2023 NTU ICPC Team Preliminary

National Taiwan University

2023/07/29

Language	Version	Compile Flags	Extensions
C	gcc 10.2.1	`-O2 -std=c11 -static -lm`	`.c`
C++	g++ 10.2.1	`-O2 -std=c++17 -static`	`.cc, .cpp`
Python3	pypy3 3.7.10	`None`	`.py`

Problem	Problem Name	Time Limit	Memory Limit
A	Area in Convex	1 s	512 MB
B	Bogosort	2 s	512 MB
C	Communication Problem	3 s	512 MB
D	Decision Problem	15 s	512 MB
E	Er Wei Shu Dian	2 s	512 MB
F	Fake Solution	2 s	512 MB
G	Grouping Problem	1 s	512 MB
H	Harmony Coloring	1 s	512 MB
I	IMO Problem	1 s	512 MB
J	Japanese Monsters	2 s	512 MB
K	Known Problem	6 s	512 MB
L	List of Orders	1 s	512 MB
M	More Japanese Monsters	2 s	512 MB

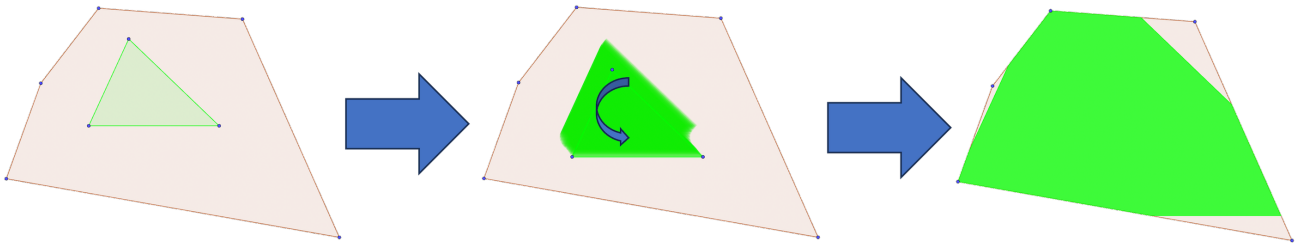
This page is intentionally left blank.

A. Area in Convex

Problem ID: area

Today, oToToT received two convex polygons, denoted as A and B, as his toys. After playing with them for a while, oToToT accidentally dropped the convex polygon B into a puddle of paint! As a result, any area covered by convex polygon B will be colored.

Being an enthusiastic competitive programmer, oToToT immediately came up with a question: If convex polygon B is translated (no rotation allowed) inside convex polygon A, what is the area that is possible to be colored by convex polygon B?



However, oToToT recently encountered a dreadful 3D geometry problem, Codeforces 102452F, hence doesn't want to touch any computational geometry problem recently. Therefore, he entrusts you, who is forced to participate in this contest, to help solve the problem.

Note: We say convex polygon B is inside convex polygon A when all points of convex polygon B lie within convex polygon A.

Input

The input begins with two integers N and M, denoting number of points of convex polygon A and convex polygon B, respectively.

Then, N lines are provided, each containing two integers x_i and y_i, representing the i-th point of convex polygon A.

Following that, M lines are provided, each containing two integers p_i and q_i, representing the i-th points of convex polygon B.

- $3 \leq N, M \leq 10^4$

- $0 \leq |x_i|, |y_i|, |p_i|, |q_i| \leq 10^6$
- The points of the two convex polygons are given in counterclockwise order.
- It is guaranteed that B can be placed inside A.
- Under the condition that B remains inside A, there exist two non-parallel directions in which B can be moved a positive distance.
- Let C be the maximum absolute value among all coordinates. It is guaranteed that C^2 is no more than 10^5 times the area of B.

Output

Output a real number in a single line, representing the area that can be colored within convex polygon A by convex polygon B. Your answer will be considered correct if the relative or absolute error compared to the correct answer is within 10^{-6}.

Sample Input 1	Sample Output 1
4 3 0 0 10 0 10 5 0 5 4 0 6 0 5 4	46.000000000000000

Sample Input 2	Sample Output 2
6 4 -2 2 5 0 10 4 5 12 0 10 -3 5 1 7 3 7 3 9 1 9	94.677042606516295

B. Bogosort

Problem ID: bogosort

This is an interactive problem.

There is an array $[a_1, a_2, \ldots, a_N]$ initialized with a secret permutation of size $N = 100$.

