adjustment
L	Jigsaw Puzzle
M	Empire of Construction

This booklet contains 13 problems on 28 pages.
Please notify a volunteer immediately if your booklet is incomplete.

November 1, 2026

Hosted & Problem Set Prepared by



武漢大學
WUHAN UNIVERSITY



武漢大學
ACM/ICPC 协会（学生）

Problem A. It Is Late at Night, Remember to Rest

Rose is a lovely girl, and she is messing around in the ICPC Wuhan Regional Contestants Group.



Figure 1: The group-chat log.

There are n messages in the group chat, each of which is a string consisting only of lowercase letters. All messages fall into three categories:

- `yeshenle` (meaning "it is late at night, remember to rest"), which can only appear between 00:00 and 00:59.
- `zaoshanghao` (meaning "good morning, another wonderful day has begun"), which can only appear between 07:00 and 07:59.
- Other messages, which may appear at any time of a day.

Note that only messages that **exactly match** `yeshenle` or `zaoshanghao` have restrictions on when they may appear; messages such as `yeshenlexiuxi` and `zaoshang` may appear at any time of a day.

Define the range of a day as 00:00 to 23:59. Now, given these n messages in chronological order by sending time, please help her find the minimum number of days that these messages span.

Input

The first line contains an integer n ($1 \le n \le 10^5$), indicating the number of messages.

The next n lines each contain a string s ($1 \le |s| \le 15$) consisting only of lowercase letters, giving all messages in chronological order.

Output

Output one line containing an integer, indicating the minimum number of days that these messages span.

Example

standard input	standard output
6 yeshenle welcome yeshenle zaoshanghao balloons yeshenle	2
3 cloud lightbulb balloon	1

Note

The first sample is shown in the image in the statement. It can be proved that, according to the rules in the statement, these messages cannot be sent within the same day, so they span at least two days.

Problem B. When Does BPE Stop Doubling?

A tokenizer is an important natural language processing tool that splits a continuous text string into tokens that a model can use for training and inference. Byte-Pair Encoding is an algorithm for training a tokenizer.

Now there is a corpus used for tokenizer training. In the training system, it will be represented as d token strings. The training system denotes a token string in the form $[a_1, a_2, ..., a_n]$, where a_i is the token ID of the i-th token in the string, and n is the length of the string.

Initially, the i-th string in the corpus is $\underbrace{[0, 0, \ldots, 0]}_{n_i \text{ zeros}}$. Here, $1 \le n_i < 2^{200\,000}$.

The *frequency* of a token string is defined as the total number of times it occurs as a contiguous substring among all strings in the corpus. For example, suppose the corpus contains two strings $[0, 0, 0]$ and $[0, 0, 1]$; then the *frequency* of $[0, 0]$ is 3.

Training of Byte-Pair Encoding is performed in rounds. The j-th round (numbered starting from 1) performs the following operations in order:

1. Choose the **length-2** token string with the current maximum *frequency*. If multiple length-2 token strings tie for the maximum *frequency*, choose the **lexicographically smallest one**†. If there is no length-2 token string with *frequency* greater than 0, terminate training and do not perform the remaining steps of this round and the next round.
2. Let the chosen string be $[x_j, y_j]$. The algorithm creates the j-th *merge rule*, defined as follows: for a string, **repeatedly** replace the current leftmost contiguous substring $[x_j, y_j]$ by $[j]$ until no contiguous substring $[x_j, y_j]$ exists in the string.
   For example, when applying the *merge rule* "replace $[0, 0]$ by $[1]$" to the string $[0, 0, 0, 0, 0]$, it becomes $[1, 1, 0]$.
3. Apply the j-th *merge rule* to every corpus string.

A *merge rule* is called a *self-merge rule* if $x_j = y_j$, and a *mixed rule* otherwise.

Determine whether a *mixed rule* will appear before training stops. If it will, also find four numbers k, x_{k+1}, y_{k+1}, f, with the following meanings:

- The first k *merge rules* are all *self-merge rules*;
- The $(k+1)$-th *merge rule* is the first *mixed rule* in training; it processes $[x_{k+1}, y_{k+1}]$;
- f is the *frequency* of $[x_{k+1}, y_{k+1}]$ in the entire corpus when the $(k+1)$-th rule is selected.

† $[x, y]$ is lexicographically smaller than $[x', y']$ if and only if $x < x'$, or $x = x'$ and $y < y'$.

Input

There are multiple test cases.

The first line contains an integer T ($1 \le T \le 2 \times 10^5$), the number of test cases.

For each test case:

- the first line contains an integer d ($1 \le d \le 2 \times 10^5$), the number of token strings in the corpus.
- The next d lines follow. The i-th line contains an integer n_i ($1 \le n_i < 2^{200\,000}$), given in its **binary representation without leading zeros**.

It is guaranteed that the sum of the **bit lengths** of n_i over all test cases does not exceed 2×10^5.

