UCPC 2026

Union of Clubs for Programming Contests
Summer Contest 2026

Preliminaries

Official Problemset

June 27th, 2026, 14:00 – 17:00

Hosted and Organized by

Union of Clubs for Programming Contests

Sponsored by

Jane Street®

HRT hudson river trading

HYUNDAI MOBIS

CITADEL | CITADEL Securities

MOLOCO

ASTEROMORPH

NEXON

STARTLINK

김동현(kdh9949) 김영현(kipa00) 나정휘(jhnah917) 정재호(cubic)

UCPC 2026

Preliminaries

Problem List

Please verify that the problem set contains a total of 12 problems.

A Physical Education Is a Math Subject 3

B Bakejoon Online Judge

C Hat Trick

D Parentheses Game

E Airport

F Parentheses and Queries

G Laser Tower

H Empty Box

I Paper Factory

J Hard Sorting Problem

K Merchant

L Probabilistic Sorting

All problems have the same memory limit of 1024 MB.

UCPC 2026

Preliminaries

Problem A. Physical Education Is a Math Subject 3

Time Limit 1 second Memory Limit 1024 MB

"I'm late, I'm late!"

Reference image created with generative AI

Minchurl, who attends UCPC Elementary School, was shocked when he looked at the clock this morning. It was already past 8 AM.

Minchurl hurried out of his house and is going to run to the school front gate. At exactly 9 AM, a scary teacher at UCPC Elementary School slams the school front gate shut. Whether Minchurl is late is determined by the time he arrives at the school front gate. Therefore, arriving at the front gate at exactly 9 AM also counts as being late, and he must arrive before then.

Students at UCPC Elementary School have to write a reflection essay if they are late. Minchurl does not want to write one, so he plans to run as fast as possible. However, if he's going to be late no matter how hard he runs, he figures he might as well stop for tteokbokki on the way—he'll have to write the reflection essay anyway.

You are given the remaining distance to the school front gate, the current time, and Minchurl's running speed. Minchurl has good stamina, so he can run at a constant speed until he reaches the school front gate. Determine whether Minchurl can avoid being late if he starts running right now.

Input

The first line contains three integers separated by spaces: the remaining distance D meters to the school front gate, the value M when the current time is 8:M:00 AM (M minutes past 8:00 AM), and Minchurl's running speed V. ($1 \le D \le 10\,000$, $1 \le M \le 59$, $50 \le V \le 300$)

Minchurl's speed is V meters per minute.

It is guaranteed that all numbers given in the input are integers.

Output

If Minchurl can arrive at the school front gate before 9 AM, print `run`.

Otherwise, print `late`.

Example

stdin	stdout
`300 55 100`	`run`
`700 55 100`	`late`
`600 50 60`	`late`

Note

In the first example, the current time is 8:55 AM, and there are 5 minutes left before the school front gate closes.

Minchurl can run at a speed of 100 meters per minute, so it takes him 3 minutes to travel 300 meters. Therefore, he can arrive at the school front gate at 8:58 AM, so he is not late.

In the second example, the current time is 8:55 AM, and there are 5 minutes left before the school front gate closes.

Minchurl can run at a speed of 100 meters per minute, so it takes him 7 minutes to travel 700 meters. Therefore, he cannot arrive at the school front gate before 9 AM, so he is late.

In the third example, the current time is 8:50 AM, and there are 10 minutes left before the school front gate closes.

Minchurl can run at a speed of 60 meters per minute, so it takes him exactly 10 minutes to travel 600 meters. Therefore, he arrives at the school front gate exactly at 9:00 AM. According to the condition in the statement, arriving at 9 AM also counts as being late, so the answer is `late`.

Problem B. Bakejoon Online Judge

Time Limit 2 second Memory Limit 1024 MB

Bakejoon Online Judge is an online judge where users bake and submit bread that satisfies unusual conditions.

As a user of Bakejoon Online Judge, you must submit N **fresh** loaves of bread **all at once** satisfying the following conditions.

- At most one loaf of bread can be baked at a time, and once you start baking a loaf, you cannot stop it midway.
- Each loaf of bread has three characteristics: preparation time P_i, baking time T_i, and freshness duration F_i.
  - Since preparation is needed before baking, the i-th loaf can be baked starting from time P_i.
  - It takes T_i units of time to bake the i-th loaf.
  - The i-th loaf remains fresh for F_i units of time from the moment it is fully baked.