Formally speaking, $\begin{cases} a_i \in \mathbb{N}, & \forall i = 1, 2, \ldots, N \\ 1 \le a_i \le N, & \forall i = 1, 2, \ldots, N \\ a_i \ne a_j, & \forall i \ne j \end{cases}$.

You are asked to sort it in ascending order within 50000 operations. In each operation, you can perform any one of the following actions:

- Pick a consecutive subarray of size at least 5 and randomly shuffle it.
- Query the k-th smallest value of $|a_i - i|$ for $i = 1, 2, \ldots, N$.
- Claim that the array has been sorted.

Can you do it?

Interaction

The interaction begins without reading anything such as N. To perform an operation, please output a line:

- `# l r`: Shuffle the interval $a[l..r]$. l, r should satisfy $1 \le l \le r \le 100$ and $r-l+1 \ge 5$. After calling this operation, the judge shuffles $[a_l, a_{l+1}, \ldots, a_r]$ randomly. That is, each possible $(r - l + 1)!$ permutations is equiprobably produced. After that, you should read a single integer v written on a separate line. $v = 0$ if the shuffle action is successfully executed. Otherwise, you might have made an incorrect shuffle request; in this case, your program should terminate immediately to avoid undefined behavior.
- `? k`: Query the k-th smallest value of $|a_i - i|$ for $i = 1, 2, \ldots, N$. k should satisfy $1 \le k \le N$. After that, you should read an integer b, the query result. If $b = -1$, you might have made an invalid query; in this case, your program should terminate immediately to avoid undefined behavior. Otherwise, b is the value you asked for.
- `!`: Claim that the array has been sorted. Once choosing this action, the judge takes a look on the current array to see if it is really sorted and gives you a verdict accrodingly. Your program should terminate immediately after calling this operation. Please note that this is also counted as an operation so it cannot be made on the 50001-th move.

No matter what operation you have chosen, don't forget to output the end of line and flush the output. Otherwise, I don't know what is going to happen either. To flush the output, use:

- `fflush(stdout)` in C;
- `fflush(stdout)` or `cout.flush()` in C++;
- `stdout.flush()` in Python;

Additional Information

Let me tell you a secret — there are 20 test cases on the judge.

Notes

Oh no! The shuffling operation is too powerful so the sample explanation is misplaced before the Sample section.

In the first sample test case, you asked to shuffle the subarray $[a_{13}, a_{14}, \ldots, a_{37}]$. The judge accepted your request and did the operation successfully, outputting 0 back to you.

Afterward, you wanted to determine if the array had been sorted. So, you output `? 100`. Astonishingly, luck was on your side as the array had indeed become sorted after your previous action, resulting in the 100-th smallest (or largest) value of all $|a_i - i|$ being equal to 0. It's important to note that this outcome happens with an incredibly low probability, just 1 out of 933,262,154,439,441,526,816,992,388,562,667,004,907,159,682,643,816,214,685,929,638,952,175,999,932,299,156,089,414,639,761,565,182,862,536,979,208,272,237,582,511,852,109,168,640,000,000,000,000,000,000,000. Therefore, it's unlikely you'll be so fortunate next time.

After knowing that the array has been sorted, you output `!`, terminating your program.

Sample Input	Sample Output
0 0	# 13 37 ? 100 !

C. Communication Problem

Problem ID: communication

One day, Edisonhello, YP, and Baluteshih were testing a programming contest named PDAO. After reading problem A, Edisonhello conceptualized it and explained what to do to their main coder, YP. However, due to a communication problem, YP misinterpreted the description, resulting in an incorrect understanding. As YP found the misunderstood version of the problem difficult enough (and he did not want to waste any piece of code written), he would like to include it as one of the problems in the 2023 NTU ICPC Team Preliminary.

The intended version of the problem in PDAO was:

Given an integer N, for each $k = 1, 2, \ldots, N$, output

$$\max_{(A,B)\in\wp(N,k)} \frac{f(A,B)}{\prod_{i=1}^{k} p_i}$$

where $\wp(N,k)$ collects all partitions (A,B) of $S = \{1,2,\ldots,N\}$ such that $\begin{cases} A \cap B = \emptyset \\ A \cup B = S \\ |B| = k \end{cases}$, $f(A,B)$ denotes the cardinality of the set $\{(a,b) \mid (b \equiv 0 \mod a) \wedge (a,b) \in A \times B\}$, and p_i denotes the i-th smallest prime number (e.g., $p_1 = 2, p_2 = 3, p_3 = 5$).

