Codeforces Round #693 (Div. 3)

A. Cards for Friends

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

For the New Year, Polycarp decided to send postcards to all his n friends. He wants to make postcards with his own hands. For this purpose, he has a sheet of paper of size $w \times h$, which can be cut into pieces.

Polycarp can cut any sheet of paper $w \times h$ that he has in only two cases:

- If w is even, then he can cut the sheet in half and get two sheets of size $\frac{w}{2} \times h$;
- If h is even, then he can cut the sheet in half and get two sheets of size $w \times \frac{h}{2}$;

If w and h are even at the same time, then Polycarp can cut the sheet according to any of the rules above.

After cutting a sheet of paper, the total number of sheets of paper is increased by 1.

Help Polycarp to find out if he can cut his sheet of size $w \times h$ at into n or more pieces, using only the rules described above.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing three integers w, h, n ($1 \le w, h \le 10^4, 1 \le n \le 10^9$) — the width and height of the sheet Polycarp has and the number of friends he needs to send a postcard to.

Output

For each test case, output on a separate line:

- "YES", if it is possible to cut a sheet of size $w \times h$ into at least n pieces;
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

Example

input
5 2 2 3 3 3 2 5 10 2 11 13 1 1 4 4
output
YES NO YES YES YES

Note

In the first test case, you can first cut the 2×2 sheet into two 2×1 sheets, and then cut each of them into two more sheets. As a result, we get four sheets 1×1. We can choose any three of them and send them to our friends.

In the second test case, a 3×3 sheet cannot be cut, so it is impossible to get two sheets.

In the third test case, you can cut a 5×10 sheet into two 5×5 sheets.

In the fourth test case, there is no need to cut the sheet, since we only need one sheet.

In the fifth test case, you can first cut the 1×4 sheet into two 1×2 sheets, and then cut each of them into two more sheets. As a result, we get four sheets 1×1.

B. Fair Division

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob received n candies from their parents. **Each candy weighs either 1 gram or 2 grams**. Now they want to divide all candies among themselves fairly so that the total weight of Alice's candies is equal to the total weight of Bob's candies.

Check if they can do that.

Note that candies **are not allowed to be cut in half**.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 100$) — the number of candies that Alice and Bob received.

The next line contains n integers $a_1, a_2, \ldots, a_n$ — the weights of the candies. The weight of each candy is either 1 or 2.

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

Output
For each test case, output on a separate line:

- "YES", if all candies can be divided into two sets with the same weight;
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

Example

input
5 2 1 1 2 1 2 4 1 2 1 2 3 2 2 2 3 2 1 2
output
YES NO YES NO NO

Note
In the first test case, Alice and Bob can each take one candy, then both will have a total weight of 1.

In the second test case, any division will be unfair.

In the third test case, both Alice and Bob can take two candies, one of weight 1 and one of weight 2.

In the fourth test case, it is impossible to divide three identical candies between two people.

In the fifth test case, any division will also be unfair.

C. Long Jumps

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp found under the Christmas tree an array a of n elements and instructions for playing with it:

- At first, choose index i ($1 \le i \le n$) — starting position in the array. Put the chip at the index i (on the value a_i).
- While $i \le n$, add a_i to your score and move the chip a_i positions to the right (i.e. replace i with $i + a_i$).
- If $i > n$, then Polycarp ends the game.

For example, if $n = 5$ and $a = [7, 3, 1, 2, 3]$, then the following game options are possible:

- Polycarp chooses $i = 1$. Game process: $i = 1 \xrightarrow{+7} 8$. The score of the game is: $a_1 = 7$.
- Polycarp chooses $i = 2$. Game process: $i = 2 \xrightarrow{+3} 5 \xrightarrow{+3} 8$. The score of the game is: $a_2 + a_5 = 6$.
- Polycarp chooses $i = 3$. Game process: $i = 3 \xrightarrow{+1} 4 \xrightarrow{+2} 6$. The score of the game is: $a_3 + a_4 = 3$.
- Polycarp chooses $i = 4$. Game process: $i = 4 \xrightarrow{+2} 6$. The score of the game is: $a_4 = 2$.
- Polycarp chooses $i = 5$. Game process: $i = 5 \xrightarrow{+3} 8$. The score of the game is: $a_5 = 3$.

Help Polycarp to find out the maximum score he can get if he chooses the starting index in an optimal way.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, output on a separate line one number — the maximum score that Polycarp can get by playing the game on the corresponding array according to the instruction from the statement. Note that Polycarp chooses any starting position from 1 to n in such a way as to maximize his result.

Example

input
4 5 7 3 1 2 3 3 2 1 4 6 2 1000 2 3 995 1 5 1 1 1 1 1
output
7 6 1000 5

Note
The first test case is explained in the statement.

In the second test case, the maximum score can be achieved by choosing $i = 1$.

In the third test case, the maximum score can be achieved by choosing $i = 2$.