More precisely, if you start baking the i-th loaf at time $t \geq P_i$, then it becomes fully baked at time $t+T_i$, and it remains fresh from time $t+T_i$ to time $t+T_i+F_i$. The loaf is also fresh at the time $t+T_i$ and $t+T_i+F_i$.

Given the characteristics of the N loaves of bread, determine whether it is possible to submit all loaves while they are fresh, and if it is possible, find the earliest time at which they can be submitted.

Input

The first line contains the number of loaves of bread N. $(1 \leq N \leq 200\,000)$

Each of the next N lines contains three integers P_i, T_i, and F_i, separated by spaces: the preparation time, baking time, and freshness duration of each loaf of bread. $(0 \leq P_i, T_i, F_i \leq 10^9)$

Output

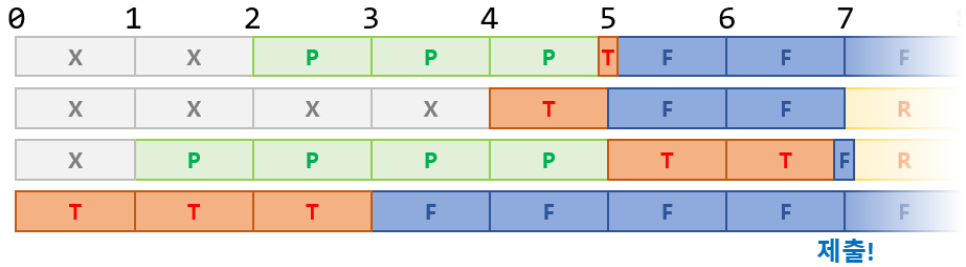
Print the earliest time at which all N loaves of bread can be submitted while they are fresh. If it is impossible to submit all N loaves while they are fresh, print `-1` instead.

Example

standard input (stdin)	standard output (stdout)
4 2 0 4 4 1 2 1 2 0 0 3 7	7
3 0 1 1 1 1 1 2 1 1	-1

Note

In the first example, if the loaves are baked in the following way, all loaves can be submitted while they are fresh at time 7.



In the figure, a gray X means that the loaf is not ready to be baked yet, a green P means that the preparation is complete, a red T means that the loaf is being baked in the oven, a blue F means that the loaf is fresh, and a yellow R means that the loaf is no longer fresh.

In the second example, no matter how the loaves are baked, it is impossible to make all 3 loaves fresh at the same time.

Problem C. Hat Trick

Time Limit 0.25 seconds Memory Limit 1024 MB

This is a two-step problem.

While watching a magic show, Brue saw the following trick.

- The assistant prepares a total of $2N$ hats, of which N are red and N are blue.
- The assistant visits N audience members one by one, in the order from audience member 1 to audience member N, and lets each of them choose and take one hat of any color they want from the remaining hats. The magician does not see this process.
- Afterwards, the assistant arranges the N remaining hats in a row on a long table.
- The magician looks only at the hats on the table and guesses the color of the hat taken by each audience member.

Before the trick begins, the assistant and the magician may agree on how they will act. However, after the trick begins, the only way for the assistant to give information to the magician is by placing hats on the table.

Implement a strategy for the assistant and the magician that makes the trick succeed.

In this problem, your program is executed twice for each test case. In the first execution, it must play the role of the assistant, and in the second execution, it must play the role of the magician.

Input

The first line contains a string R, indicating the role of the current execution. R is either `assistant` or `magician`, meaning that the current execution plays the role of the assistant or the magician, respectively.

The second line contains the number of audience members N. $(1 \le N \le 100)$

If R is `assistant`, the third line contains a string C of length N. C_i is R if audience member i took a red hat, and B if they took a blue hat.

If R is `magician`, the third line contains a string D of length N. D_i is R if the hat placed by the assistant in the i-th position from the left is red, and B if it is blue. This string is identical to the string output by the program playing the role of the assistant in the first execution of the same test case.

Output

If R is `assistant`, output on the first line the colors of the N hats that the assistant will place, from left to right, without spaces. Output R for a red hat and B for a blue hat. The hats placed by the assistant together with the hats taken by the audience members must contain exactly N red hats and exactly N blue hats.

If R is `magician`, output on the first line the colors of the hats guessed by the magician for each audience member, in order from audience member 1 to audience member N, without spaces. Output R for a red hat and B for a blue hat. This output must be exactly equal to the string C given in the first execution.