Output

For each test case, output one line.

If there is never a *mixed rule* during training, output one line containing a single integer -1.

Otherwise, output one line containing four integers k, x_{k+1}, y_{k+1}, f, whose meanings are given above.

Example

standard input	standard output
6 1 11 4 1 10 100 1000 1 111 3 11 11 100 8 1101011 10010 101 1101 10001 10000 111010 10111 7 11 1001 1001 100001 100001 100001 100001	1 1 0 1 -1 2 1 0 1 1 1 0 2 5 1 0 2 4 4 0 4

Note

For the first test case of the sample, $n_1 = 3$, the only corpus string is $[0, 0, 0]$.

In the first round, the only length-2 string that occurs is $[0, 0]$, and it occurs 2 times, so its *frequency* is 2. The first rule merges $[0, 0]$ into $[1]$; after applying it from left to right, the corpus string becomes $[1, 0]$.

In the second round, the only length-2 string that occurs is $[1, 0]$, with *frequency* 1, so the second rule chooses $[1, 0]$. It is the first *mixed rule*. Before it, there is exactly 1 *self-merge rule*, so the answer is `1 1 0 1`.

Problem C. Congruence Equation of Puro

Puro found a congruence equation in the library: $x_1^a + x_2^a + \cdots + x_n^a \equiv r \pmod{p}$, where the variables satisfy $x_i \in \{0, 1, \ldots, p-1\}$. Of course, Puro knows how to solve this equation, but now he wants to know the number of solutions to this equation, that is, the number of n-tuples $(x_1, x_2, \ldots, x_n)$ satisfying the above conditions. Since Dr. K's computer screen is too small to display this astronomical number, you only need to output the number of solutions modulo $10^9 + 7$.

Input

The first line contains three integers p, a, r ($2 \le p \le 10^7$, $1 \le a \le 300$, $0 \le r < p$). It is guaranteed that p is prime.

The second line contains an integer n ($1 \le n < 10^{300}$).

Output

Output one line containing an integer, indicating the number of valid n-tuples modulo $10^9 + 7$.

Example

standard input	standard output
3 2 2 2	4
2 297 1 9876543210	216768474
97 10 0 2	193

Note

For the first sample, there are four valid pairs (x_1, x_2) satisfying $x_1^2 + x_2^2 \equiv 2 \pmod{3}$: $(1, 1), (1, 2), (2, 1), (2, 2)$. Thus, the output is 4.

This page is intentionally left blank.

Problem D. Travel Plan

Rose is a girl who enjoys traveling. She has arrived in a new city.

The city has n attractions, numbered $1 \sim n$, connected by m two-way roads. Each road connects two distinct attractions, and there is at most one road between any pair of attractions. Any attraction can be reached from any other attraction via a sequence of roads. In other words, all attractions and roads in the city form a simple connected undirected graph with n vertices and m edges.

Rose wants to start from one attraction and visit all attractions in depth-first search (DFS) order. Specifically, for every attraction u, she arranges the vertices reached by each road starting from u into an ordered list $\mathrm{adj}(u)$ in some order. She then starts from the selected starting vertex s and travels according to the following pseudocode:

Algorithm 1 Depth-First Search

```
function DFS(u, d)
    visited[u] ← true
    depth[u] ← d
    for each v ∈ adj(u) do
        if ¬visited[v] then
            dfs-tree ← dfs-tree ∪ {(u, v)}
            DFS(v, d + 1)
        end if
    end for
end function

dfs-tree ← ∅
visited[1 . . . n] ← false
depth[1 . . . n] ← 0
DFS(s, 0)
```

After the traversal, she wants the depth of every attraction u in the DFS tree (namely, $\mathrm{depth}[u]$ in the pseudocode) to be exactly a_u.

Determine whether there exists a plan satisfying the requirements. If so, construct a suitable s and $\mathrm{adj}(u)$.

Input

The first line contains two integers n, m ($2 \le n \le 2 \times 10^5$, $1 \le m \le 2 \times 10^5$), indicating the number of attractions and roads.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i < n$), indicating the required depth of every attraction in the DFS tree.

The next n lines describe the graph. The i-th of them starts with an integer d_i ($1 \le d_i < n$), indicating the number of roads starting from vertex i, followed by d_i integers $\mathrm{adj}'(i,1), \mathrm{adj}'(i,2), \cdots, \mathrm{adj}'(i,d_i)$ ($1 \le \mathrm{adj}'(i,j) \le n$), indicating the vertices reached by each road starting from vertex i.

It is guaranteed that $\sum_{i=1}^{n} d_i = 2m$, and that all attractions and roads form a simple connected undirected graph.

Output

If there is no solution, output one line containing an integer -1.