YP misunderstood the definition of the denominator in the expression. So the problem you need to solve is instead:

Given an integer N, for each $k = 1, 2, \ldots, N$, output

$$\max_{(A,B)\in\wp(N,k)} \frac{f(A,B)}{\prod_{b\in B} b}$$

where the definition of $\wp$ and f remains the same.

Input

The first and only line of the input contains a single integer N.

- $1 \leq N \leq 10^6$

Output

Output a single line containing N space-separated integer, where the i-th number is the answer for $k = i$. Since YP has no time to write a checker for high-precision floating-point number, please output their congruence modulo 998244353. Formally, let $M = 998244353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \mod M$ for each $k = 1, 2, \ldots, N$. In other words, for each $k = 1, 2, \ldots, N$, output such an integer x that $0 \leq x < M$ and $x \cdot q \equiv p \pmod{M}$.

Sample Input 1	Sample Output 1
3	499122177 332748118 0

D. Decision Problem

Problem ID: decision

Dinner time! oToToT is looking for a good restaurant, but he has been overwhelmed by the amount of choice. There are n ($1 \leq n \leq 3 \cdot 10^5$) restaurants and n types of dishes in the city. The i-th restaurant serves the a_i-th type of dish. Now, oToToT is curious about a question:

"If he walks from the x-th restaurant to the y-th restaurant, is the k-th dish an appropriate choice?"

Since route planning can be exhausting, oToToT has filtered out some redundant streets to ensure that there are exactly one path between any two restaurants; the streets preserved by oToToT form an undirected tree. oToToT answers the question with the following method:

1. Initially, the k-th dish is believed *appropriate*.
2. oToToT moves from the x-th restaurant to the y-th restaurant.
3. Whenever oToToT reaches a restaurant (including the x-th and the y-th restaurant) that serves the k-th dish, he simply changes his opinion.
4. oToToT confirms his answer after he reaches the y-th restaurant. There's an exception: if the k-th dish is not served at any of the restaurants on the path, it cannot be *appropriate*.

oToToT also wants to know the number of *appropriate* dishes, but he is too tired to solve the problem by himself. oToToT will list q ($1 \leq q \leq 3 \cdot 10^5$) pairs of (x_i, y_i). Help him count the number of *appropriate* dishes between the x_i-th restaurant and the y_i-th restaurant. You must figure out the answer before the next query arrives.

Input

The first line of the input contains an integer n ($1 \leq n \leq 3 \cdot 10^5$) —the number of restaurants.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq n$) —the dish served at each restaurant.

The following $n-1$ lines of the input contains 2 integers u_i and v_i ($1 \leq u_i, v_i \leq n$) —indicating a street between the u_i-th restaurant and the v_i-th restaurant. It is guaranteed that the given streets form a tree.

The next line of the input contains an integer q ($1 \leq q \leq 3 \cdot 10^5$) —the number of queries.

The following q lines of the input contains 2 integers x'_i and y'_i ($1 \leq x'_i, y'_i \leq n$).

If the last answer is t, then:

$$x_i = ((x'_i + t) \mod n) + 1$$
$$y_i = ((y'_i + t) \mod n) + 1$$

where mod is the modulo operator (e.g. $11 \mod 4 = 3$). Especially, t is defined 0 before the first query.

Output

For each query, output a integer representing the answer.

Hint

Decoded queries of **Sample Input 2**

```
15
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
```

Sample Input 1

```
6
1 1 5 5 3 3
1 2
6 2
5 6
4 6
3 5
4
6 1
3 1
3 2
2 4
```

Sample Output 1

```
1
0
2
2
```

Sample Input 2

```
5
1 1 2 2 2
1 2
2 3
3 4
4 5
15
5 5
5 1
4 1
4 2
3 2
5 5
1 2
1 3
5 3
2 2
2 3
1 3
3 3
3 4
3 3
```

Sample Output 2

```
0
1
1
2
1
0
0
1
0
0
1
0
0
1
0
```

This page is intentionally left blank.

E. Er Wei Shu Dian

Problem ID: erwei

Urgent!!