In the fourth test case, the maximum score can be achieved by choosing $i = 1$.

D. Even-Odd Game

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

During their New Year holidays, Alice and Bob play the following game using an array a of n integers:

- Players take turns, Alice moves first.
- Each turn a player chooses any element and removes it from the array.
- If Alice chooses **even value**, then she adds it to her score. If the chosen value is odd, Alice's score does not change.
- Similarly, if Bob chooses **odd value**, then he adds it to his score. If the chosen value is even, then Bob's score does not change.

If there are no numbers left in the array, then the game ends. The player with the highest score wins. If the scores of the players are equal, then a draw is declared.

For example, if $n = 4$ and $a = [5, 2, 7, 3]$, then the game could go as follows (there are other options):

- On the first move, Alice chooses 2 and get two points. Her score is now 2. The array a is now $[5, 7, 3]$.
- On the second move, Bob chooses 5 and get five points. His score is now 5. The array a is now $[7, 3]$.
- On the third move, Alice chooses 7 and get no points. Her score is now 2. The array a is now $[3]$.
- On the last move, Bob chooses 3 and get three points. His score is now 8. The array a is empty now.
- Since Bob has more points at the end of the game, he is the winner.

You want to find out who will win if both players play optimally. **Note that there may be duplicate numbers in the array**.

Input
The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in the array a.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the array a used to play the game.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, output on a separate line:

- "Alice" if Alice wins with the optimal play;
- "Bob" if Bob wins with the optimal play;
- "Tie", if a tie is declared during the optimal play.

Example

input
4 4 5 2 7 3 3 3 2 1 4 2 2 2 2 2 7 8
output
Bob Tie Alice Alice

E. Correct Placement

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has invited n friends to celebrate the New Year. During the celebration, he decided to take a group photo of all his friends. Each friend can stand or lie on the side.

Each friend is characterized by two values h_i (their height) and w_i (their width). On the photo the i-th friend will occupy a rectangle $h_i \times w_i$ (if they are standing) or $w_i \times h_i$ (if they are lying on the side).

The j-th friend can be placed in front of the i-th friend on the photo if his rectangle is lower and narrower than the rectangle of the i-th friend. Formally, **at least one** of the following conditions must be fulfilled:

- $h_j < h_i$ **and** $w_j < w_i$ (both friends are standing or both are lying);
- $w_j < h_i$ **and** $h_j < w_i$ (one of the friends is standing and the other is lying).

For example, if $n = 3$, $h = [3, 5, 3]$ and $w = [4, 4, 3]$, then:

- the first friend can be placed in front of the second: $w_1 < h_2$ and $h_1 < w_2$ (one of the them is standing and the other one is lying);
- the third friend can be placed in front of the second: $h_3 < h_2$ and $w_3 < w_2$ (both friends are standing or both are lying).

In other cases, the person in the foreground will overlap the person in the background.

Help Polycarp for each i find any j, such that the j-th friend can be located in front of the i-th friend (i.e. at least one of the conditions above is fulfilled).

Please note that you do not need to find the arrangement of all people for a group photo. You just need to find for each friend i any other friend j who can be located in front of him. Think about it as you need to solve n separate independent subproblems.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of friends.

This is followed by n lines, each of which contains a description of the corresponding friend. Each friend is described by two integers h_i and w_i ($1 \le h_i, w_i \le 10^9$) — height and width of the i-th friend, respectively.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case output n integers on a separate line, where the i-th number is the index of a friend that can be placed in front of the i-th. If there is no such friend, then output -1.

If there are several answers, output any.

Example

input
4 3

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

output

```
-1 3 -1
-1 -1 -1
-1 -1 2 2
3 3 -1 3
```

Note

The first test case is described in the statement.

In the third test case, the following answers are also correct:

- $[-1, -1, 1, 2]$;
- $[-1, -1, 1, 1]$;
- $[-1, -1, 2, 1]$.

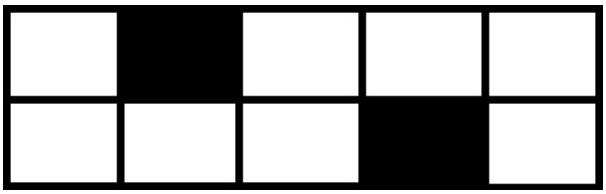
F. New Year's Puzzle

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

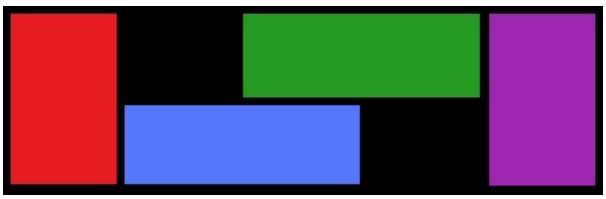
Every year Santa Claus gives gifts to all children. However, each country has its own traditions, and this process takes place in different ways. For example, in Berland you need to solve the New Year's puzzle.