Example

standard input (stdin)	standard output (stdout)
assistant 4 RBBR	BRBR
magician 4 BRBR	RBBR

Note

The two examples above represent the first and second executions of a single test case, respectively. Note that the string BRBR output in the first execution is the same as the input string in the second execution.

Problem D. Parentheses Game

Time Limit 1 second Memory Limit 1024 MB

You are given an even positive integer N and integers $A_1, A_2, \ldots, A_N$. Alice and Bob play the parentheses game. The game proceeds as follows.

1. Alice rearranges the integers $A_1, A_2, \ldots, A_N$ in any order she wants.
2. Bob looks at the rearranged sequence A and chooses a valid parentheses string S of length N. (The definition of a valid parentheses string can be found at the end of the problem.)

The game score is the sum of A_i for all opening parentheses in S minus the sum of A_i for all closing parentheses. Or in a more formal format,

$$\sum_{i:S_i=(} A_i - \sum_{i:S_i=)} A_i$$

Alice wants to maximize the score, while Bob wants to minimize it. Assuming both Alice and Bob use optimal strategies, determine the final score of the game.

Input

The first line contains an integer N. ($2 \le N \le 200\,000$; N is even)

The second line contains N integers $A_1, A_2, \ldots, A_N$, separated by spaces. ($1 \le A_i \le 10^9$)

Output

Print the final score of the game when both players use optimal strategies.

Example

standard input (stdin)	standard output (stdout)
4 1 2 3 4	2

standard input (stdin)	standard output (stdout)
8 3 3 3 3 3 3 3 3	0

Note

A valid parentheses string is defined as follows.

- The empty string is a valid parentheses string.
- For two valid parentheses strings S and T, their concatenation ST is also a valid parentheses string.

Preliminaries

- For a valid parentheses string S, the string (S) obtained by enclosing S in parentheses is also a valid parentheses string.
- Any string that cannot be obtained by the rules above is not a valid parentheses string.

Problem E. Airport

Time Limit 1 second Memory Limit 1024 MB

There are N cities. However, since no transportation facilities have been built yet, it is currently impossible to travel between cities.

There are M road construction plans. If the i-th plan is chosen, a bidirectional road connecting city u_i and city v_i can be built at a cost of w_i.

In addition, an airport can be built in each city. The cost of building an airport in city i is A_i. Between any two cities with airports, one can travel freely by airplane.

You want to choose some of the road construction plans and some cities in which to build airports, so that for any two cities, it is possible to travel from one city to the other using roads and airplanes. Find the minimum total construction cost required to satisfy this condition.

Input

The first line contains two integers N and M, separated by a space: the number of cities and the number of road construction plans. $(1 \le N \le 10^5; 0 \le M \le 10^5)$

The next M lines contain the information about the road construction plans. Among them, the i-th line contains three integers u_i, v_i, w_i, separated by spaces. This means that a bidirectional road connecting city u_i and city v_i can be built at a cost of w_i. $(1 \le u_i, v_i \le N; u_i \ne v_i; 1 \le w_i \le 10^9)$

The last line contains N integers $A_1, A_2, \ldots, A_N$, separated by spaces. A_i is the cost of building an airport in city i. $(1 \le A_i \le 10^9)$

Multiple road construction plans may connect the same pair of cities.

Output

Output the minimum total construction cost required to make all cities mutually reachable.

Example

standard input (stdin)	standard output (stdout)
4 2 1 2 4 3 4 4 3 10 2 10	13
3 3 1 2 1 2 3 1 1 3 10 100 100 100	2

Note

In the first example, it is optimal to build the road connecting city 1 and city 2, the road connecting city 3 and city 4, and airports in cities 1 and 3. The total cost is $4 + 4 + 3 + 2 = 13$.

In the second example, it is optimal to build no airports and instead build the roads connecting city 1 and city 2, and city 2 and city 3. The total cost is $1 + 1 = 2$.

Problem F. Parentheses and Queries

Time Limit 5 seconds Memory Limit 1024 MB

You are given two sequences a and b of length N. You must process the following Q queries.

- l r k: For each position i in the interval $[l, r]$, choose either (or) so that the resulting string, read from position l to position r, is a valid parentheses string. (The definition of a valid parentheses string can be found at the end of the problem.) If the parenthesis at position i is (, you gain a_i points; if it is), you gain b_i points. Among all ways to choose a parentheses string that maximize the total score, if the parenthesis at position k is the same in every such way, output that parenthesis. If the parenthesis at position k is not uniquely determined, output x.