The two great legends, Tseng and Chou, are debating the pronunciation of "Er Wei Shu Dian." You need to stop their quarrel in time; otherwise, the world would be diminished due to the disorder of the two great powers. In order to stand out from them, you must solve this variant of the classic "Er Wei Shu Dian" problem first.

Now, you are given N points, the i-th point can be described by two integers (x_i, y_i), which means that the i-th point is placed at (x_i, y_i) on a 2D plane. For the i-th point, you need to find out how many points are strictly inside the upper-left corner and the upper-right corner. Since this is an urgent task, to save your time, you only need to output the sum of these values.

Input

The first line of the input contains a single integer N. Then, N lines are provided, each containing two integers x_i and y_i, representing the i-th point.

- $1 \leq N \leq 3 \times 10^5$
- $-10^9 \leq x_i, y_i \leq 10^9$

Output

Output a single line representing the answer.

Sample Input 1	Sample Output 1
6 1 1 1 1 4 4 5 5 1 1 4 4	11

Sample Input 2	Sample Output 2
7 8 9 -1 -2 -3 -4 2 5 0 0 3 5 8 10	19

F. Fake Solution

Problem ID: fakesolution

This is a problem from a story about misreading the problem statement and coming up with a (fake) solution.

Consider a sequence of N non-negative integers $a_1, \ldots, a_N$. A single *operation* consists of the following 3 steps:

1. Choose any index $1 \leq i \leq N$.
2. Choose any non-negative integer j such that $2^j \leq a_i$.
3. Change a_i to $a_i \mid (a_i - 2^j)$. Here, $-$ denotes natural subtraction, and $\mid$ denotes bitwise-or.

Your goal is to make the bitwise-or of the whole sequence as big as possible by performing zero or more operations. You also need to solve the minimum number of operations to obtain the maximum bitwise-or.

Input

There are multiple testcases.

The first line is an integer T denote the number of testcases. Each testcase starts with a positive integer N, and then N space-seperated non-negative integers on the next line is a_i.

- $1 \leq T \leq 10^6$
- $1 \leq N \leq 10^6$
- The sum of N in each testcases is not greater than 10^6.
- $0 \leq a_i < 2^{30}, \forall 1 \leq i \leq N$

Output

For each testcases, output a single line consists of two space-seperated integers, denoting the maximum bitwise-or and the minimum operation needed to obtain that maximum.

Sample Input 1	Sample Output 1
1 3 1 5 9	15 1

Sample Input 2	Sample Output 2
1 1 21	31 2

G. Grouping Problem

Problem ID: grouping

In the NTU field, there are N people standing on a land shaped like a number line, waiting to participate in the unique NTU game. Each person can be considered to be standing at coordinate x_i on the number line.

As the game master of the event, you need to group these people. For each group, let l be the minimum coordinate, and r be the maximum coordinate among the people in the group. Then, you have to pay $a + b \times (r - l)$ dollars to organize a game for that group.

You quickly realize how to minimize the cost of grouping everyone efficiently. However, you soon discover that the cost is still very high. To address this, you wish to dismiss some people to reduce the overall cost of the game.

After conducting some investigations, you find out that asking the i-th person to leave requires spending c_i dollars. Moreover, there are M hidden friend relations among these people. Each friend relation is represented as a triple (u_i, v_i, w_i), which means there is a friendship between u_i and v_i. If **exactly one** person among u and v is asked to leave, you will need to spend an additional w_i dollars to compensate for their friendship!

Despite the complexity of these relationships, you still aim to minimize the total cost of the game. Please write a program that takes the given input and outputs the minimum possible cost.

Input

The input begins with four integers N, M, a, and b, denoting the number of people, the number of friend relations, and the parameters a, b, respectively.

The second line contains N integers $x_1, x_2, \ldots, x_N$, representing that the coordinate of the i-th person is x_i.

The third line contains N integers $c_1, c_2, \ldots, c_N$, representing that the cost for asking the i-th person to leave.

Following that, M lines are provided, each containing three integers u_i, v_i, and w_i, representing that the i-th friend relation connecting person u_i and person v_i with cost w_i.

- $1 \leq N \leq 200$

- $0 \leq M \leq \min(200, \frac{N \times (N-1)}{2})$
- $1 \leq a, b, c_i, w_i \leq 10^9$
- $1 \leq x_1 < x_2 < \cdots < x_N \leq 10^6$
- $1 \leq u_i, v_i \leq N$, $u_i \neq v_i$
- No two friend relations connect the same pair of two people.

