D1. Diadrash (Easy Version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes

This is the Easy version of the problem. The difference between the versions is that in this version, you can make at most $\max(300, \lceil \frac{n}{2} \rceil + 2)$ queries. You can hack only if you solved all versions of this problem.

This problem is interactive.

There is a permutation* p of the integers from 0 to $n - 1$ hidden from you. Additionally, you are given q ranges $[l_1, r_1], [l_2, r_2], \ldots, [l_q, r_q]$ where $1 \le l_i \le r_i \le n$.

You need to discover the maximum MEX among the values of p in the q ranges that are given to you in the input. Formally, you must discover the value of $\max_{i=1}^{q} \mathrm{MEX}([p_{l_i}, p_{l_i+1}, \ldots, p_{r_i}])^{\dagger}$. To do this, you may make at most $\max(300, \lceil \frac{n}{2} \rceil + 2)$ queries of the following form:

- Choose **any** two integers $1 \le l \le r \le n$ and you will receive the value $\mathrm{MEX}([p_l, p_{l+1}, \ldots, p_r])$.

*A permutation of the integers from 0 to $n - 1$ is a sequence of n elements where every integer from 0 to $n - 1$ appears exactly once. For example, the sequence $[0, 3, 1, 2]$ is a permutation, but the sequence $[0, 0, 2, 1]$ is not.

Codeforces Round 1063 (Div. 2)
比赛进行中
01:24:56
Contestant

→ 提交?

语言: GNU G++20 13.2 (64 bit, w

选择文件: 选择文件 未选择文件

Be careful: there is 50 points penalty for submission which fails the pretests or resubmission (except failure on the first test, denial of judgement or similar verdicts). "Passed pretests" submission verdict doesn't guarantee that the solution is absolutely correct and it will pass system tests.

提交

→ 评分表

	Score
Problem A	438

† The MEX of a sequence is defined as the smallest non-negative integer that does not appear in that sequence. For example, $\text{MEX}([0,0,1,3]) = 2$ and $\text{MEX}([1,2,2]) = 0$.

有道 翻译

这是这个问题的简单版本。两个版本的不同之处在于，在这个版本中，您最多可以进行 $\max(300, \lceil \frac{n}{2} \rceil + 2)$ 查询。只有解决了这个问题的所有版本，你才能破解

这个问题是互动的。

有一个从 0 到 $n-1$ 的整数的排列 * p 对你隐藏。此外，您将获得 q 范围 $[l_1, r_1], [l_2, r_2], \ldots, [l_q, r_q]$ ，其中 $1 \le l_i \le r_i \le n$ 。

您需要在输入中提供给您的 q 范围内的 p 值中找出最大的 MEX 值。正式地，您必须发现 $\max_{i=1}^{q} \text{MEX}([p_{l_i}, p_{l_i+1}, \ldots, p_{r_i}])$ † 的值。要做到这一点，你最多可以进行如下形式的 $\max(300, \lceil \frac{n}{2} \rceil + 2)$ 查询：

—选择**任意**两个整数 $1 \le l \le r \le n$ ，您将收到值 $\text{MEX}([p_l, p_{l+1}, \ldots, p_r])$ 。

从 0 到 $n-1$ 的整数的排列是一个由 n 元素组成的序列，其中从 0 到 $n-1$ 的每个整数恰好出现一次。例如，序列 $[0,3,1,2]$ 是一个排列，但序列 $[0,0,2,1]$ 不是。

序列的 MEX 被定义为不出现在该序列中的最小非负整数。例如， $\text{MEX}([0,0,1,3]) = 2$ 和 $\text{MEX}([1,2,2]) = 0$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains exactly two integers n and q ($4 \le n \le 10^4$, $1 \le q \le 3 \cdot 10^5$) — the size of the permutation and the number of ranges, respectively.

The i-th of the next q lines contains two integers l_i, r_i ($1 \le l_i \le r_i \le n$).

It is guaranteed that the sum of n and q do not exceed 10^4 and $3 \cdot 10^5$ respectively over all test cases.

Additional Constraint: it is guaranteed that no range is repeated in the same test case.

