## Codeforces Round #766 (Div. 2)

# A. Not Shading

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

There is a grid with $n$ rows and $m$ columns. Some cells are colored black, and the rest of the cells are colored white.

In one operation, you can select some **black** cell and do **exactly one** of the following:

- color all cells in its row black, or
- color all cells in its column black.

You are given two integers $r$ and $c$. Find the minimum number of operations required to make the cell in row $r$ and column $c$ black, or determine that it is impossible.

### Input

The input consists of multiple test cases. The first line contains an integer $t$ ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains four integers $n$, $m$, $r$, and $c$ ($1 \le n, m \le 50$; $1 \le r \le n$; $1 \le c \le m$) — the number of rows and the number of columns in the grid, and the row and column of the cell you need to turn black, respectively.

Then $n$ lines follow, each containing $m$ characters. Each of these characters is either 'B' or 'W' — a black and a white cell, respectively.

### Output

For each test case, if it is impossible to make the cell in row $r$ and column $c$ black, output $-1$.

Otherwise, output a single integer — the minimum number of operations required to make the cell in row $r$ and column $c$ black.

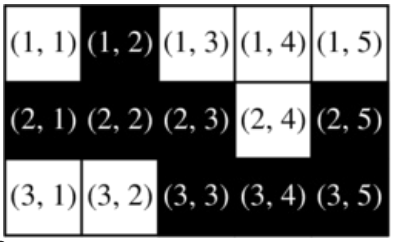
**Example**

| input |
| --- |
| 9<br>3 5 1 4<br>WBWWW<br>BBBWB<br>WWBBB<br>4 3 2 1<br>BWW<br>BBW<br>WBB<br>WWB<br>2 3 2 2<br>WWW<br>WWW<br>2 2 1 1<br>WW<br>WB<br>5 9 5 9<br>WWWWWWWWW<br>WBWBWBBBW<br>WBBBWWBWW<br>WBWBWBBBW<br>WWWWWWWWW<br>1 1 1 1<br>B<br>1 1 1 1<br>W<br>1 2 1 1<br>WB<br>2 1 1 1<br>W<br>B |

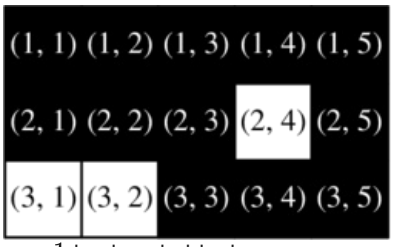
| output |
| --- |
| 1<br>0<br>-1<br>2<br>2<br>0<br>-1<br>1<br>1 |

**Note**

The first test case is pictured below.



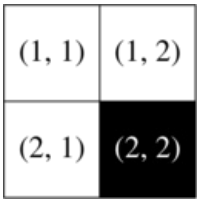

We can take the black cell in row $1$ and column $2$, and make all cells in its row black. Therefore, the cell in row $1$ and column $4$ will become black.



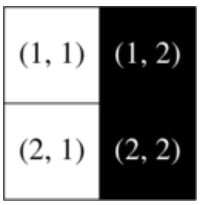

In the second test case, the cell in row $2$ and column $1$ is already black.

In the third test case, it is impossible to make the cell in row $2$ and column $2$ black.

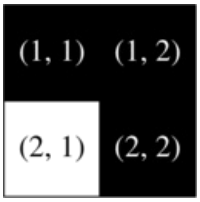
The fourth test case is pictured below.




We can take the black cell in row $2$ and column $2$ and make its column black.




Then, we can take the black cell in row $1$ and column $2$ and make its row black.




Therefore, the cell in row $1$ and column $1$ will become black.

## B. Not Sitting

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

Rahul and Tina are looking forward to starting their new year at college. As they enter their new classroom, they observe the seats of students are arranged in a $n \times m$ grid. The seat in row $r$ and column $c$ is denoted by $(r, c)$, and the distance between two seats $(a, b)$ and $(c, d)$ is $|a - c| + |b - d|$.

As the class president, Tina has access to **exactly** $k$ buckets of pink paint. The following process occurs.

- First, Tina chooses exactly $k$ seats in the classroom to paint with pink paint. One bucket of paint can paint exactly one seat.
- After Tina has painted $k$ seats in the previous step, Rahul chooses where he sits. He will not choose a seat that has been painted pink due to his hatred of the colour pink.
- After Rahul has chosen his seat, Tina chooses a seat for herself. She can choose any of the seats, painted or not, other than the one chosen by Rahul.

Rahul wants to choose a seat such that he sits as close to Tina as possible. However, Tina wants to sit as far away from Rahul as possible due to some complicated relationship history that we couldn't fit into the statement!

