The 2nd Universal Cup

Stage 23: Shanghai

February 17-18, 2024

This problem set should contain 13 problems on 25 numbered pages.

Based on

International Collegiate Programming Contest (ICPC)

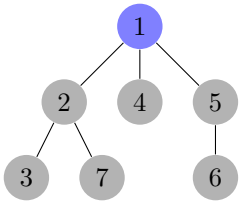
Hosted by

Problem A. DFS Order 4

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Little Cyan Fish, also known as Qingyu Xiao, loves the concept of DFS order. Today, he has a rooted tree T with n vertices labeled from 1 to n. The root of the tree is vertex 1, and the parent of vertex i ($2 \le i \le n$) is vertex f_i ($1 \le f_i < i$).

A DFS order $D = (D_1, D_2, \cdots, D_n)$ represents the sequence of nodes visited during a depth-first search of the tree. A vertex appearing at the j-th position in this order (where $1 \le j \le n$) indicates that it is visited after $j-1$ other vertices. During the depth-first search, if a vertex has multiple children, they are visited in **ascending order** of their indices. Thus, in this problem, each rooted tree has a **unique** DFS order.



A tree with 7 vertices. The DFS Order of the tree is $[1, 2, 3, 7, 4, 5, 6]$.

The following pseudocode describes a way to generate the DFS order given a rooted tree T. T is uniquely represented by the array $f = \{f_2, \ldots, f_n\}$. The function GENERATE() returns the DFS order starting at the root vertex 1:

Algorithm 1 An implementation of the depth-first search algorithm

```
1:  procedure DFS(vertex x)
2:      Append x to the end of dfs_order
3:      for each child y of x do          ▷ Children are iterated in ascending order of index.
4:          DFS(y)
5:      end for
6:  end procedure
7:  procedure GENERATE()
8:      Let dfs_order be a global variable
9:      dfs_order ← empty list
10:     DFS(1)
11:     return dfs_order
12: end procedure
```

Let D be the array returned by GENERATE(). There are $(n-1)!$ different possible configurations for the array f, each representing a distinct tree T. Little Cyan Fish wonders: for all these $(n-1)!$ configurations of f, how many distinct DFS orders D can be generated? We consider two DFS orders D and D' to be different if and only if there exists an index $1 \le i \le n$ such that $D_i \ne D'_i$. Given that the number can be very large, your task is to compute this number modulo a given prime integer P.

Input

The first line of the input contains two integers n and P ($1 \le n \le 800$, $10^8 \le P \le 1.01 \times 10^9$).

It is guaranteed that P is a prime number.

Output

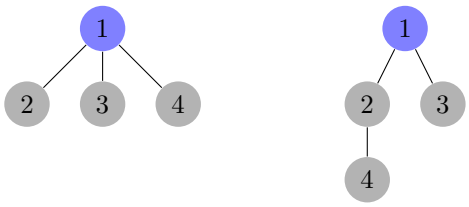
Output a single line containing a single integer, indicating the answer.

Examples

standard input	standard output
4 114514199	2
10 998244353	11033
100 1000000007	270904395

Note

In the first example, there are two distinct DFS orders: $D_1 = [1, 2, 3, 4]$ and $D_2 = [1, 2, 4, 3]$, which can be obtained by $T_1 : f_2 = 1, f_3 = 1, f_4 = 1$ and $T_2 : f_2 = 1, f_3 = 1, f_4 = 2$, respectively.



T_1 T_2

Problem B. Roman Master

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Little Cyan Fish, also known as Qingyu Xiao, loves all types of numbers. Today, he is learning the Roman numerals. Roman numerals are a numeral system that originated in ancient Rome and remained the usual way of writing numbers throughout Europe well into the Late Middle Ages. In this numeral system, numbers are written with combinations of letters from the Latin alphabet, each letter with a fixed integer value. In this problem, we will only consider the digits 1 to 8, and they are written in the following forms.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
I	II	III	IV	V	VI	VII	VIII

For a given string consisting of only letters `I` and `V`, Little Cyan Fish would like to convert it to a single integer. To do that, he will first decompose the string into several separated substrings, so that each substring represents a digit from 1 to 8. Then, he will write down the digits in order to get the integer. For example, for the string `VIIIV`, Little Cyan Fish can decompose it into `VII` and `IV`. By looking up in the table above, the two substrings represent the digit 7 and the digit 4. He then gets the integer 74.