Otherwise, output an integer s on the first line, indicating the selected starting vertex; then output n lines. On the i-th of them, output d_i integers $\mathrm{adj}(i,1), \mathrm{adj}(i,2), \cdots, \mathrm{adj}(i,d_i)$ separated by single spaces, indicating the ordered list corresponding to vertex i. $\mathrm{adj}(i)$ must be a permutation of $\mathrm{adj}'(i)$.

Example

standard input	standard output
5 7 2 0 1 3 1 3 3 4 2 4 1 4 5 3 3 4 1 2 3 1 3 2 1 2	2 4 3 2 5 3 1 4 1 4 2 1 3 2 2
5 7 1 0 2 2 1 3 5 2 4 3 4 1 5 1 5 3 2 5 1 4 1 2 3 4	-1

Note

The DFS tree obtained in the first sample is shown below. The bold edges are DFS tree edges, and the number on an edge near a vertex u indicates the position of that edge in the ordered list adj(u).

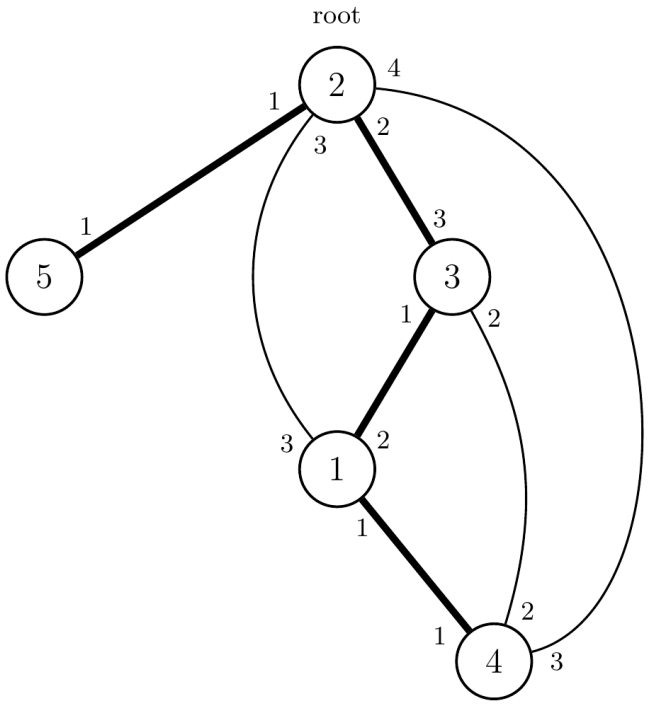


Figure 2: The DFS tree in the first sample.

Problem E. Coffee Record

This is an interactive problem.

Dew assigns a lowercase English letter to every cup of coffee she drinks, then concatenates these letters in the order in which she drinks the coffee. To look back on her coffee experiences, she wants to choose some consecutive cups from her record and divide them, in their original order, into three groups to form a *coffee collection*.

She sets the following requirements and scoring rule. Let the current record be a string s. A *coffee collection* consists of a **contiguous substring** P of s, together with a way to split P into three nonempty strings A, B, C, so that

$$P = A + B + C \,^{\dagger}$$

The strings P, A, B, C must all be **palindromes**‡. The score of a *coffee collection* satisfying this requirement is the number of cups in its smallest group, namely

$$\min(|A|, |B|, |C|)$$

where $|X|$ denotes the length of a string X.

The *optimal score* of the current record is the maximum score among all *coffee collections* satisfying the requirement. If there is no such collection, the *optimal score* is 0.

Dew would like you to help her find the *optimal score* of the current record after each character is appended to it. However, she does not want to reveal her coffee record to you directly, so you can only obtain the information you need by asking her questions.

Specifically, the string s is initially empty, and the interaction consists of n rounds. In each round, one lowercase English letter is appended to the end of s; in round t, $|s| = t$. The contents of s are hidden from you. You may use the following two types of queries to obtain information:

- Given l, r, ask whether the substring $s_l \cdots s_r$ is a palindrome.
- Given r, x, ask whether there is a palindromic substring ending at position r whose length is at least x.

In each round, you may make **at most three queries in total across both types**, after which you must report the *optimal score* of the current s.

† For two strings A and B, $A + B$ denotes the concatenation of A and B in this order. For example, `butterfly` + `dew` = `butterflydew`.

‡ A string $u = u_1 u_2 \cdots u_m$ of length m is a **palindrome** if and only if $u_i = u_{m-i+1}$ for all $1 \le i \le m$. For example, `a`, `abba`, and `abcba` are palindromes, while `abca` is not.

Interaction Protocol

There are multiple test cases.

The first line of input contains an integer T ($1 \le T \le 100$), indicating the number of test cases.

For each test case, the first line of input contains an integer n ($1 \le n \le 10^4$), indicating the number of interaction rounds.

