Task: BOI
BOI acronym

BOI TORUŃ 2025

BOI 2025, Day 1. Available memory: 256 MB. *2025.04.26*

As you certainly know, *BOI* is an acronym for the name of the *Baltic Olympiad in Informatics*.

The organisers find the acronym *BOI* too easy to pronounce (it forms a single syllable in the English language, after all). Therefore they came up with a new acronym. In order to easily distinguish it from the other regional Olympiads (like *CEOI*), the new acronym still consists only of characters "`B`", "`O`" and "`I`". Additionally, "`B`" is strictly the most common character in the acronym. That is, there are strictly more occurrences of "`B`" than "`O`", and there are also strictly more occurrences of "`B`" than "`I`".

For example, acronyms "`OBOIIBB`" and "`B`" are valid, but "`IBIIBB`", "`BOI`", "`O`" and "`BCB`" are not.

To make things more exciting, instead of publishing it in full they have only provided some hints. Namely, for each consecutive substring of the new acronym, they gave you the number of occurrences of the most common character in this substring. Note that this character is not necessarily "`B`", and also the most common character is not necessarily unique. Surprisingly, it can be proven that this information is actually enough to recover all the occurrences of "`B`". Can you find them?

Input

The first line contains an integer n ($1 \leq n \leq 2000$), denoting the length of the new acronym.

The following n lines describe the hints. The i-th line contains $n - i + 1$ integers $M_{i,i}, M_{i,i+1}, \ldots, M_{i,n}$ ($1 \leq M_{\ell,r} \leq n$), where $M_{\ell,r}$ denotes the number of occurrences of the most common character in the substring that starts at the ℓ-th position and ends at the r-th position of the acronym. The positions are numbered from 1 to n.

You can assume that there exists at least one valid acronym that is consistent with the given hints.

Output

Output one line with the positions of all occurrences of "`B`", **in the increasing order**, separated by single spaces. Each position must be an integer in the range from 1 to n.

Example

For the input data:

```
6
1 1 2 3 3 3
1 1 2 2 2
1 2 2 2
1 1 2
1 2
1
```

the correct result is:

```
1 3 4
```

Scoring

Subtask	Constraints	Points
1	$n \leq 10$	11
2	The sought acronym contains only characters "`B`" and "`O`".	12
3	The sought acronym has no two consecutive equal characters.	10
4	$n \leq 40$	11
5	$n \leq 500$	19
6	No additional constraints.	37

Task: TOU
Tour

BOI 2025, Day 1. Available memory: 1024 MB. *2025.04.26*

There are many tourist attractions in Toruń. Our tour guides prepared a list of m one-way walks connecting n meeting points in the city center. The walks are numbered from 1 to m and similarly the meeting points are numbered from 1 to n. Each walk leads from one meeting point to another and allows the participants to see a single attraction on the way. It might be possible to see the same attraction on different walks and there might be mutliple walks between the same pair of meeting points. We would like to organise an *interesting tour* on our day off.

A *tour* is a sequence of walks, such that every walk starts at the meeting point where the previous one ends. Furthermore, the last walk ends at the meeting point where the very first walk begins.

We call such a tour *interesting* if it doesn't contain the same attraction twice in a row. In other words, every two consecutive walks from the tour allow us to see different attractions, and additionally the very first and very last walks from the tour allow us to see different attractions as well. Note that we do not mind if some non-consecutive walks allow us to see the same attraction. In particular, the same walk might be used multiple times on the tour (but not twice in a row).

Your task is to check if it is possible to form an interesting tour, and if so to find one. You can output any interesting tour that consists of at most m walks. It can be proven that if there exists an interesting tour, then there exists one consisting of at most m walks.

Input

The first line contains a positive integer t ($1 \leq t \leq 5 \cdot 10^5$) denoting the number of test cases.

The first line of each test case contains positive integers n and m ($2 \leq n$, $1 \leq m$) denoting the number of meeting points and walks, respectively.

Each of the subsequent m lines describes one of the m walks. The i-th line contains three positive integers x_i, y_i and c_i ($1 \leq x_i, y_i \leq n$, $x_i \neq y_i$, $1 \leq c_i \leq m$), which indicate that the i-th walk starts at the meeting point x_i, ends at the meeting point y_i, and allows us to see the attraction c_i.

Let N and M denote the sum of n and m, respectively, over all test cases. You can assume that $N, M \leq 10^6$.

Output

For each test case, in the first line you should output `YES` if it is possible to organise an interesting tour and `NO` otherwise. In the former case, the second line should first contain a positive integer k ($2 \leq k \leq m$) denoting the number of walks forming the interesting tour. This should be followed by k integers $p_1, p_2, \ldots, p_k$ separated by single spaces. These numbers should describe an interesting tour, where we first follow walk p_1, then p_2, and so on, and finally we follow walk p_k returning to the original meeting point.