Output

Output a single line with the minimum possible cost.

Sample Input 1

```
5 3 4 2
1 5 6 9 10
2 10 1 10 10
1 2 1
3 4 8
4 5 9
```

Sample Output 1

```
15
```

Sample Input 2

```
6 9 5 3
1 4 6 7 11 12
4 3 9 5 7 6
2 6 3
5 2 7
3 2 2
4 5 6
1 5 6
4 6 4
4 3 9
1 6 1
3 1 6
```

Sample Output 2

```
26
```

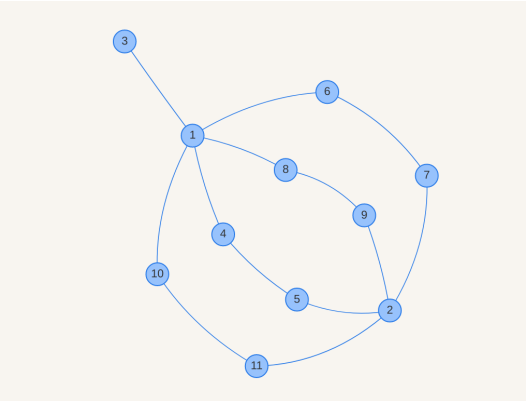
H. Harmony Coloring

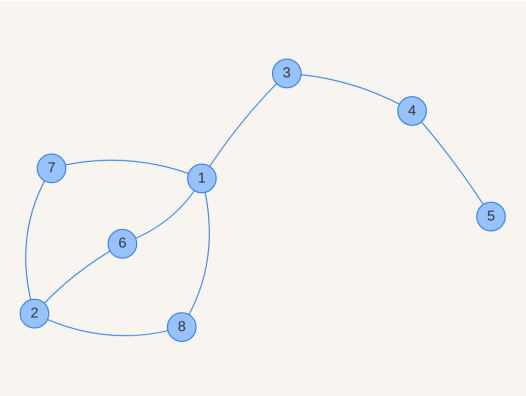
Problem ID: harmony

A *Suica Graph* is defined as an undirected graph that can be constructed using the following procedures.

1. Start with an empty graph (with no vertices or edges).
2. Add a head vertex H.
3. Add a tail vertex T.
4. Add N stripe chains $C_1, C_2, \ldots, C_N$. Each chain is a set of degree-2 vertices. When adding chain C_i, we first add l_i vertices, say $v_1, v_2, \ldots, v_{l_i}$. Then add edges between adjacent vertices; i.e., for all $1 \leq j < l_i$, add edges between v_j and v_{j+1}. Finally, add two edges $\{H, v_1\}$ and $\{v_{l_i}, T\}$.
5. Add the stem chain. Suppose that the length of the stem chain is S, then add S vertices denoted by $u_1, u_2, \ldots, u_S$. Secondly, add edges between adjacent vertices; i.e., for all $1 \leq j < S$, add edges between u_j and u_{j+1}. Finally, add edge $\{H, u_1\}$.

Here are some examples of *Suica Graphs*:





You are given a *Suica Graph*. For each vertex, you can color it red or blue. Please count how many different *Harmony* colorings there are. Since the answer could be large, you only need to output the remainder of the answer modulo by 998244353.

A coloring is called *Harmony* if and only if:

1. There is at least one red vertex and at least one blue vertex.

2. For each pair of red vertices (x, y), x can reach y through some edges without touching blue vertices.
3. For each pair of blue vertices (x, y), x can reach y through some edges without touching red vertices.

In other words, a coloring is *Harmony* if and only if the induced subgraph of red vertices is connected, the induced subgraph of blue vertices is connected, and both of them are not empty graphs (i.e., having at least one vertex).

Two coloring ways are considered different if and only if there exists a vertex colored differently in the two coloring ways.

Input

The first line of the input is the parameter N, S, seperated by space. N, S are the number of stripe chains and the length of stem chain, respectively. The second line consists of N integers $l_1, l_2, \ldots, l_N$.

- $1 \leq N \leq 10^5$
- $1 \leq S \leq 10^5$
- $1 \leq l_i \leq 10^5$, $\forall 1 \leq i \leq N$.

Output

Output a single integer between 0 and 998244352 represent the remainder of the number of different coloring ways.