Now, Little Cyan Fish is wondering, for a given string S, what will be the minimum integer he could get by performing the process above. Please help him to find it!

Input

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \le T \le 10^5$), indicating the number of test cases.

For each test case, the input contains a single line containing a single string S ($1 \le |S| \le 10^5$, S contains only letters `I` and `V`).

It is guaranteed that the sum of $|S|$ over all test cases will not exceed 10^6.

Output

For each test case, output a single line containing a single integer, indicating the answer.

Example

standard input	standard output
3 II IVI VIIIIV	2 16 634

Note

In the first test case, the Roman numeral `II` can be decomposed to [`I`, `I`] or [`II`], which corresponds to the integers 11 and 2, respectively. Therefore, the answer is 2.

In the second test case, the Roman numeral `IVI` can be decomposed to [`I`, `V`, `I`], [`IV`, `I`], and [`I`, `VI`], which corresponds to the integers 151, 41, and 16, respectively. Therefore, the answer is 16.

Problem C. Equal Sums

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

After learning Elegia's mind on how to use generating function tricks to solve combinatorial counting problems, Little Cyan Fish would like to solve the following problem.

Little Cyan Fish has $n+m$ integers, denoted by $x_1, x_2, \cdots, x_n$ and $y_1, y_2, \cdots, y_m$. Little Cyan Fish knows that:

- $l_i^{(x)} \le x_i \le r_i^{(x)}$ for all $1 \le i \le n$.
- $l_j^{(y)} \le y_j \le r_j^{(y)}$ for all $1 \le j \le m$.

Little Cyan Fish has not decided the exact value of x_i and y_j. He is wondering, for any pair of integers (a, b) $(1 \le a \le n,\ 1 \le b \le m)$, how many ways there are to set the value of x_i $(1 \le i \le a)$ and y_j $(1 \le j \le b)$, such that the sum $\sum_{i=1}^{a} x_i$ equals $\sum_{j=1}^{b} y_j$. As the number can be quite large, you only need to output it modulo 998 244 353.

Input

The first line of the input contains two integers n and m $(1 \le n, m \le 500)$.

The next n lines describe the constraints of the array $x_1, x_2, \cdots, x_n$. The i-th line of these lines contains two integers $l_i^{(x)}$ and $r_i^{(x)}$ $(1 \le l_i^{(x)} \le r_i^{(x)} \le 500)$, indicating a constraint.

The next m lines describe the constraints of the array $y_1, y_2, \cdots, y_m$. The j-th line of these lines contains two integers $l_j^{(y)}$ and $r_j^{(y)}$ $(1 \le l_j^{(y)} \le r_j^{(y)} \le 500)$, indicating a constraint.

Output

Output n lines, each of which contains m integers. The b-th integer in the a-th line indicates the number of ways to set the values of $x_1, x_2, \cdots, x_a$ and $y_1, y_2, \cdots, y_b$ so that $\sum_{i=1}^{a} x_i = \sum_{j=1}^{b} y_j$, modulo 998 244 353.

Example

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

Note

For $a = 1$ and $b = 1$, there are two ways to set the values, as follows:

x_1	y_1
1	1
2	2

For $a = 2$ and $b = 1$, there are three ways to set the values, as follows:

x_1	x_2	y_1
1	2	3
1	3	4
2	2	4

For $a = 2$ and $b = 2$, there are four ways to set the values, as follows:

x_1	x_2	y_1	y_2
1	2	1	2
1	3	2	2
2	2	2	2
2	3	3	2

For $a = 2$ and $b = 3$, there are four ways to set the values, as follows:

x_1	x_2	y_1	y_2	y_3
1	3	1	2	1
2	2	1	2	1
2	3	1	2	2
2	3	2	2	1

Problem D. Random Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Little Cyan Fish is engaged in an online competitive programming course called "Big-God-Live-Class" taught by Prof. **B**. In a recent lesson, Prof. **B** introduced a trick for finding the median in a sequence using two priority queues. Inspired by this method, Little Cyan Fish decides to implement it for practice.

Specifically, Little Cyan Fish has written a program that reads a permutation $p_1, p_2, \ldots, p_n$ of length n from an input file. The program's objective is to calculate the sum of the medians of all consecutive subsequences $p_l, p_{l+1}, \cdots, p_r$ for every $1 \le l \le r \le n$.