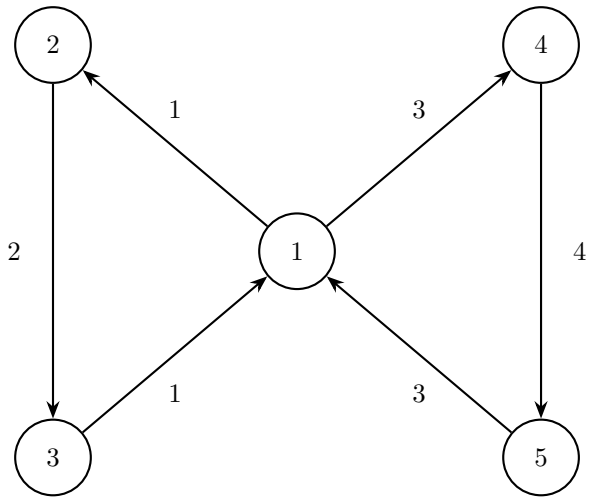


Illustration of the 4th test case from the example. The arrows represent the walks between meeting points.

Example

For the input data:

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

one of the correct results is:

```
NO
YES
2 2 3
NO
YES
6 3 4 5 6 1 2
YES
4 2 4 2 3
```

Scoring

Subtask	Constraints	Points
1	$m \leq 10$ and $t \leq 100$	9
2	$M \leq 5000$	23
3	$M \leq 5 \cdot 10^4$	19
4	$M \leq 2 \cdot 10^5$	25
5	No additional constraints.	24

Task: TOW
Tower

BOI 2025, Day 1. Available memory: 256 MB. *2025.04.26*

There are many legends concerning the Leaning Tower of Toruń. The wall of the tower is a circle with $N \geq 3$ evenly spaced doors (in other words, the doors are the vertices of a regular N-gon). The doors are numbered from 0 to $N-1$, but in a **random order**. Please refer to the scoring section for more details about this.

One of the less known legends describes how every new inhabitant of the tower had to complete a certain challenge. The goal of the challenge was to list the doors, starting with some door and then walking around the circle (clockwise or counterclockwise), visiting each door exactly once.

This needs to be done without actually seeing the tower. Instead, the new inhabitant can ask questions of the following form: "Given three distinct doors x, y, z, which pairs of doors are the closest to each other: $\{x, y\}$, $\{y, z\}$, or $\{z, x\}$?". The answer to such a question are all pairs (among $\{x, y\}$, $\{y, z\}$ and $\{z, x\}$) of doors with the smallest Euclidean distance. The distance is simply the length of the shortest segment connecting the doors. Your task is to write a program that will ask a small number of such questions to determine the order of the doors.

Interaction

This is an interactive task. You should write a program which finds a correct solution to the task and communicates with the interactor by reading from the standard input and writing to the standard output.

At the beginning of the interaction, your program should read two integers t and k ($1 \leq t \leq 100$, $1 \leq k \leq 12\,000$) from the standard input, denoting the number of test cases and the maximum allowed average number of queries, respectively. See the scoring section for more information about the latter.

For each test case, your program should first read a single integer n ($3 \leq n \leq 500$) from the standard input, denoting the number of doors in the tower.

Then your program should ask the questions in the following way:

- Your program should write a single line in the form of

  `?` x y z

  to the standard output, where x, y, and z are distinct integers ($0 \leq x, y, z \leq n-1$). This line represents a single question concerning doors x, y, and z.
- The response will be given as:

  r
  a_1 b_1
  ...
  a_r b_r

  where r is an integer ($1 \leq r \leq 3$) representing the number of pairs of doors with the smallest distance. Each such pair is described by two integers a_i and b_i ($a_i, b_i \in \{x, y, z\}$ and $a_i < b_i$).

Once you have determined the order of the doors, you should write a single line in the form of

`!` x_0 x_1 ... x_{n-1}

to the standard output, where $x_0, x_1, \ldots, x_{n-1}$ is the order of the doors as described in the task statement. Please note that there are exactly $2n$ possible correct answers since you can output the order starting from any door and then going in either direction. Any of them will be accepted.

Keep in mind that after each query or answer you have to flush the output buffer using `cout.flush()` (or `fflush(stdout)` if using `printf`) in C++ or `sys.stdout.flush()` in Python. Otherwise your program may receive a `Time Limit Exceeded` verdict.

After writing the answer to the interactor, your program should immediately proceed to the next test case or end the interaction if all test cases have been processed.

Your program cannot open any files or use any other resources. It can use the standard error stream for debugging purposes, but please mind that writing to this stream takes time.