Sample Input 1	Sample Output 1
4 1 2 2 2 2	188

Sample Input 2	Sample Output 2
3 3 1 1 1	28

Sample Input 3	Sample Output 3
9 159 114 514 191 918 103 314 159 265 358	37949969

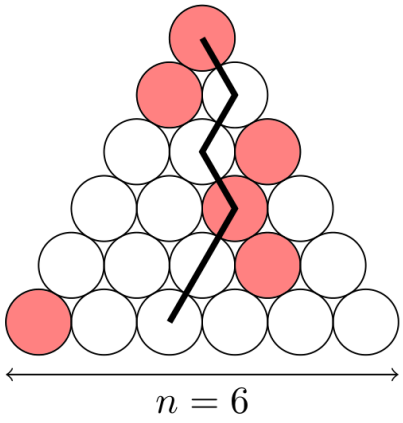
This page is intentionally left blank.

I. IMO Problem

Problem ID: imoproblem

The section below is from IMO 2023 problem 5.

Let n be a positive integer. A *Japanese triangle* consists of $1 + 2 + \cdots + n$ circles arranged in an equilateral triangular shape such that for each $i = 1, 2, \ldots, n$, the i-th row contains exactly i circles, exactly one of which is coloured red. A *ninja path* in a *Japanese triangle* is a sequence of n circles obtained by starting in the top row, then repeatedly going from a circle to one of the two circles immediately below it and finishing in the bottom row. Here is an example of a Japanese triangle with $n = 6$, along with a ninja path in that triangle containing two red circles.



Note that this is not the ninja path containing maximum red circles.

Now, you are given a configuration of a *Japanese triangle*, please find the *ninja path* containing maximum red circles.

Input

The first line of the input contains an integer n. The second line of the input contains n integers a_i, seperated by space. $a_i = x$ means the x-th circle of the i-th row is red.

- $1 \leq n \leq 10^6$
- $1 \leq a_i \leq i \ \forall 1 \leq i \leq n$.

Output

Output a single line, the maximum number.

Sample Input 1	Sample Output 1
6 1 1 3 3 4 1	4

Sample Input 2	Sample Output 2
6 1 1 1 3 5 6	3

J. Japanese Monsters

Problem ID: japanese

The only differences between this problem and ***More Japanese Monsters*** *are Input and Output.*

The great player Rgnerd is playing a well-known Japanese game called Pekomon. In this game, he will encounter many different Peko Monsters. He can either capture the Peko Monster he meets or decide to kill it directly. Each Peko Monster has different skills, and some of them are considered legendary due to their high-level skills.

One day, Rgnerd bumped into a legendary Peko Monster, Splash Splash Rage Shark. Even though Splash Splash Rage Shark is a legendary Peko Monster, Rgnerd still thinks it is pretty easy to defeat it.

One of the skills of Splash Splash Rage Shark is to metamorphose. After the metamorphosis, Splash Splash Rage Shark will hide in a string S. Rgnerd knows that if Splash Splash Rage Shark does exist in that string, it must be in one of the suffixes of the string. Similar to the abbreviation of Splash Splash Rage Shark's form, Rgnerd also identifies that it must be hidden in the form of $AABA$, where A and B are some non-empty strings.

To demonstrate how easily Rgnerd can kill Splash Splash Rage Shark, he will showcase his quantum sword skill. He will kill all possible Splash Splash Rage Sharks at the same time. You want to know how many possible kinds of Splash Splash Rage Sharks are killed by Rgnerd at the same time.

Formally, given a string S, for each suffix $\text{Suf}_i = S_{i...|S|}$, you need to find out how many possible ways Suf_i can be decomposed into the form of $AABA$. We will denote this number as A_i.

Input

The input only contains a single line of string S. Besides, S only consists of lowercase English letters.

- $1 \leq |S| \leq 2 \times 10^5$

Output

The i-th line of output should be A_i.

Sample Input 1	Sample Output 1
aaaaa	1 1 0 0 0

Sample Input 2	Sample Output 2
easyeasytooeasy	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

K. Known Problem

Problem ID: known

Here comes a well-known problem,

Given an integer sequence $a_1, a_2, \ldots, a_N$. The task is to respond to Q queries that seek the smallest value among the absolute differences of all pairs of elements within a specified range $[l_i, r_i]$.

This problem has appeared in various online judges, including TIOJ 1905, Codeforces 765F, Codeforces 1793F, and even CS Academy Closest Numbers. Why do we still need to solve this problem?