To test if his program is correct, Little Cyan Fish would like to **uniformly generate a random permutation** of $1, 2, \cdots, n$, and calculate the sum. He plans to compare the sum calculated by his program with the correct answer. Now, your task is to help him to find the correct answer.

Recall that the median of a list of numbers $a_1, a_2, \cdots, a_t$ is defined as the $\lfloor \frac{t}{2} \rfloor$-th smallest number in the list. For instance, the median of $[1, 5, 4]$ is 4, and the median of $[11, 4, 5, 14]$ is 5.

Input

The first line of the input contains a single integer n ($1 \le n \le 3 \times 10^5$).

The next line of the input contains n integers $p_1, p_2, \cdots, p_n$ ($1 \le p_i \le n$), indicating the permutation. It is guaranteed that p is generated by uniformly choosing one of the $n!$ permutations (by a pseudorandom number generator).

There are at most 20 test files. (Note that each test file contains only a single test case as described above.)

Output

Output a single line containing a single integer, indicating the answer.

Example

standard input	standard output
4 1 4 2 3	22

Note

- The median of $[p_1] = [1]$ is 1.
- The median of $[p_2] = [4]$ is 4.
- The median of $[p_3] = [2]$ is 2.
- The median of $[p_4] = [3]$ is 3.
- The median of $[p_1, p_2] = [1, 4]$ is 1.
- The median of $[p_2, p_3] = [4, 2]$ is 2.
- The median of $[p_3, p_4] = [2, 3]$ is 2.
- The median of $[p_1, p_2, p_3] = [1, 4, 2]$ is 2.
- The median of $[p_2, p_3, p_4] = [4, 2, 3]$ is 3.

- The median of $[p_1, p_2, p_3, p_4] = [1, 4, 2, 3]$ is 2.

Therefore, the answer equals $1 + 4 + 2 + 3 + 1 + 2 + 2 + 2 + 3 + 2 = 22$.

Problem E. Colorful Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Having delved into the complex theory of quantum chromodynamics, Little Cyan Fish has become fascinated with the concept of *color charge*. To test your understanding of this theory, he has proposed the following task to you.

Little Cyan Fish presents you with a graph consisting of N vertices. Each vertex is assigned a color, with colors numerically labeled from 1 to n. The number of vertices assigned color i is exactly a_i, such that the total number of vertices is $\sum_{i=1}^{n} a_i = N$. Initially, the graph contains no edges.

Your task is to add edges to this graph under a specific constraint: any two vertices of the same color must have a distance between them that is greater than or equal to d. This constraint is to avoid having vertices of the same color too close to each other. Notably, if there is no path between a pair of vertices, their distance is considered infinite.

Your goal is to maximize the number of edges in the graph while adhering to the above constraint. Determine and output the maximum number of edges that can be added to the graph under these conditions.

Input

There are multiple test cases in a single test file. The first line of the input contains a single 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 and d ($1 \le n \le 5 \times 10^5$, $1 \le d \le n$).

The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$ for all $1 \le i \le n$; $\sum_{i=1}^{n} a_i \le 10^9$).

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

Output

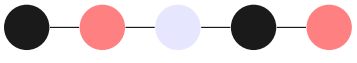
For each test case, output a single line containing a single integer, indicating the answer.

Example

standard input	standard output
4 3 3 2 2 1 3 3 2 3 3 5 2 1 1 1 1 1 1 1 1	4 7 10 0

Note

In the first example, at most 4 edges can be added to the graph. Here is a possible plan.



Problem F. Dot Product

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

This is another story about Kevin, a friend of Little Cyan Fish.

Kevin is the chief judge of the International Convex Polygon Championship (ICPC). He proposed a geometry task for the contest. However, since he is inexperienced in computational geometry, he couldn't generate a correct convex polygon for the tests of the task.

Now, Little Cyan Fish is teaching Kevin about how to use the cross product to check if a given polygon is convex. Kevin learned it so well. After that, Kevin decided to learn more knowledge about other forms of the product, especially the dot product.