有道 翻译

Problem B	876
Problem C	1095
Problem D1	1314
Problem D2	1095
Problem E	2409
Successful hack	100
Unsuccessful hack	-50
Unsuccessful submission	-50
Resubmission	-50

* If you solve problem on 00:31 from the first attempt

输入** **

每个测试包含多个测试用例。第一行包含测试用例的数量 t （ $1 \le t \le 100$ ）。下面是测试用例的描述。

每个测试用例的第一行包含两个整数 n 和 q （ $4 \le n \le 10^4$ ， $1 \le q \le 3 \cdot 10^5$ ）——分别是排列的大小和范围的数量。

接下来的 q 行的第 i 包含两个整数 l_i, r_i （ $1 \le l_i \le r_i \le n$ ）。

保证在所有测试用例中 n 和 q 的和分别不超过 10^4 和 $3 \cdot 10^5$ 。

附加约束：保证在相同的测试用例中没有重复的范围。

Interaction

To ask a query, output a line in the following format (without the quotes):

- "? l r" ($1 \le l \le r \le n$)

The jury will return a single integer, the value of $\text{MEX}([p_l, p_{l+1}, \ldots, p_r])$.

When you have found the answer, output a single line in the following format:

- "! x" ($0 \le x \le n$).

After that, proceed to process the next test case or terminate the program if it was the last test case. Printing the answer does not count as a query.

The interactor is **not** adaptive, meaning that the values of the permutation are known before the participant asks the queries.

If your program makes more than $\max(300, \lceil \frac{n}{2} \rceil + 2)$ queries, your program should immediately terminate to receive the verdict `Wrong Answer`. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

After printing a query do not forget to output the end of line and flush the output. Otherwise, you may get the `Idleness Limit Exceeded` verdict. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Hacks

For hacks, use the following format.

The first line of the input should contain a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case should contain exactly two integers n and q ($4 \le n \le 10^4$, $1 \le q \le 3 \cdot 10^5$) — the size of the permutation and the number of ranges you are given, respectively.

The second line should contain n integers $p_1, p_2, \ldots, p_n$ — where p_i is the i-th element of the permutation. It should hold that p is a permutation containing integers 0 to $n - 1$.

The i-th of the next q lines should contain two integers l_i, r_i ($1 \le l_i \le r_i \le n$).

It should hold that the sum of n and q do not exceed 10^4 and $3 \cdot 10^5$ respectively over all test cases.

Additional Constraint: no range should be repeated in the same test case.