Let us present an advanced version of it. In this version, we provide Q queries for a given integer sequence $a_1, a_2, \ldots, a_N$, where each query asks to find the **second smallest** value among the absolute differences of all pairs of elements within a specified range $[l_i, r_i]$.

In this context, the second smallest value within a multiset S is determined as the minimum value v, for which there are at least two elements in S that are not greater than v.

Input

The input begins with two integers N and Q, denoting the length of the sequence and the number of operations, respectively.

The second line contains N integers $a_1, a_2, \ldots, a_N$, representing the elements of the initial sequence.

Following that, Q lines are provided, each containing two integers l_i and r_i, representing the i-th query.

- $3 \leq N \leq 10^5$
- $1 \leq Q \leq 3 \times 10^6$
- $0 \leq a_i \leq 10^9$
- $1 \leq l_i, r_i \leq N$
- $r_i - l_i > 1$

As you can see, the time limit is tight since Q can be up to 3×10^6. Therefore, it is advisable to avoid extensive use of slow data structures like `std::map`, `std::unordered_map`, etc.

Output

For each query, output the second smallest difference within the range $[l_i, r_i]$.

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

L. List of Orders

Problem ID: list

NTU Factory recently received N **orders**, each containing the following information:

1. The final item to be produced.
2. The earliest start time. For example, the required materials for the final item may arrive at a certain time.
3. The expected **deadline** d_i.

To complete each final item, the factory needs to go through multiple works. These works cannot be parallelized; that is, the $(i+1)$-th work can only start after the i-th work is done. More precisely, each order is divided into k_i **works** $w_{i,1}, w_{i,2}, \ldots, w_{i,k_i}$, representing the k_i works within the order. For each work a, there is an **item** z_a required to be produced. Once all these k_i works are done, the final item is then magically produced.

To maintain the responsiblility of each factory labor, there are certain constraints when multiple works aim to produce the **same item**:

1. These works cannot be performed simultaneously; one must be completed before the other starts.
2. To determine which work should be done earlier, the labors should follow the following rule. Recall that each work belongs to an order, right? A work belonging to an order with earlier deadline should be done first. That is, if we focus on all the works producing a specific item z and sort these works in chronological order of their completion time, it should have also been sorted by the deadlines of their belonging order.

To complete the numerous works, the factory needs M **resources**. For each resource, the factory must assign a production sequence, specifying the order of works in which each resource should process. These assigned production sequences form a **production plan**. Once the production plan is determined, each resource starts processing works according to its assigned sequence. However, if a resource finds that the next work it needs to process depends on other works that have not been completed, the resource will pause until all dependencies are resolved before continuing.

Currently, NTU Factory's production plan is manually determined, but the existing method, though free from sequence conflicts, is highly inefficient, leading to many delayed orders. To

improve production efficiency, the NTU Factory owner asks the engineers to use programming techniques to optimize the production plan.

The engineer has come up with the following method: Define a state as the production sequence for each resource. The initial state can be obtained from the current production plan. After obtaining the initial state, the following improvements are repeatedly carried out:

Step 1. Select a **continuous segment of works** on one resource.

Step 2. Attempt to insert this continuous segment of works into the production sequence of a certain resource (which can be the same resource).

Step 3. Temporarily store the new state as a test production plan and evaluate whether to accept this new state based on production efficiency, the number of delayed orders, and other evaluation metrics. The new state will not be adopted if it performs poorly.

The current issue faced by the engineer is that Step 2 may lead to **conflicts** in the work sequence. For example, the production plan may cause two resources to wait for each other's works to be completed, resulting in both resources pausing and not processing any works.

Therefore, the engineer entrusts you with the task of implementing a procedure that can input the current production plan, select a continuous segment of works in Step 1, and then repeatedly answer whether conflicts will occur in the work sequence after attempting to insert this segment of works into a specified new position in Step 2.

Input

The input is divided into three parts: information of the orders, production plan, and queries, each separated by an empty line for easy identification.

Information of the Orders

The first line contains two positive integers N and W, representing the number of orders and the number of works.

The following N lines represent the N orders. The i-th line begins with two positive integers d_i and k_i, indicating the deadline and the number of works for the i-th order. Following that are k_i positive integers, where the j-th integer $w_{i,j}$ represents the index of the j-th work in the order, representing the execution order of the works required for this order.