Please note that the definition of the dot product in this problem is nonstandard: Consider two arrays of length n, denoted by $a_1, a_2, \cdots, a_n$ and $b_1, b_2, \cdots, b_n$. The dot product $c = a \cdot b$ results in a new array of length n, where each element $c_i = a_i \cdot b_i$. For instance, $[0, 8] \cdot [1, 5] = [0, 40]$ and $[2, 3] \cdot [4, 5] = [8, 15]$.

Little Cyan Fish gives Kevin a permutation $a_1, a_2, \cdots, a_n$ of the integers $1, 2, \cdots, n$ and another permutation defined by $b_i = i$. The challenge is to modify the permutation a by swapping adjacent elements so that the sequence resulting from $a \cdot b$ is monotonically non-decreasing. For example, the permutation $a = [2, 1, 3]$ satisfies this condition since $a \cdot b = [2, 2, 9]$. However, the permutation $[3, 2, 1]$ does not meet the criteria, as $a \cdot b = [3, 4, 3]$ is not monotonically non-decreasing.

You are given the permutation a. Can you help Kevin calculate the minimum number of swap operations required to achieve the goal set by Little Cyan Fish?

Input

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \le T \le 10^5$), indicating the number of test cases.

For each test case, the first line of the input contains one positive integer n ($1 \le n \le 5 \times 10^5$).

The next line of the input contains n integers $a_1, \cdots, a_n$ ($1 \le a_i \le n$) denoting the permutation a. It is guaranteed that $a_1, \ldots, a_n$ form a permutation, i.e., $a_i \neq a_j$ for $i \neq j$.

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

Output

Output a single line containing a single integer, indicating the answer.

Example

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

Problem G. Flow 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Little Cyan Fish, alongside his teammate Qingyu Xiao from Peking University, is engrossed in an algorithm class. Their current topic of interest is the paper *Maximum Flow and Minimum-Cost Flow in Almost-Linear Time*[1], and the paper *All-Pairs Max-Flow is no Harder than Single-Pair Max-Flow: Gomory-Hu Trees in Almost-Linear Time*[2]. Little Cyan Fish is particularly fascinated by the maximum flow problem and seeks your assistance in tackling a related challenge. Before diving into the problem, let's revisit some fundamental concepts from the textbook to ensure a clear understanding.

We call an undirected graph $G(V, E)$ a simple graph if and only if it does not have multiple edges or self-loops. For a simple graph $G(V, E)$ with $|V| \geq 3$, let s and t be any two distinct vertices (called **source** and **sink**, respectively). A **flow** is a map $f : E \mapsto \mathbb{R}_{\geq 0}$ that satisfies the following:

- **(Capacity constraints)**: the flow of an edge cannot exceed 1, in other words, $0 \leq f_{uv} \leq 1$ for all $(u, v) \in E$.
- **(Conservation of flows)**: The sum of the flows entering a node must equal the sum of the flows exiting that node, except for the source and the sink. In other words,

$$\forall v \in V \setminus \{s, t\} : \sum_{u:(u,v)\in E} f_{uv} = \sum_{u:(v,u)\in E} f_{vu}$$

Please note that, in this problem, the capacity of all the edges equals 1. In other words, **the edges in the given graph are all unweighted**.

The **value of flow** is the amount of flow passing from the source to the sink. Formally for a flow $f : E \mapsto \mathbb{R}_{\geq 0}$ it is given by:

$$|f| = \sum_{v:(s,v)\in E} f_{sv} = \sum_{u:(u,t)\in E} f_{ut}$$

The maximum s, t-flow, denoted by $\text{maxflow}(s, t)$, is the maximum value of all possible flows f with the source s and sink t. Specifically, we define $\text{maxflow}(u, u) = 0$ for all $u \in V$.

Now, Little Cyan Fish is doing the assignment for the algorithm class. The assignment gives Little Cyan Fish a simple undirected graph $G(V, E)$ with n vertices, labeled from 1 to n. His task is to compute the matrix $A = F(G)$, where $A_{u,v} = \text{maxflow}(u, v)$ for every $u, v \in V$. Finding this straightforward, Little Cyan Fish is more intrigued by the inverse problem: Given a matrix $A_{u,v}$, how can one construct an undirected simple graph G such that $F(G) = A$?

Of course, this problem is really hard. So Little Cyan Fish will give you some simpler matrices — the matrix consisting **only of the integers** $\{0, 1, 2, 3\}$. In this case, can you solve the challenge by Little Cyan Fish?