Example

input Copy

```
3
4 3
1 2
2 4
1 3

2

0

1

4

6 6
```

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

6

1

2

4 4
1 1
2 2
3 3
4 4
```

output Copy

```
? 1 3

? 4 4

? 1 1

? 1 4

! 2




? 1 6
```

```
? 3 3

? 2 4

! 2



! 1
```

Note

In the first example, the hidden permutation is $p = [0, 3, 1, 2]$ and the ranges are $[1, 2], [2, 4], [1, 3]$. The third range is optimal, as $\text{MEX}([p_1, p_2, p_3]) = \text{MEX}([0, 3, 1]) = 2$, which is maximum.

In our first query, we ask about $l = 1, r = 3$ and the judge gives us the value $\text{MEX}([p_1, p_2, p_3]) = 2$. In the second query, we ask about $l = 4, r = 4$ and the judge gives us the value $\text{MEX}([p_4]) = 0$. Likewise, $\text{MEX}([p_1]) = 1$ and $\text{MEX}([p_1, p_2, p_3, p_4]) = 4$.

Somehow, we figure out that the answer we are looking for is 2.

In the second example, $p = [3, 5, 0, 1, 4, 2]$.

In the third example, $p = [0, 1, 2, 3]$.

Note that this is just an explanation of the way the interaction works and does not show any strategy to solve the problem.

有道 翻译

注意

在第一个例子中，隐藏的排列是 $p = [0, 3, 1, 2]$ ，范围是 $[1, 2], [2, 4], [1, 3]$ 。第三个范围是最优的，如 $\text{MEX}([p_1, p_2, p_3]) = \text{MEX}([0, 3, 1]) = 2$ ，它是最大值。

在我们的第一个查询中，我们询问 $l = 1, r = 3$ ，裁判给了我们值 $\text{MEX}([p_1, p_2, p_3]) = 2$ 。在第二个查询中，我们询问 $l = 4, r = 4$ ，裁判给了我们值 $\text{MEX}([p_4]) = 0$ 。同样， $\text{MEX}([p_1]) = 1$ 和 $\text{MEX}([p_1, p_2, p_3, p_4]) = 4$ 。

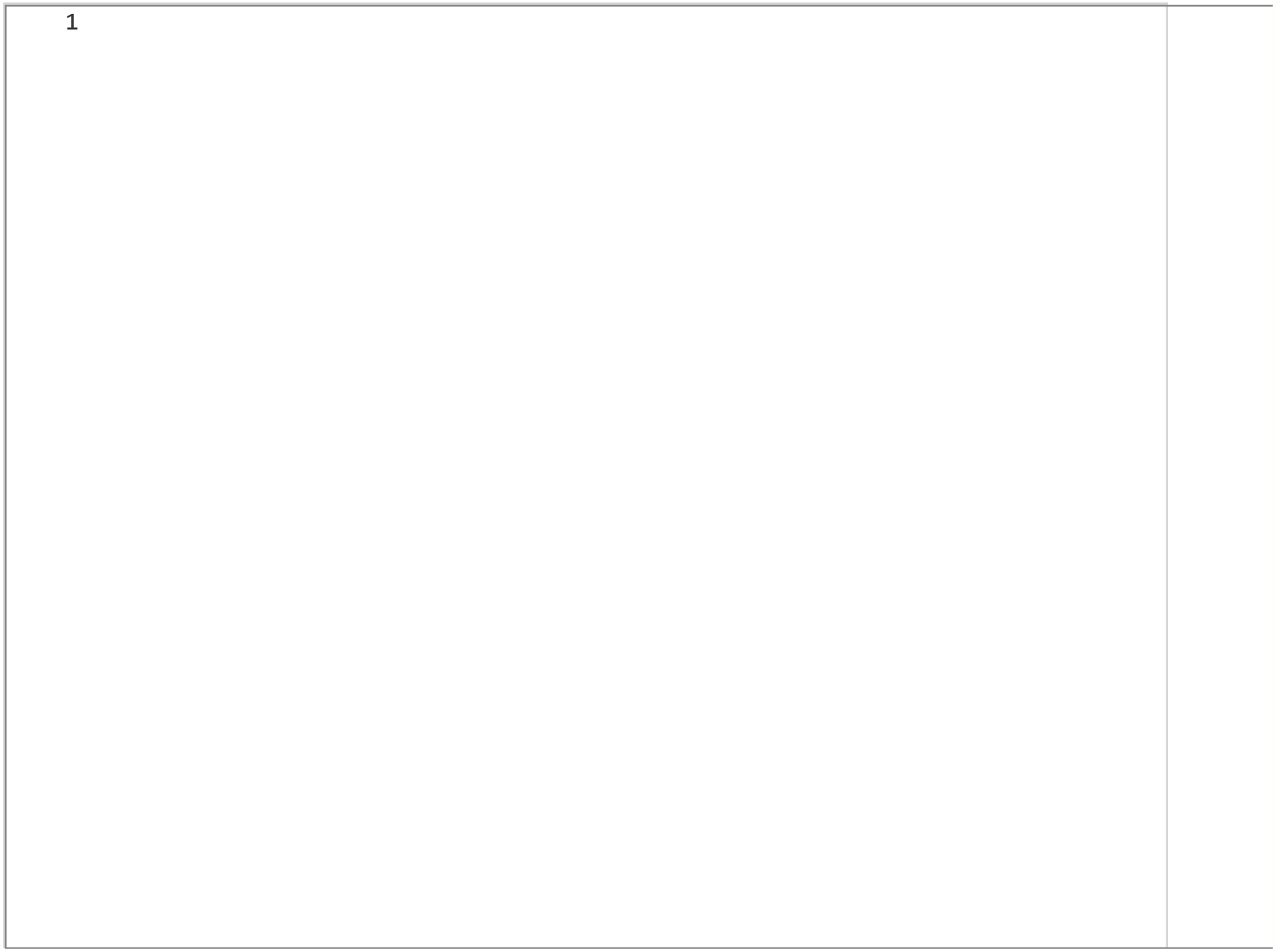
不知怎么的，我们算出我们要找的答案是 2 。

在第二个示例中， $p = [3, 5, 0, 1, 4, 2]$ 。

在第三个示例中， $p = [0, 1, 2, 3]$ 。

**请注意，这只是对交互工作方式的解释，并没有显示任何解决问题的策略

GNU G++20 13.2 (64 bit, winlibs)



▶ 自定义测试数据(自动保存)