Next is one line with W positive integers $z_1, z_2, \ldots, z_W$, where z_i represents the item number that the i-th work is intended to produce.

- $1 \leq N, W \leq 10^5$
- $1 \leq d_i \leq 10^9$
- All d_i are distinct.
- $\sum_{i=1}^{N} k_i = W$
- $1 \leq w_{i,j} \leq W$
- Each work will appear once in exactly one order.
- Works within the same order will not produce the same item.
- $1 \leq z_i \leq 10^9$

Production Plan

The first line contains a positive integer M, representing the number of resources.

The following M lines represent the initial production plan. The i-th line begins with a positive integer t_i, indicating the number of works assigned to resource i. Following that are t_i positive integers, where the j-th integer $x_{i,j}$ represents the index of the j-th work in the order of execution for resource i.

- $1 \leq M \leq 10^5$
- $\sum_{i=1}^{M} t_i = W$
- $1 \leq x_{i,j} \leq W$
- Each work will appear once in exactly one resource.
- It is guaranteed that the initial production plan does not have any conflicts.

Queries

The first line contains four positive integers Q, u, l, r, representing the number of queries, the resource number specified in Step 1, and the continuous segment of works selected in Step 1, which is the interval $x_{u,l}, x_{u,l+1}, \ldots, x_{u,r}$.

The following Q lines each contain two integers v_i, p_i, representing the selection of inserting this segment of works into resource v_i after the p_i-th work in the production sequence. Specifically, if $p_i = 0$, it means inserting this segment of works at the beginning of the production sequence of resource v_i, that is, before the first work $x_{v_i,1}$.

- $1 \leq Q \leq 10^5$
- $1 \leq u \leq M$
- $1 \leq l \leq r \leq t_u$
- $1 \leq v_i \leq M$
- $0 \leq p_i \leq t_{v_i}$
- In particular, if $u = v_i$, then $p_i < l$ or $p_i > r$.

Output

For each query, output one line with `Yes` if inserting this way will not cause conflicts in the work sequence, or `No` otherwise.

Sample Input 1	**Sample Output 1**
3 10 10 3 1 2 3 20 4 4 5 6 7 30 3 8 9 10 42 49 114 514 42 49 114 49 42 2023 4 2 1 8 3 2 5 6 2 3 4 3 7 9 10 5 2 2 3 1 1 2 0 3 2 3 1 4 3	Yes No Yes No No

M. More Japanese Monsters

Problem ID: more

The only differences between this problem and ***Japanese Monsters*** *are Input and Output.*

The great player Rgnerd is playing a well-known Japanese game called Pekomon. In this game, he will encounter many different Peko Monsters. He can either capture the Peko Monster he meets or decide to kill it directly. Each Peko Monster has different skills, and some of them are considered legendary due to their high-level skills.

One day, Rgnerd bumped into a legendary Peko Monster, Splash Splash Rage Shark. Even though Splash Splash Rage Shark is a legendary Peko Monster, Rgnerd still thinks it is pretty easy to defeat it.

One of the skills of Splash Splash Rage Shark is to metamorphose. After the metamorphosis, Splash Splash Rage Shark will hide in a string S. Rgnerd knows that if Splash Splash Rage Shark does exist in that string, it must be in one of the suffixes of the string. Similar to the abbreviation of Splash Splash Rage Shark's form, Rgnerd also identifies that it must be hidden in the form of $AABA$, where A and B are some non-empty strings.

To demonstrate how easily Rgnerd can kill Splash Splash Rage Shark, he will showcase his quantum sword skill. He will kill all possible Splash Splash Rage Sharks at the same time. You want to know how many possible kinds of Splash Splash Rage Sharks are killed by Rgnerd at the same time.

Formally, given a string S, for each suffix $\text{Suf}_i = S_{i...|S|}$, you need to find out how many possible ways Suf_i can be decomposed into the form of $AABA$. We will denote this number as A_i.

Input

The input only contains a single line of string S. Besides, S only consists of lowercase English letters.

- $1 \leq |S| \leq 4 \times 10^6$

Output

Since the output might be too large, you only need to output a single integer

$$(\sum_{i=1}^{|S|} 114514^{i-1} \times A_i) \mod 998244353$$

Sample Input 1	Sample Output 1
aaaaa	114515

Sample Input 2	Sample Output 2
easyeasytooeasy	1