The interaction then proceeds in rounds. In each round, first make **at most three queries in total**, then report one answer. In round t, the current string is $s = s_1 s_2 \cdots s_t$. The two types of queries have the following formats:

- Print `? 1` l r, where $1 \le l \le r \le t$. The interactor returns 1 if $s_l \cdots s_r$ is a palindrome, and 0 otherwise.
- Print `? 2` r x, where $1 \le x \le r \le t$. The interactor returns 1 if there exists $1 \le l \le r - x + 1$ such that $s_l \cdots s_r$ is a palindrome, and 0 otherwise.

To report your answer, print `!` a, where a is the *optimal score* of the current s. After flushing the output

buffer, the round ends immediately. Proceed to the next round if $t < n$; otherwise, read n for the next test case, or terminate your program if there is none.

It is guaranteed that the sum of n over all test cases does not exceed 10^4.

The interactor is adaptive. The character appended in each round may be chosen based on the interaction and is fixed at the end of that round. All query responses and the verdict on your reported answer in that round are based on the string obtained by appending this character.

Print each operation on a separate line and flush the output buffer. You can flush the output buffer as follows:

- C++: `fflush(stdout)` or `cout.flush()`;
- Java: `System.out.flush()`;
- Python: `sys.stdout.flush()`.

Example

standard input	standard output
1 10 1 0 1 0 1 1	 ! 0 ? 1 1 2 ! 0 ? 2 3 3 ! 0 ! 0 ! 1 ? 2 6 5 ! 2 ! 2 ! 2 ! 2 ? 1 2 7 ? 2 8 5 ? 1 3 4 ! 2

Note

In the sample, one possible final string is `aabbaabbaa`.

In round 2, the query `? 1 1 2` returns 1, because $s_1 s_2 =$ `aa` is a palindrome.

In round 3, the current string is `aab`. The query `? 2 3 3` returns 0, because the only substring ending at position 3 with length at least 3 is `aab`, which is not a palindrome.

In round 6, the query `? 2 6 5` returns 1, because $s_1 \cdots s_6 =$ `aabbaa` is a palindrome of length 6 ending at position 6, and its length is at least 5.

In the last round, the current string is `aabbaabbaa`, and three queries are made:

- `? 1 2 7` returns 0, because $s_2 \cdots s_7 =$ `abbaab` is not a palindrome.
- `? 2 8 5` returns 1, because $s_3 \cdots s_8 =$ `bbaabb` is a palindrome of length 6 ending at position 8, and its length is at least 5.
- `? 1 3 4` returns 1, because $s_3 s_4 =$ `bb` is a palindrome.

One can choose the entire string and split it into

$$\texttt{aabbaa} + \texttt{bb} + \texttt{aa}$$

The entire string and all three parts are palindromes. The score is $\min(6, 2, 2) = 2$, which is the *optimal score*.

This page is intentionally left blank.

Problem F. Skill Activation Order

Dew writes a program to generate a skill activation order for a team in *BanG Dream!*. However, an off-by-one error prevents the program from inserting a removed member at the end of the list.

The team consists of n members, numbered from 1 to n. Initially, the list is $L = (1, 2, \ldots, n)$. All indices in the list are 0-based.

The program performs n iterations in order. In iteration i ($0 \le i < n$), it performs the following operations:

1. Remove the member x **currently at index** i from the list.
2. Choose an integer j satisfying $0 \le j \le n-2$. Insert x into the shortened list so that its index becomes j. The member previously at index j in the shortened list, and all subsequent members, are shifted one position to the right.

After each iteration, the list still contains n members. In particular, the removed member **cannot be inserted at the end of the list.**

Given a permutation P, determine the number of integer sequences $(j_0, j_1, \ldots, j_{n-1})$ for which the final list is exactly P, where j_i is the insertion index chosen in iteration i. Output the answer modulo 998 244 353.

Two sequences are considered different if they differ in at least one insertion index.

Input

There are multiple test cases.

The first line of the input contains an integer T ($1 \le T \le 1\,000$), indicating the number of test cases.

For each test case:

- The first line contains an integer n ($2 \le n \le 500$), indicating the number of members.
- The second line contains n distinct integers $p_1, p_2, \cdots, p_n$ ($1 \le p_i \le n$), indicating the target permutation.

It is guaranteed that the sum of n over all test cases does not exceed 2 000.

Output

For each test case, output one line containing an integer, indicating the number of valid sequences modulo 998 244 353.

Example

standard input	standard output
5 2 2 1 2 1 2 3 1 3 2 3 3 2 1 5 2 1 3 5 4	1 0 2 2 19

Note

In the first test case of the sample, the only valid sequence of insertion indices is $(0, 0)$.

In the third test case of the sample, the two valid sequences are $(0, 1, 1)$ and $(1, 0, 1)$. For the latter sequence, the list changes as follows:

$$(1, 2, 3) \longrightarrow (2, 1, 3) \longrightarrow (1, 2, 3) \longrightarrow (1, 3, 2)$$

Problem G. Shuffle Playlist

During the Card Incident, Haniyasushin Keiki collected n types of ability cards. The card types are numbered from 1 to n, and there are a_i cards numbered i. In addition, Tenkyu Chimata gave her k blank cards.