Input

The first line contains two integers N and Q, separated by a space. ($1 \le N, Q \le 500\,000$)

The second line contains N integers $a_1, a_2, \cdots, a_N$, separated by spaces. ($1 \le a_i \le 10^9$)

The third line contains N integers $b_1, b_2, \cdots, b_N$, separated by spaces. ($1 \le b_i \le 10^9$)

Each of the next Q lines contains three integers l, r, k representing a query, separated by spaces. ($1 \le l \le k \le r \le N$; $r - l + 1$ is even)

Output

Print the answer to each query on a separate line.

Example

standard input (stdin)	standard output (stdout)
6 3 3 6 6 5 1 2 4 3 3 3 2 3 1 4 2 3 6 4 3 6 5	x ( )

Note

A valid parentheses string is defined as follows.

- The empty string is a valid parentheses string.
- For two valid parentheses strings S and T, their concatenation ST is also a valid parentheses string.
- For a valid parentheses string S, the string (S) obtained by enclosing S in parentheses is also a valid parentheses string.
- Any string that cannot be obtained by the rules above is not a valid parentheses string.

Problem G. Laser Tower

Time Limit 2 second Memory Limit 1024 MB

There is one tower at each of the points $x = 1, 2, \ldots, N$ on the horizontal axis. The height of the tower at position $x = i$ is h_i, and $h_1, h_2, \ldots, h_N$ is a **permutation** of the integers from 1 to N.

From the top of each tower, that is, from the point (i, h_i), a laser is fired either to the left or to the right with equal probability.

- If the laser is fired to the left, it covers the rectangular region satisfying $0 \le x \le i$ and $0 \le y \le h_i$.
- If the laser is fired to the right, it covers the rectangular region satisfying $i \le x \le N + 1$ and $0 \le y \le h_i$.

The directions in which the lasers are fired from all towers are determined independently.

Let S be the area of the union of the regions covered after lasers are fired from all towers. Find the expected value of 2^S modulo 998244353.

Input

The first line contains the number of towers N. $(2 \le N \le 200000)$

The second line contains N integers $h_1, h_2, \ldots, h_N$, separated by spaces, representing the heights of the towers. The given heights form a **permutation** in which each integer from 1 to N appears exactly once.

Output

Output the expected value of 2^S modulo 998244353. The number 998244353 is prime.

More specifically, the expected value is guaranteed to be a rational number. If it is written as an irreducible fraction P/Q, it can be shown that $Q \neq 0 \pmod{998244353}$. You must output the value of $P \times Q^{-1} \pmod{998244353}$.

Example

standard input (stdin)	standard output (stdout)
2 1 2	16
3 1 2 3	768
6 6 5 2 3 4 1	448265500

Note

In Example 1, the entire space ranges from $x = 0$ to $x = 3$. For the four possible combinations of directions in which towers 1 and 2 fire their lasers, the area S of the union of the covered regions and the value of 2^S are as follows. Each case occurs with probability $1/4$.

- **Left, left:** The interval $0 \le x \le 2$ is covered up to height 2. $(S = 4,\ 2^S = 16)$
- **Left, right:** The interval $0 \le x \le 1$ is covered up to height 1, and the interval $2 \le x \le 3$ is covered up to height 2. $(S = 3,\ 2^S = 8)$
- **Right, left:** The interval $0 \le x \le 2$ is covered up to height 2, and the interval $2 \le x \le 3$ is covered up to height 1. $(S = 5,\ 2^S = 32)$
- **Right, right:** The interval $1 \le x \le 2$ is covered up to height 1, and the interval $2 \le x \le 3$ is covered up to height 2. $(S = 3,\ 2^S = 8)$

Therefore, the expected value of 2^S is $\frac{16+8+32+8}{4} = 16$.

Problem H. Empty Box

Time Limit 1 second Memory Limit 1024 MB

Nina has an empty box with capacity X. This empty box is unusually long, so a value can be inserted either at the front or at the back, whichever side is desired. Also, this box has the following special property.

If, after inserting a value, the number of values inside the box exceeds X, that is, becomes $X+1$, then the smallest value among the values inside the box is removed. If there are several smallest values, the one closest to the front among them is removed.