[1]by Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva, in 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), available at arXiv:2203.00671

[2]by Amir Abboud, Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak, in 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)

Input

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \le T \le 10^3$), indicating the number of test cases.

For each test case, the first line of the input contains a single integer n ($1 \le n \le 300$).

The next n lines describe the matrix A. The i-th ($1 \le i \le n$) line of these lines contains n integers $A_{i,1}, A_{i,2}, \cdots, A_{i,n}$. It is guaranteed that $0 \le A_{i,j} \le 3$ for all $1 \le i, j \le n$.

It is guaranteed that the sum of n^2 over all test cases does not exceed 9×10^6.

Output

For each test case, if there is no such graph G satisfying $F(G) = A$, print a single line "No".

Otherwise, the first line of the output contains a single line "Yes". The next line of the output contains a single integer m, indicating the number of edges you used. The next m lines should contain two integers x and y, indicating an edge. You need to make sure that your solution is a simple graph without multiple edges or self-loops.

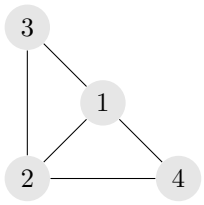
If there are multiple solutions, you may print any of them.

Example

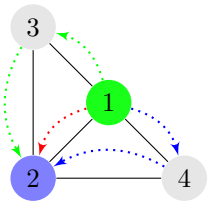
standard input	standard output
4 4 0 3 2 2 3 0 2 2 2 2 0 2 2 2 2 0 8 0 2 2 0 0 1 1 1 2 0 2 0 0 1 1 1 2 2 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 2 2 1 1 1 0 0 2 0 2 1 1 1 0 0 2 2 0 3 0 1 2 1 2 3 2 3 1 12 0 2 2 2 2 2 2 2 2 1 1 1 2 0 2 2 2 2 2 2 2 1 1 1 2 2 0 2 2 2 2 2 2 1 1 1 2 2 2 0 2 2 2 2 2 1 1 1 2 2 2 2 0 2 2 2 2 1 1 1 2 2 2 2 2 0 2 2 2 1 1 1 2 2 2 2 2 2 0 2 2 1 1 1 2 2 2 2 2 2 2 0 2 1 1 1 2 2 2 2 2 2 2 2 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0	Yes 5 1 2 3 1 3 2 4 1 4 2 Yes 8 1 2 2 3 3 1 6 7 7 8 8 6 1 6 4 5 No Yes 12 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 1 1 10 10 11 11 12

Note

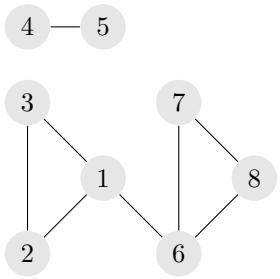
In the first test case, one possible graph is shown in the following figure.



Taking $A_{1,2} = 3$ as an example, the following figure shows that $|f_{\max}| = 3$. So the constraints for maxflow$(1, 2)$ is satisfied.



In the second test case, one possible graph is shown in the following figure.



In the third test case, it is obvious that $A_{u,u} = 0$ was not satisfied. Therefore, there was no possible graph corresponding to this matrix.

Problem H. Map 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Noah has played an RPG game in which he steps on the back of an enormous moose to find a secret shipyard. The wayfinding system was terrible. Noah keeps dreaming about what would happen if the game had proper map navigation.

The map in the game can be represented by a simple polygon M. Noah cannot walk out of M. The shipyard is located at a specific point p inside (or on the boundary of) M. A proper map navigation system should always display the shortest path between p and Noah's current location. Given M, p, and several locations $a_1, \ldots, a_Q$ of Noah, write a program that computes the length of the shortest paths between a_i and p for all $1 \le i \le Q$.

Input

There are multiple test cases in a single test file. The first line contains a single integer T ($1 \le T \le 5000$) – the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($3 \le n \le 5000$) – the number of vertices of the simple polygon, M.

Each of the next n lines contains a vertex of M. The vertices are given in counterclockwise order.

The next line contains the target p.

The next line contains a single integer Q ($1 \le Q \le 5000$) – the number of queries.

Each of the next Q lines contains a query a_i.