Now Keiki wants to play her carefully selected song collection in the Primate Spirit Garden. The song collection contains n songs, numbered $1, 2, \ldots, n$. When playing song i, if there is still a card numbered i, consume one; otherwise, if there is a blank card, consume one blank card; if neither is available, stop playback.

To avoid playing the same song twice in a row, a playlist is generated from this song collection according to the following random shuffle rule:

- At the beginning, the n songs form a playlist in an arbitrary order, and are played in that order;
- When the last song in the current playlist is about to start playing, if it is playable, the current playlist is immediately replaced by a new playlist, in which this song is the first, and the remaining $n - 1$ songs are arranged arbitrarily after it.

Assume that all shuffles turn out in the most favorable way for Keiki. Please help her find the maximum number of songs she can play.

Input

There are multiple test cases.

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

For each test case:

- The first line contains two integers n, k ($2 \le n \le 10^5$, $0 \le k \le 10^{12}$), indicating the number of card types and the number of blank cards, respectively.
- The second line contains n positive integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^{12}$), indicating the number of cards of each type.

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

Output

For each test case, output one line containing an integer, indicating the maximum number of songs Keiki can play in the optimal case.

Example

standard input	standard output
8 3 0 1 1 1 4 0 4 4 4 1 4 1 4 4 4 1 2 3 147154220721 1 4 9 1 6 11 16 3 1 1 5 4 3 1000000000000 1 5 4 6 2 2 1 1 1 1 1	3 9 14 9 41 10 1000000000010 9

Note

In the first test case of the sample, there are no blank cards. Suppose the initial playlist is $[1, 2, 3]$. First, songs 1 and 2 are played. When song 3 is about to start playing, the playlist is updated to $[3, 1, 2]$ (or $[3, 2, 1]$). Then song 3 is played. After song 3 is played, the next song is song 1 (or song 2), but the corresponding card has already been consumed earlier, so playback stops. A total of 3 songs are played.

In the second test case of the sample, one possible optimal case is as follows: the playlist changes as $[1, 2, 3, 4] \to [4, 1, 2, 3] \to [3, 1, 2, 4]$, where the final song 4 in the last playlist cannot be played, and a total of 9 songs are played. Note that when playback reaches the last song of the list, if it cannot be played, playback stops immediately and the playlist is not updated.

In the third test case of the sample, in the optimal case, the second time song 4 is played, a blank card is consumed.

Problem H. Galaxy War

Rose dreamed that she had become the commander-in-chief of an interstellar war.

The battlefield is a square in which both coordinates lie in $[-10^4, 10^4]$. There are n civilizations on the battlefield. For civilization i, its base C_i is a circle centered at (x_i, y_i) with radius r_i. These bases may intersect, but no two of them have the same center.

Light-particle strikes are a common form of attack in interstellar warfare. A light particle can be regarded as a point traveling at the speed of light (a constant). It is launched from an orbit along the boundary of a base, and its trajectory is a **tangent** to the base boundary at the launch point. Therefore, for a point P that is not inside any base, the time required for civilization i to attack P is proportional to the length of a tangent segment from P to circle C_i. The civilization that requires the least time to attack P is called the lord of point P, and P belongs to that civilization's *territory*. **Points inside any base do not belong to the *territory* of any civilization.**

Rose belongs to civilization 1. Please determine the area of the *territory* of civilization 1.

Input

The first line contains an integer n ($2 \le n \le 10^5$), indicating the number of civilizations.

The next n lines each contain three real numbers x_i, y_i, r_i ($-10^4 < x_i, y_i < 10^4$, $0 < r_i \le 10^4$), indicating the base of civilization i.

Every input real number has at most three digits after the decimal point. All bases are entirely contained within the battlefield. Bases may overlap, but no two bases have the same center.

Output

Output one line containing a real number, indicating the area of the *territory* of civilization 1.

Your answer is considered correct if and only if its absolute or relative error from the standard answer does not exceed 10^{-4}.

Example

standard input	standard output
4 1.000 1.000 2.000 4.000 5.000 2.000 -2.000 0.000 1.000 5.000 -7.000 3.000	31.965189405482097
2 0.000 0.000 1000.000 0.000 4000.000 1000.000	236858407.346410206761537
4 1.000 -1.000 1.000 2.000 -1.000 2.000 0.000 0.000 2.000 -3.000 -2.000 4.000	0.000000000000000

standard input	standard output
8 15.000 -3.000 7.000 13.000 -4.000 2.000 14.000 -8.000 3.000 22.000 1.000 4.000 42.000 7.000 1.000 -1.000 0.000 2.000 -3.000 0.000 1.000 0.000 -15.000 5.000	163.354738705171682

Note

The exact answer for the first sample is $\frac{718205}{16128} - 4\pi$.