Nina does not know the exact capacity X of the box. Instead, suppose she is additionally given the information that $1 \le X \le P$. If, through the following process, it is possible to determine exactly which one of $1, 2, \cdots, P$ the value of X is no matter what the actual value of X is, then this P is called possible.

In other words, if the observed results are always different for any two distinct capacities $X_1, X_2 \in \{1, 2, \cdots, P\}$, then P is possible.

- An integer sequence $A_1, A_2, \cdots, A_N$ of length N and a binary sequence $B_1, B_2, \cdots, B_N$ of length N are given.
- The box is initially empty, and for $i = 1, 2, \cdots, N$, the following operations are performed in order.
  - If $B_i = 0$, insert A_i at the front of the box.
  - If $B_i = 1$, insert A_i at the back of the box.
  - If necessary, remove the smallest value.
  - After that, observe the value at the very front of the box.

Find the maximum possible value of P.

Input

The first line contains an integer N. $(1 \le N \le 10^5)$

The second line contains N integers $A_1, A_2, \cdots, A_N$, separated by spaces. $(1 \le A_i \le N)$

The third line contains N integers $B_1, B_2, \cdots, B_N$, separated by spaces. $(B_i \in \{0, 1\})$

Output

Print the maximum possible value of P.

Example

standard input (stdin)	standard output (stdout)
2 1 2 0 1	2
4 2 4 3 1 1 1 0 0	2
10 1 3 1 2 3 1 2 3 3 1 1 1 0 0 1 1 1 1 0 1	5

Note

In the first example, if $X = 1$, the states of the box are $[] \to [1] \to [2]$. Therefore, the observed values are $1, 2$.

If $X = 2$, the states of the box are $[] \to [1] \to [1, 2]$. Therefore, the observed values are $1, 1$.

Since the results of the two cases are different, if the information $1 \le X \le 2$ is given, the value of X can be determined exactly.

In the second example, the arrays of observed values for each X are as follows.

X	Array of observed values
1	$2, 4, 4, 4$
2	$2, 2, 3, 3$
3	$2, 2, 3, 3$
4	$2, 2, 3, 1$

Since the array of observed values when $X = 2$ is the same as the array of observed values when $X = 3$, even if the information $1 \le X \le 3$ is given, the value of X cannot be determined exactly.

Therefore, $P = 3$ is impossible, and the maximum possible value is 2.

Problem I. Paper Factory

Time Limit 1 second Memory Limit 1024 MB

Asteromorph, with the mission of accelerating scientific progress through AI, is developing Spacer, a system that can autonomously discover creative and fact-grounded scientific concepts without human intervention. (For details, see arXiv:2508.17661.) The core philosophy of Spacer is *deliberate decontextualization*: breaking information down into atomic units called keywords, then deriving new ideas from connections between them that no one has made yet.

Spacer's inspiration engine, Nuri, collects keywords from numerous papers to build a knowledge graph, and finds keywords within it that may lead to new research ideas.

One day, while developing Nuri, Donghyun was looking at the usage counts of keywords arranged in a row and had the following thought.

> Can I make the usage counts of all these keywords equal?

The keywords are numbered from 1 to N from left to right. Keyword $i (1 \le i \le N)$ was initially used A_i times.

Donghyun can write as many papers as he wants, including zero. However, each paper must discuss two adjacent keywords together, and writing such a paper increases the usage counts of both keywords by 1.

In other words, he may perform the following operation any number of times:

> Choose an integer $i (1 \le i < N)$, and increase both A_i and A_{i+1} by 1.

Can Donghyun make the usage counts of all keywords equal? Help Donghyun.

Input

The first line contains the number of test cases $T (1 \le T \le 100)$.

The first line of each test case contains the number of keywords $N (1 \le N \le 1\,000)$. The second line contains the initial usage counts of the keywords, $A_1, A_2, \cdots, A_N$, separated by spaces.

For each $i (1 \le i \le N)$, $0 \le A_i \le 1\,000$, and the sum of all N over the input is at most 5000. All numbers given in the input are integers.

Output

For each test case, output `Yes` if it is possible to make the usage counts of all keywords equal, and `No` otherwise, one per line.

Example

standard input (stdin)	standard output (stdout)
5 1 7 2 3 3 2 1 2 3 1 0 1 4 1 0 1 0	Yes Yes No Yes No

Problem J. Hard Sorting Problem

Time Limit 1 second Memory Limit 1024 MB

An integer sequence A of length N is given. The sequence is partitioned into K non-overlapping segments, where the length of the i-th segment from the left is l_i. You may perform the following operation any number of times.