Please also note that the interactor is not adaptive, meaning that the initial order of the doors is fixed beforehand in each test case and does not change during the interaction.

Example interaction

Suppose we have only one test case with $n = 6$, and the order of the doors is $5, 3, 0, 2, 1, 4$. The interaction could look as follows:

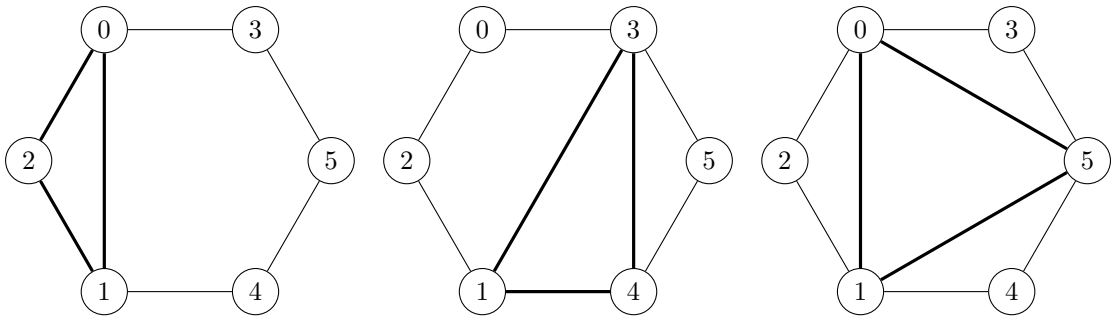
Interactor	Your program	Comment
`1 100`		$t = 1$ and $k = 100$.
`6`		Interactor gives the number of doors in the first test case.
	`? 0 1 2`	Your program asks which pairs of doors are the closest.

`2`
`0 2`
`1 2`
		Pairs of doors $\{0, 2\}$ and $\{1, 2\}$ are the closest.

	`? 4 1 3`	Your program asks which pairs of doors are the closest.
`1` `1 4`		Pair $\{1, 4\}$ is the closest.
	`? 0 5 1`	Your program asks which pairs of doors are the closest.

`3`
`0 5`
`0 1`
`1 5`
		Pairs $\{0, 5\}$, $\{0, 1\}$, and $\{1, 5\}$ are the closest.

	`! 4 5 3 0 2 1`	Your program correctly outputs the order of the doors.



Explanation of the example: The pictures above show the doors with their numbers along the walls of the tower. In the first picture from the left a triangle formed by the doors with numbers $0, 1, 2$ is shown, corresponding to the first query of your program. We can see that the pairs $\{0, 2\}$ and $\{1, 2\}$ are the closest. In the middle picture a triangle formed by the doors with numbers $1, 4, 3$ is shown, corresponding to the second query of your program. We can clearly see that the pair $\{1, 4\}$ is the closest. In the third picture from the left a triangle formed by the doors with numbers $0, 1, 5$ is shown, corresponding to the third query of your program. We can clearly see that all the pairs of doors are equally close to each other.

Please note that the sequences $0, 2, 1, 4, 5, 3$ or $5, 4, 1, 2, 0, 3$ (and a couple others) would also be correct answers in this case.

Scoring

Scoring for this problem is divided into subtasks. For each subtask there is exactly one test and this single test contains exactly $t = 100$ test cases. For each test, the average number of queries asked by your program is calculated by taking the total number of queries among all test cases and dividing it by the number of test cases. If this average is greater than k for a given subtask, you will receive a score of 0 for that subtask. Otherwise, for subtasks 1 to 4, you will receive full score for that subtask.

For the last subtask, your score will be calculated as follows. Let k^* be the actual average number of queries asked by your program. Then, the number of points is given by the following formula:

$$\left\lceil 56 \cdot \min\left(1, \frac{12000 - k^*}{7800}\right)\right\rceil,$$

meaning that your score increases linearly from 0 to 56 as k^* goes from 12000 to 4200.

Please note that if your program gives an incorrect answer to any test case, you will receive a score of 0 for that subtask regardless of the number of queries asked.

The additional constraints for each subtask are in the table below.

Subtask	Constraints	Points
1	$k = 8000, 3 \leq n \leq 9$	6
2	$k = 4500, 40 \leq n \leq 50$	7
3	$k = 3000, 90 \leq n \leq 100$	9
4	$k = 4500, n = 400$, there is a correct answer $x_0, \ldots, x_{n-1}$ where $x_i = i$ for $200 \leq i \leq 399$	22
5	$k = 12\,000, n = 500$	up to 56

Moreover, you can assume that each test case has been generated by first choosing n **uniformly at random** from all values of n satisfying the constraints of a given subtask, and then choosing the order of the doors **uniformly at random** from all orders of n doors satisfying the constraints of a given subtask.