Problem I. Karma Gate

Slugcat is traveling through the map of Rain World. The map consists of several regions, and the regions are connected by some karma gates, which form an undirected graph. Slugcat found a mural on some hanging structure, which depicts a map of another world and records the following problem that you need to help solve:

You are given an undirected graph with n vertices and m edges. The vertices are numbered from 1 to n, and the edges are numbered from 1 to m in input order. The graph has no self-loops or multiple edges.

For an interval $[l, r]$ $(1 \le l \le r \le m)$ on the sequence of edge indices, let $k = r - l + 1$. It is called *valid* if there exist $k + 1$ **pairwise distinct** vertices $p_0, p_1, \ldots, p_k$ such that the edges with indices in $[l, r]$ can be rearranged and oriented as $(p_0, p_1), (p_1, p_2), \ldots, (p_{k-1}, p_k)$. In other words, an interval is *valid* if all edges in this interval **form exactly a simple path**.

Find the number of *valid* intervals.

Input

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

For each test case:

- The first line contains two integers n and m $(2 \le n \le 10^5, 1 \le m \le 10^5)$, the numbers of vertices and edges.
- The next m lines: the i-th line contains two integers u_i and v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$, denoting an undirected edge. There is at most one edge between each pair of distinct vertices.

It is guaranteed that both the sum of n and the sum of m over all test cases do not exceed 10^5.

Output

For each test case, output an integer, the number of *valid* intervals.

Example

standard input	standard output
3 7 7 1 2 3 4 2 3 4 5 1 5 5 6 6 7 7 8 4 5 5 6 4 7 6 7 3 5 1 2 2 3 1 3 4 3 1 2 2 3 3 4	16 17 6

Note

In the first test case of the sample, all 7 intervals of length 1 are *valid*. The other *valid* intervals are:

- Length 2: $[2, 3]$, $[4, 5]$, $[5, 6]$, and $[6, 7]$;
- Length 3: $[1, 3]$, $[2, 4]$, and $[5, 7]$;
- Length 4: $[1, 4]$ and $[2, 5]$.

Therefore, the answer is $7 + 4 + 3 + 2 = 16$.

For example, the edges in $[1, 4]$, taken in the order $1, 3, 2, 4$, form the simple path $1 \to 2 \to 3 \to 4 \to 5$. The edges in $[1, 5]$ form a cycle, so this interval is *invalid*.

Problem J. Number Game

Rose is playing a number game.

She writes n non-negative integers $a_1, a_2, \cdots, a_n$ on the blackboard, which form a multiset A.

In each operation, she may choose some numbers on the blackboard and change them **simultaneously** to the mex† of the multiset they form.

She also has a multiset $B = \{b_1, b_2, \cdots, b_n\}$ as the goal of this number game. Please determine whether she can transform the numbers on the blackboard from A into B using a finite number of operations.

† For a multiset S consisting of non-negative integers, define $\operatorname{mex}(S)$ as the smallest non-negative integer that does not belong to S. For example, $\operatorname{mex}\{1, 2, 3\} = 0$ and $\operatorname{mex}\{0, 1, 1, 4, 5\} = 2$.

Input

There are multiple test cases.

The first line of the input contains an integer T ($1 \le T \le 2 \times 10^5$), indicating the number of test cases.

For each test case:

- The first line contains an integer n ($1 \le n \le 2 \times 10^5$), indicating the sizes of the multisets A, B.
- The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i \le 10^9$), indicating the multiset A.
- The third line contains n integers $b_1, b_2, \cdots, b_n$ ($0 \le b_i \le 10^9$), indicating the multiset B.

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

Output

For each test case, output one line containing a string `YES` or `NO`. If A can be transformed into B using a finite number of operations for the corresponding test case, print `YES`; otherwise, print `NO`. The output is case-insensitive.

Example

standard input	standard output
4 3 1 2 3 2 2 3 3 0 0 0 2 3 3 3 3 3 2 2 3 3 4 10000 10000 10000 10000 2 3 3 3	YES NO YES YES

Note

In the first test case of the sample, $\{1, 2, 3\}$ can be transformed into $\{2, 2, 3\}$ by the following operations: (choose the subset formed by the boxed elements)

- $\{1, \boxed{2}, 3\} \to \{1, 0, 3\}$.
- $\{\boxed{1}, \boxed{0}, 3\} \to \{2, 2, 3\}$.

This page is intentionally left blank.

Problem K. Salary Adjustment

You are an employee of Incredibly Clever Pig Corporation (ICPC).

There are n employees in the corporation, numbered $1, 2, \ldots, n$. The superior-subordinate relationships among the employees form a rooted tree with employee 1 as the root. For $i = 2, 3, \ldots, n$, the direct superior of employee i is employee f_i. It is guaranteed that $1 \le f_i < i$.