- Choose one of the K segments.
- Let m and M be the minimum and maximum values, respectively, among the elements in the chosen segment immediately before the operation. Replace every element in that segment with an arbitrary integer between m and M, inclusive.

The goal is to make the sequence non-decreasing.

After all operations are performed, let the resulting sequence be B. For a value x, let $c_A(x)$ be the number of times x appears in the initial sequence A, and let $c_B(x)$ be the number of times x appears in the resulting sequence B. The cost of B is defined as follows:

$$\sum_x \max(0, c_B(x) - c_A(x)).$$

Determine whether it is possible to make the sequence non-decreasing, and output the minimum cost of such a final sequence if possible.

Input

The first line contains two integers N and K, separated by a space. ($1 \le K \le N \le 200\,000$)

The second line contains N integers $A_1, A_2, \cdots, A_N$, separated by spaces. ($0 \le A_i \le 10^9$)

The third line contains K integers $l_1, l_2, \cdots, l_K$, separated by spaces. ($1 \le l_i \le N; \sum_{i=1}^{K} l_i = N$) These indicate that the sequence is partitioned into K segments of lengths $l_1, l_2, \cdots, l_K$, from left to right.

Output

If it is possible to make the sequence non-decreasing, output the minimum possible cost. Otherwise, output -1.

Example

standard input (stdin)	standard output (stdout)
5 3 3 1 4 2 5 2 1 2	1

In the example above, the sequence A is partitioned into the segments $[1,2]$, $[3,3]$, and $[4,5]$.

It is possible to obtain a non-decreasing sequence $[1,3,4,5,5]$. For the initial sequence $A = [3,1,4,2,5]$ and the final sequence $B = [1,3,4,5,5]$, the cost is calculated as follows.

x	1	2	3	4	5
$c_A(x)$	1	1	1	1	1
$c_B(x)$	1	0	1	1	2

Thus, the cost is $0+0+0+0+1=1$, and it can be proven that no other non-decreasing final sequence has a smaller cost.

standard input (stdin)	standard output (stdout)
4 4 3 1 2 4 1 1 1 1	-1

Since every segment has length of 1, it is impossible to change the sequence. Therefore, the final sequence must be $[3,1,2,4]$, which is not non-decreasing.

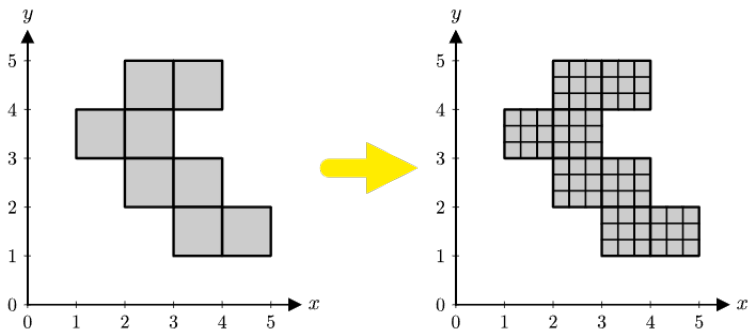
Problem K. Merchant

Time Limit 5 second Memory Limit 1024 MB

There are N villages on a grid. Each village occupies a square region of size 1×1, and the i-th village is located at coordinates (X_i, Y_i). The coordinates follow the usual x-y coordinate system.

There is exactly one path between any two villages. In other words, if we consider the graph whose vertices are the villages and whose edges connect villages adjacent in one of the four cardinal directions, then this graph is a tree.

There are 9 merchants living in each village. Each village is divided into 3×3 regions of equal size, and each region contains exactly one merchant. Every merchant sells a different kind of item. The following is an example illustration.



Example of village and region

You want to buy exactly one of every item while moving only between neighboring regions that share a side. Whenever you visit a region, you must buy one item from the merchant in that region. Since you do not have enough money, you must never be forced to buy two or more items from the same merchant. Also, after buying all items, you must return to the region where you started. However, when you return to the starting region at the end, you are considered not to buy an item again.

Determine whether such a movement is possible, and if it is possible, output one such movement.