Points are given by a pair of coordinates x and y separated by a single space. All coordinates are integers with absolute values no more than 2000. All points in the input are inside or on the boundary of M. It is guaranteed that M is simple, i.e., its vertices are distinct and no two edges of the polygon intersect or touch, other than consecutive edges which touch at their common vertex. The sum of n over all test cases does not exceed 5000. The sum of Q over all test cases does not exceed 5000.

Output

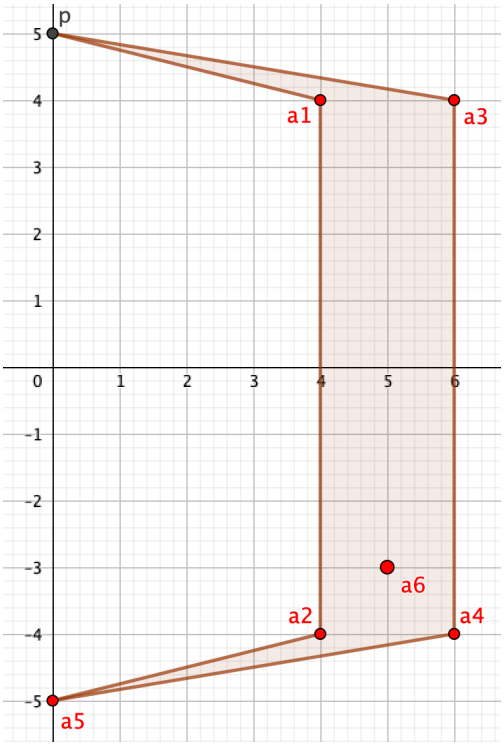
For each test case, output Q lines. The i-th line should contain the length of the shortest path between a_i and p. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Example

standard input	standard output
1 6 0 5 4 4 4 -4 0 -5 6 -4 6 4 0 5 6 4 4 4 -4 6 4 6 -4 0 -5 5 -3	4.123105625618 12.123105625618 6.082762530298 12.369316876853 16.246211251235 11.194173437483

Note

Here is the illustration for the example.



Problem I. Balance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Little Cyan Fish is honing his skills in balancing, under the guidance of Qingyu Xiao. Now, Qingyu Xiao prepared a connected undirected graph $G = (V, E)$ with n vertices, labeled from 1 to n, and m edges, denoted by (u_i, v_i) $(1 \le i \le m)$. Each vertex i was labeled by a number a_i. The graph **may contain multiple edges between the same pair of vertices**, but there will be no self-loops.

The task for Little Cyan Fish is to achieve a state of balance in the graph. Specifically, his goal is to make

$$\sum_{(u,v)\in E} |a_u - a_v| \le \max_{1\le i\le n}\{a_i\} - \min_{1\le i\le n}\{a_i\}$$

. To do this, he can swap the labels between any pairs of vertices any number of times.

Your task is to help Little Cyan Fish determine if it is possible to meet the goal. Furthermore, if it is possible, you need to provide a possible solution.

Input

There are multiple test cases in a single test file. The first line of the input contains a single 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 and m $(2 \le n \le 10^5, n - 1 \le m \le 2 \times 10^5)$.

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

The next line contains n integers $a_1, \ldots, a_n$ $(1 \le a_i \le n)$ indicating the sequence a.

The graph **may contain multiple edges between the same pair of vertices**. It does not contain any self-loop.

It is guaranteed that the sum of n over all test cases is no more than 5×10^5 and that the sum of m over all test cases is no more than 10^6.

Output

For each test case, output "Yes" in the first line if there is a possible way to rearrange a in the first line, or "No" otherwise. If it is possible, output a possible solution $a_1, a_2, \ldots, a_n$ in the next line.

Example

standard input	standard output
5 5 4 1 2 2 3 3 4 4 5 1 2 3 4 5 5 4 1 2 1 3 1 4 1 5 1 2 3 4 5 5 4 1 2 1 3 1 4 1 5 1 2 2 2 3 5 6 1 2 1 2 2 3 3 4 4 5 3 5 1 2 1 2 1 2 2 1 2 1 2 1 2	Yes 5 4 3 2 1 No Yes 2 2 2 3 1 Yes 2 2 1 1 1 No

Problem J. Travel 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

Little Cyan Fish, a zealous traveler, is on a mission to circumnavigate the world of fish consisting of n cities, numbered from 1 to n. Starting at city 1, he wishes to map out all m bidirectional roads connecting these cities. It is guaranteed that it is possible to travel to any city by using the roads. Furthermore, each road connects two distinct cities, and no two cities have more than one road between them. In other words, the cities and roads can be treated as a simple and connected undirected graph. However, without a map, Little Cyan Fish is unaware of the roads' existence, so he has absolutely no idea which roads exist in this world. Therefore, his goal is to find the information on all the roads.