The corporation is about to adjust the salaries of all its employees. The current salary of employee i is a_i, and the target salary is b_i. The whole adjustment process lasts exactly n days: each day, exactly one employee who has not been adjusted yet is chosen, and on that day the salary of the chosen employee is changed directly from its current value to its target value. Therefore, the adjustment process corresponds to a permutation $p = (p_1, p_2, \ldots, p_n)$ of 1 to n, where p_k denotes the employee adjusted on the k-th day.

However, the corporation has strict rules about salaries. Specifically, if employee u is a superior of employee v, then at any moment the salary of employee u **must not be lower than** the salary of employee v. It is guaranteed that both the salaries a_i before the adjustment and the salaries b_i after the adjustment satisfy this rule.

A permutation p corresponding to an adjustment process is *legal* if and only if this rule still holds after each day's adjustment is completed.

Please find the number of *legal* permutations p, modulo 998 244 353.

Input

There are multiple test cases.

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

For each test case:

- The first line contains an integer n ($2 \le n \le 2 \times 10^5$), indicating the number of employees.
- The second line contains $n - 1$ integers $f_2, f_3, \cdots, f_n$ ($1 \le f_i < i$), where f_i indicates the direct superior of employee i.
- The third line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$), indicating the current salary of each employee.
- The fourth line contains n integers $b_1, b_2, \cdots, b_n$ ($1 \le b_i \le 10^9$), indicating the target salary of each employee.

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

Output

For each test case, output one line containing an integer, indicating the number of *legal* permutations p modulo 998 244 353.

Example

standard input	standard output
2 4 1 1 2 5 3 4 2 7 6 5 4 8 1 1 1 2 5 5 3 6 4 3 5 4 3 4 2 9 5 7 7 2 1 1 5	4 1680

Note

In the first test case of the sample, the superior-subordinate relationships are as follows: employee 1 is the direct superior of employees 2 and 3, and employee 2 is the direct superior of employee 4. The current salaries are $(5, 3, 4, 2)$, and the target salaries are $(7, 6, 5, 4)$.

The four *legal* permutations p are $(1, 2, 3, 4)$, $(1, 2, 4, 3)$, $(1, 3, 2, 4)$, and $(3, 1, 2, 4)$.

For example, when $p = (1, 2, 3, 4)$, the salaries change as follows:

$$(5, 3, 4, 2) \longrightarrow (7, 3, 4, 2) \longrightarrow (7, 6, 4, 2) \longrightarrow (7, 6, 5, 2) \longrightarrow (7, 6, 5, 4)$$

After the adjustment on each of these days is completed, the salary of a superior is not lower than the salary of a subordinate.

Problem L. Jigsaw Puzzle

This is an interactive problem.

You have a box of jigsaw puzzle pieces, containing n pieces numbered $1, 2, \ldots, n$. These pieces can be assembled into one row, and in that row the i-th piece from the left is numbered a_i. **This is the only correct arrangement.** But you have lost the reference picture of the jigsaw puzzle, so you **do not know** what the permutation a is.

Luckily, you are an expert at jigsaw puzzles. When you arrange these n pieces into one row, for every pair of adjacent pieces in the row you can tell whether they *fit together*. Here *fit together* means that the two pieces are adjacent in the correct arrangement, and their left-to-right order is unchanged. Therefore you decide to determine the permutation a using several *placements*, and a *placement* goes as follows:

1. Choose a permutation b. Arrange the n pieces into one row so that the i-th piece from the left is numbered b_i.
2. You will then obtain a binary string of length $n-1$, namely $c = c_1 c_2 \ldots c_{n-1}$. Here:
   - $c_i = 1$ if and only if there exists $1 \le j \le n-1$ such that $b_i = a_j$ and $b_{i+1} = a_{j+1}$;
   - otherwise, $c_i = 0$.

You have to determine the permutation a using at most $n-1$ *placements*, and finally *report* it. The step of *reporting* the permutation a does not count toward the number of *placements*.

Interaction Protocol

There are multiple test cases.

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

For each test case, the first line of input contains an integer n ($2 \le n \le 200$), indicating the number of pieces.

After that, you may perform *placements* and finally *report* your answer:

- *Placement*: print `?` b_1 b_2 ... b_n. Here $b_1, \ldots, b_n$ must be a permutation. After flushing the output buffer, you have to read a binary string of length $n-1$, namely $c = c_1 c_2 \ldots c_{n-1}$, which is the result of this *placement*.
- *Report*: print `!` a_1 a_2 ... a_n. You declare that the unknown permutation is $a_1, \ldots, a_n$. This operation does not count toward the number of *placements*. After flushing the output buffer, the current test case ends immediately. You then proceed to the next test case, or terminate your program if there is none.

You may perform at most $n-1$ *placements*, and you may *report* your answer at any moment within this limit.