Polycarp got the following problem: given a grid strip of size $2 \times n$, some cells of it are blocked. You need to check if it is possible to tile all free cells using the 2×1 and 1×2 tiles (dominoes).

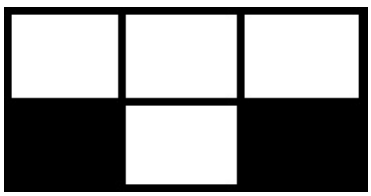
For example, if $n = 5$ and the strip looks like this (black cells are blocked):



Then it can be tiled, for example, using two vertical and two horizontal tiles, as in the picture below (different tiles are marked by different colors).



And if $n = 3$ and the strip looks like this:



It is impossible to tile free cells.

Polycarp easily solved this task and received his New Year's gift. Can you solve it?

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case is preceded by an empty line.

The first line of each test case contains two integers n and m ($1 \le n \le 10^9$, $1 \le m \le 2 \cdot 10^5$) — the length of the strip and the number of blocked cells on it.

Each of the next m lines contains two integers r_i, c_i ($1 \le r_i \le 2, 1 \le c_i \le n$) — numbers of rows and columns of blocked cells. It is guaranteed that all blocked cells are different, i.e. $(r_i, c_i) \neq (r_j, c_j), i \neq j$.

It is guaranteed that the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print on a separate line:

- "YES", if it is possible to tile all unblocked squares with the 2×1 and 1×2 tiles;
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

Example

input
3 5 2 2 2 1 4 3 2 2 1 2 3 6 4 2 1 2 3 2 4 2 6
output
YES NO NO

Note

The first two test cases are explained in the statement.

In the third test case the strip looks like this:

It is easy to check that the unblocked squares on it can not be tiled.

G. Moving to the Capital

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

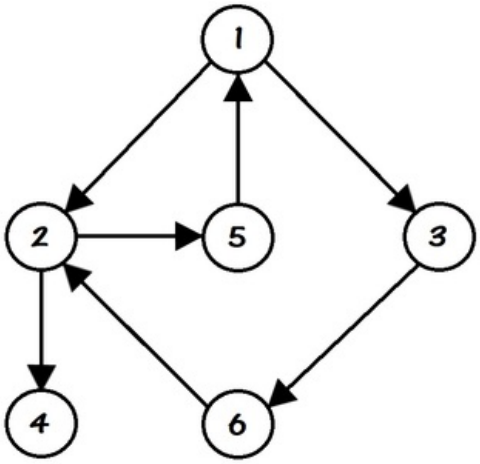
There are n cities in Berland. The city numbered 1 is the capital. Some pairs of cities are connected by a **one-way** road of length 1.

Before the trip, Polycarp for each city found out the value of d_i — the shortest distance from the capital (the 1-st city) to the i-th city.

Polycarp begins his journey in the city with number s and, being in the i-th city, chooses one of the following actions:

1. Travel from the i-th city to the j-th city if there is a road from the i-th city to the j-th and $d_i < d_j$;
2. Travel from the i-th city to the j-th city if there is a road from the i-th city to the j-th and $d_i \ge d_j$;
3. Stop traveling.

Since the government of Berland does not want all people to come to the capital, so Polycarp **no more than once** can take the second action from the list. in other words, he can perform the second action 0 or 1 time during his journey. Polycarp, on the other hand, wants to be as close to the capital as possible.



For example, if $n = 6$ and the cities are connected, as in the picture above, then Polycarp could have made the following travels (not all possible options):

- $2 \to 5 \to 1 \to 2 \to 5$;
- $3 \to 6 \to 2$;
- $1 \to 3 \to 6 \to 2 \to 5$.

Polycarp wants for each starting city i to find out how close he can get to the capital. More formally: he wants to find the minimal value of d_j that Polycarp can get from the city i to the city j according to the rules described above.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case is preceded by an empty line.

The first line of each test case contains two integers n ($2 \le n \le 2 \cdot 10^5$) and m ($1 \le m \le 2 \cdot 10^5$) — number of cities and roads, respectively.

This is followed by m lines describing the roads. Each road is characterized by two integers u and v ($1 \le u, v \le n, u \ne v$) — the numbers of cities connected by a **one-way** road.

It is guaranteed that the sums of n and m over all test cases do not exceed $2 \cdot 10^5$.

It is guaranteed that for each pair of different cities (u, v) there is at most one road from u to v (but a pair of roads from u to v and from v to u — is valid).

It is guaranteed that there is a path from the capital to all cities.

Output

For each test case, on a separate line output n numbers, the i-th of which is equal to the minimum possible distance from the capital to the city where Polycarp ended his journey.

Example

input
3 6 7 1 2 1 3 2 5 2 4 5 1 3 6 6 2 2 2 1 2 2 1 6 8 1 2 1 5 2 6 6 1 2 3 3 4 4 2 5 4
output
0 0 1 2 0 1 0 0 0 0 2 1 1 0