To accomplish this, Little Cyan Fish can walk in this world. When he stands in a city, Little Cyan Fish can see the index x of the current city and the number of roads connected to the city x, d_x. He can then choose to walk through the i-th road ($1 \le i \le d_x$) from city x. The order of roads from each city is unknown but fixed. After the walking, Little Cyan Fish will be standing at the other endpoint of the road he choose, and report to you the information of the new city, i.e., its index and the number of roads connected to it.

Your goal is to help Little Cyan Fish find a strategy to find all the roads by repeating the process above. Since Little Cyan Fish only has a limited time, and the world travel would consume a large amount of time, he can only walk no more than $(2m + 2n)$ times. Please help him!

Interaction Protocol

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \le T \le 1\,000$), indicating the number of the test cases.

For each test case, you will not know the exact value of n and m — the interaction process begins immediately. It is guaranteed that $1 \le n \le 2\,500$ and $1 \le m \le 10^4$.

At each round of the interaction, the interactor will provide the index x, indicating the city that Little Cyan Fish stands at, and d_x, indicating the number of roads connecting the city x. To make a walk, you need to print "`>` i" on a separate line ($1 \le i \le d_x$), indicating that you would like to walk through the i-th road connecting the city x. After that, the new round of the interaction begins, and the interactor will provide you with the new index and the number of the roads connecting to that city. The number of the queries you made should be no more than $2m + 2n$.

To give your answer, you need to print "`!` x_1 y_1 x_2 y_2 $\cdots$ x_m y_m", indicating all the roads you found. The order of the edges can be arbitrary, you only need to provide one of the possible plans. Printing the answer is not considered a query and does not count toward the $2m + 2n$ limit.

After you propose your answer, if you find all the m roads correctly, the interactor will print a single line `Correct` — then, the next test case will be followed immediately. Otherwise, if the m roads you found are incorrect, the interactor will print a single line `Wrong`, and the interaction process will be terminated. If you find less than m roads, the interactor will wait for you to print more edges. This may result in "Idleness Limit Exceeded".

After printing a query, do **NOT** forget to output end of line and flush the output. To do this, use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, `flush(output)` in Pascal, or `stdout.flush()` in Python.

It is guaranteed that the hidden graph is simple and connected, the sum of n over all test cases will not exceed 10^4, and the sum of m over all test cases will not exceed 4×10^4.

The interactor of the problem is **non-adaptive**, which means that the graph will be fixed in advance,

and it will not be changed based on the interaction.

Example

standard input	standard output
2 1 1 2 1 1 1 Correct 1 3 2 2 1 3 2 2 4 2 1 3 3 1 Correct	 > 1 > 1 ! 1 2 > 3 > 1 > 3 > 2 > 2 > 2 ! 1 2 1 3 1 4 2 4

Problem K. Best Carry Player 4

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

After learning elementary math, Little Cyan Fish has mastered the concept of carry[3], which is a digit that is transferred from one column of digits to another column of more significant digits.

carry	*1*		*1*	
		6	7	6
+		5	1	8
	1	1	9	4

Now, Little Cyan Fish gives two numbers A and B in the base m. For each number, you can permute its digits arbitrarily. After that, you will get two new numbers A' and B', and leading zeros are **allowed** here. What is the maximum number of carries when computing $A' + B'$ in the base m?

Input

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \le T \le 2 \times 10^5$), indicating the number of test cases.

For each test case, the first line contains one integer m ($2 \le m \le 5 \times 10^5$).

The second line contains m integers $a_0, a_1, \ldots, a_{m-1}$ ($0 \le a_i \le 10^9$). a_i indicates the number of occurrences of digit i in A.

The third line contains m integers $b_0, b_1, \ldots, b_{m-1}$ ($0 \le b_i \le 10^9$). b_i indicates the number of occurrences of digit i in B.

It is guaranteed that the sum of m over all test cases is no more than 5×10^5.

Output

For each test case, output one integer indicating the maximum number of carries.