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

The interactor is non-adaptive. The permutation a is fixed before the interaction starts and does not change with your *placements*.

Print each operation on a separate line and flush the output buffer. You can flush the output buffer as follows:

- C++: `fflush(stdout)` or `cout.flush()`;
- Java: `System.out.flush()`;
- Python: `sys.stdout.flush()`.

Example

standard input	standard output
2 4 010 000 111 5 0001 0010 0000 0111	 ? 1 2 3 4 ? 4 3 2 1 ? 1 4 2 3 ! 1 4 2 3 ? 1 3 5 4 2 ? 3 2 1 4 5 ? 1 2 3 4 5 ? 5 3 1 4 2 ! 3 1 4 2 5

Note

For the first test case of the sample, the correct arrangement is $a = (1, 4, 2, 3)$.

In the first *placement*, you choose $b = (1, 2, 3, 4)$ and obtain $c =$ `010`: among all the adjacent pieces, only pieces 2 and 3 *fit together*.

In the second *placement*, you choose $b = (4, 3, 2, 1)$ and obtain $c =$ `000`: none of the three adjacent pairs *fit together*.

In the third *placement*, you choose $b = (1, 4, 2, 3)$ and obtain $c =$ `111`: all three adjacent pairs *fit together*, so this arrangement is exactly the correct arrangement.

Finally, you *report* that the correct arrangement is $a = (1, 4, 2, 3)$.

Problem M. Empire of Construction

Alice was the old king of the Kingdom of Construction. A year ago, rebels exiled him to the kingdom's border. Rebel leader Bob then proclaimed himself emperor and renamed the kingdom the Empire of Construction. After a year of effort, Alice raised an army at the border and successfully fought his way back to the capital. At the city wall, Construction Emperor Bob was waiting for him. They decided to play a game to determine who would inherit the throne of the Empire of Construction; as Alice's strategist, you must help him win the game as quickly as possible.

You are given a positive integer n. There are n positions arranged in order on a circle, numbered from 0 to $n-1$. Alice selects a positive integer k ($1 \le k \le n$) as the maximum number of positions he may operate on in each round. Once k is chosen, it cannot be changed throughout the game.

The game is played in rounds. During the game, each position contains a number, which is either `0` or `1`. Before the first round begins, Bob selects a binary string s of length n, uses it as the initial value at each position, and keeps it secret.

Each round proceeds in the following order:

1. Alice selects at most k distinct positions, possibly selecting none.
2. Alice selects one of the following two options:
   - Flip all selected positions. Flipping changes `0` to `1` and `1` to `0`.
   - Assign each selected position a value of `0` or `1`, independently.

   Assignment and flipping cannot both be performed in the same round. The operations on all selected positions take effect simultaneously, and unselected positions remain unchanged.
3. Bob selects any non-negative integer x and moves the number originally at position i to position $(i+x) \bmod n$. The numbers at all positions move simultaneously. Bob does not tell Alice which x he selects.

At any time, including before the first round, if all positions are `0` or all positions are `1`, Bob immediately loses the game, and the game ends.

Before the game begins, Alice must determine a finite sequence of operations, specifying which positions are selected and which operation is performed in each round. **All rounds containing assignment operations must occur strictly before all rounds containing flip operations.** In addition, Bob knows the entire sequence in advance. The sequence is called *feasible* if Bob is guaranteed to have lost by the time the sequence finishes, no matter what initial string s he chooses and what x he chooses in each round.

Complete the following objectives in order:

1. Find the minimum integer k for which a *feasible* operation sequence exists.
2. Subject to k being this minimum value, find the minimum number of rounds t, and construct a shortest *feasible* operation sequence.

Input

The input contains only one integer n ($2 \le n \le 30$).

Output

Print two integers k and t on the first line: the minimum position limit and the corresponding minimum number of rounds.

Then print t lines describing a shortest *feasible* operation sequence that uses this k. Each line is a string of length n that uses only the characters `-F01`. For $0 \le i < n$, the $(i+1)$-th character from the left specifies the operation at position i:

- `-` means that the position is not selected;
- `F` means that the position is flipped;
- `0` means that the position is assigned the value `0`;
- `1` means that the position is assigned the value `1`.

Each line contains at most k characters other than `-`. No string may contain both `F` and `0`, or both `F` and `1`. Every line containing `0` or `1` must occur strictly before every line containing `F`.

If multiple shortest solutions exist, output any one of them.

It can be proven that, under the constraints of this problem, the answer t does not exceed 10^5.

Example

standard input	standard output
2	1 1 -F
4	2 6 0-0- F-F- -F-- F-F- -FF- -F-F

Note

For the first sample, Bob's initial string s can only be one of `00`, `11`, `01`, and `10`. If it is `00` or `11`, Bob loses immediately when the game begins; if it is `01` or `10`, Bob loses after the first round.