Input

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

The first line of each test case contains the number of villages N. $(1 \le N \le 10^5)$

The next N lines contain two integers X_i, Y_i, separated by a space, denoting the coordinates of the i-th village. The coordinates follow the usual x-y coordinate system. $(1 \le X_i, Y_i \le 10^9)$

All village coordinates are distinct, and it is guaranteed that the adjacency relation among the given villages forms a tree.

The sum of N over all test cases is at most 10^5.

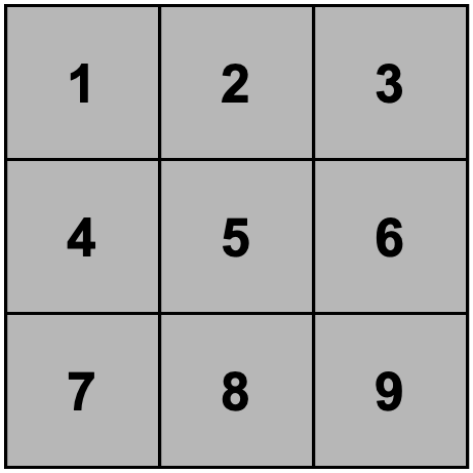
Output

For each test case, if there is no movement satisfying the conditions, output `No` on the first line.

If there exists a movement satisfying the conditions, output `Yes` on the first line.

Then output $9N+1$ lines describing the order of movement. Among them, the i-th line should contain two integers a_i, b_i, separated by a space. This means that the i-th visited region is region b_i of village a_i.

For each village, the region numbers are chosen from the integers 1 through 9 as shown in the figure below.



Region numbers

The regions printed in the first $9N$ lines must all be distinct and must include every region of every village exactly once. In other words, the ordered pairs (a_i, b_i) appearing in the first $9N$ lines must contain each possible ordered pair satisfying $1 \le a_i \le N$ and $1 \le b_i \le 9$ exactly once.

Any two consecutive regions in the visiting order must share an edge. Also, it must hold that $(a_{9N+1}, b_{9N+1}) = (a_1, b_1)$.

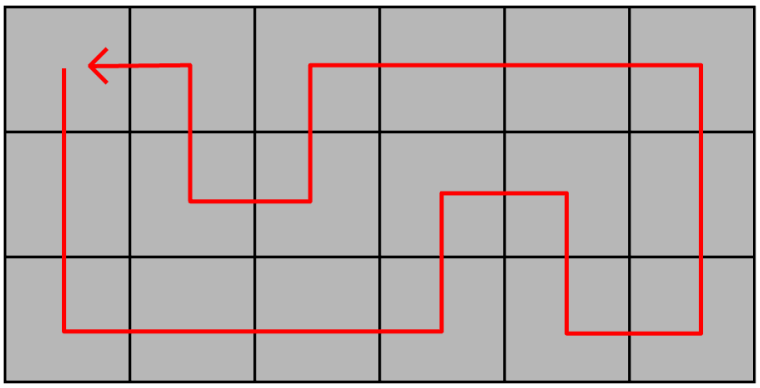
If there are multiple possible movements, you may output any of them.

Example

standard input (stdin)	standard output (stdout)
2 2 1 1 2 1 1 1 1	Yes 1 1 1 4 1 7 1 8 1 9 2 7 2 4 2 5 2 8 2 9 2 6 2 3 2 2 2 1 1 3 1 6 1 5 1 2 1 1 No

Note

The following is an illustration of the first test case.



Order of visits in the first test case

Problem L. Probabilistic Sorting

Time Limit 4 second Memory Limit 1024 MB

You are given a permutation P of length N. Until the array becomes sorted in increasing order, perform the following operation.

- Choose one of the elements currently remaining in the array uniformly at random and remove it.

Find the expected number of elements removed until the remaining array becomes sorted in increasing order.

Input

The first line contains the length N of P. $(1 \le N \le 2000)$

The second line contains the elements $P_1, P_2, \cdots, P_N$ of the permutation, separated by spaces. $(1 \le P_i \le N)$

Output

Print the expected number of removed elements modulo 998244353.

More specifically, it can be shown that the expected value is a rational number. When it is written as an irreducible fraction A/B, it can be shown that there exists integer B^{-1} where $B \times B^{-1} \equiv 1 \pmod{998244353}$. You must output the value of $A \times B^{-1} \pmod{998244353}$.

Example

standard input (stdin)	standard output (stdout)
6 5 2 3 1 4 6	582309210
6 1 2 3 4 5 6	0
6 6 5 4 3 2 1	5