[3]which means "进位" in Chinese

Example

standard input	standard output
5 2 1 2 3 4 3 1 0 1 0 1 0 4 1 0 0 1 1 1 1 1 5 123456 114514 1919810 233333 234567 20050815 998244353 0 0 0 10 5 3 5 3 2 4 2 4 1 5 9 9 8 2 4 4 3 5 3 0	5 1 2 467900 29

Problem L. Binary vs Ternary

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

A *binary string* is a sequence of bits, i.e., "0"s and "1"s. For a binary string S, you can perform the following operations for any number of times:

- Choose a non-empty substring $S[l, r] = S_l S_{l+1} \ldots S_r$, consider it as a ternary (i.e., base 3) integer, and then convert it to the corresponding binary integer. For example, $(101)_3 = (1010)_2$, so you can transform 1**101**10 into 1**1010**10.

Note that the selected substring **may have leading zeros**, but the transformed substring will have no leading zeros. We consider 0 as a proper binary integer without leading zeros. For example, you can transform 01 to 1 because $(01)_3 = (1)_2$. You can also transform 0 to 0 because $(0)_3 = (0)_2$.

Given two binary strings A and B, both starting with the digit "1", you need to determine whether A can be transformed into B in no more than 512 operations. And you need to keep the length of the string no more than 128 during the transformation. If it is possible, print a solution.

Input

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \le T \le 1000$), indicating the number of test cases.

For each test case, the first line contains the string A ($1 \le |A| \le 64$). The second line contains the string B ($1 \le |B| \le 64$).

It is guaranteed that A and B start with the digit "1" and consist of only "1" and "0".

Output

For each test, if it is impossible to transform string A to B, output "-1".

Otherwise, output an integer n ($0 \le n \le 512$) first, indicating the number of steps. In the next n lines, output two integers l, r ($1 \le l \le r$) indicating the substring you choose in each step. The indices are 1-based. r should be no more than the length of the current string.

Example

standard input	standard output
3 1 111 110110 1101010 1111 111111	-1 1 2 4 2 1 3 2 5

Note

In the first test case, it can be shown that there is no possible solution.

In the second test case, for $A = 110110$, we can choose $l = 2$ and $r = 4$ first. Since $A[2, 4] = 101$, and $(101)_3 = (1010)_2$, so 1**101**10 will be changed to 1**1010**10.

Problem M. Circular Route

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Little Cyan Fish is the king of city $\mathcal{H}$ from the Cup Universe. In city $\mathcal{H}$, there is a circular route of length L. There are L stations on the route numbered by $0, \ldots, L-1$ counterclockwise. Station $(p+1) \bmod L$ is located at the point reached by walking counterclockwise one unit distance from station p.

A bus starts at station x at time 0 and moves at a constant speed of 1 along this circular route counterclockwise, looping endlessly. The bus is small, so there can be at most one person on the bus at any moment.

Now, there are n people who want to travel by bus. The i-th person wants to travel from the station s_i to the destination station t_i. Passengers may board the bus when it arrives at their starting station and must remain on board until reaching their destination. They may also choose to not board the bus and wait for it to arrive again in a future loop.

Little Cyan Fish wants to know, starting from time 0, what is the minimum time needed to transport all individuals to their respective destinations? We assume that boarding and disembarking happen instantaneously, without requiring any additional time.

Please answer q queries regarding the shortest time needed for different starting points x.

Input

There are multiple test cases in a single test file. The first line of the input contains a single 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 and L ($1 \le n \le 2 \times 10^5$, $2 \le L \le 10^9$).

The next n lines describe travel plans for n people. The i-th line of these lines contains two integers s_i and t_i ($0 \le s_i, t_i < L, s_i \ne t_i$), indicating the beginning and the ending of the i-th person.

The next line contains an integer q ($1 \le q \le 10^6$).

The next q lines describe the queries. The i-th line of these lines contains one integer x_i indicating the position of the bus at time 0.

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5, and the sum of q over all test cases does not exceed 10^6.

Output

For each test case, output q lines. The i-th line should contain an integer indicating the answer for the starting point x_i.

Example

standard input	standard output
3 2 10 3 7 8 0 4 1 3 5 9 4 4 0 2 2 0 1 3 3 1 1 0 3 1000000000 0 999999999 0 999999999 0 999999999 1 1	9 7 12 11 9 3999999998