Now, Rahul wonders for $k = 0, 1, \ldots, n \cdot m - 1$, if Tina has $k$ buckets of paint, how close can Rahul sit to Tina, if both Rahul and Tina are aware of each other's intentions and they both act as strategically as possible? Please help satisfy Rahul's curiosity!

### Input

The input consists of multiple test cases. The first line contains an integer $t$ ($1 \leq t \leq 5 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers $n$, $m$ ($2 \leq n \cdot m \leq 10^5$) — the number of rows and columns of seats in the classroom.

The sum of $n \cdot m$ across all test cases does not exceed $10^5$.

### Output

For each test case, output $n \cdot m$ ordered integers — the distance between Rahul and Tina if both of them act optimally for every $k \in [0, n \cdot m - 1]$.

**Example**

| input |
| --- |
| 2<br>4 3<br>1 2 |
| **output** |
| 3 3 4 4 4 4 4 4 5 5 5 5<br>1 1 |

### Note

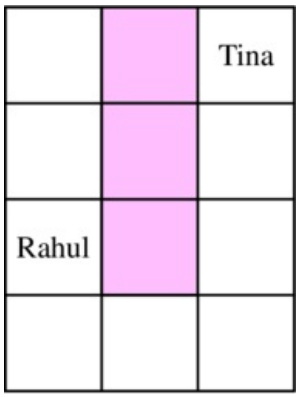
One possible sequence of choices for the first testcase where Tina has $k = 3$ buckets of paints is as follows.

Tina paints the seats at positions $(1, 2)$, $(2, 2)$, $(3, 2)$ with pink paint. Rahul chooses the seat at $(3, 1)$ after which Tina chooses to sit at $(1, 3)$.

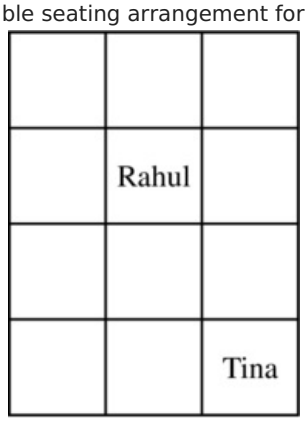
Therefore, the distance between Tina and Rahul is $|3 - 1| + |1 - 3| = 4$, and we can prove that this is indeed the minimum possible distance under the given constraints. There may be other choices of seats which lead to the same answer as well.

For $k = 0$ in the first test case, Rahul can decide to sit at $(2, 2)$ and Tina can decide to sit at $(4, 3)$ so the distance between them would be $|2 - 4| + |2 - 3| = 3$.

Below are pictorial representations of the $k = 3$ and $k = 0$ cases for the first test case.




A possible seating arrangement for $k = 3$.




A possible seating arrangement for $k = 0$.

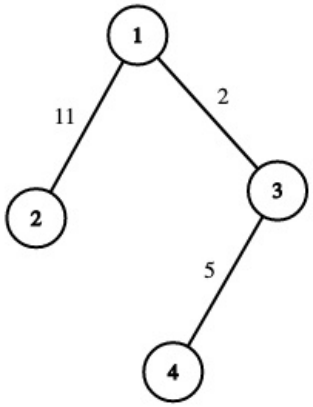
## C. Not Assigning

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a tree of $n$ vertices numbered from $1$ to $n$, with edges numbered from $1$ to $n-1$. A tree is a connected undirected graph without cycles. You have to assign integer weights to each edge of the tree, such that the resultant graph is a prime tree.

A *prime tree* is a tree where the weight of every path consisting of **one or two edges** is prime. A path should not visit any vertex twice. The weight of a path is the sum of edge weights on that path.

Consider the graph below. It is a prime tree as the weight of every path of two or less edges is prime. For example, the following path of two edges: $2 \to 1 \to 3$ has a weight of $11 + 2 = 13$, which is prime. Similarly, the path of one edge: $4 \to 3$ has a weight of $5$, which is also prime.




Print **any** valid assignment of weights such that the resultant tree is a prime tree. If there is no such assignment, then print $-1$. It can be proven that if a valid assignment exists, one exists with weights between $1$ and $10^5$ as well.

## Input

The input consists of multiple test cases. The first line contains an integer $t$ ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains one integer $n$ ($2 \leq n \leq 10^5$) — the number of vertices in the tree.

Then, $n-1$ lines follow. The $i$-th line contains two integers $u$ and $v$ ($1 \leq u, v \leq n$) denoting that edge number $i$ is between vertices $u$ and $v$. It is guaranteed that the edges form a tree.

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

## Output

For each test case, if a valid assignment exists, then print a single line containing $n-1$ integers $a_1, a_2, \ldots, a_{n-1}$ ($1 \leq a_i \leq 10^5$), where $a_i$ denotes the weight assigned to the edge numbered $i$. Otherwise, print $-1$.

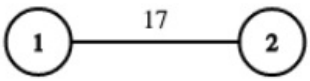
If there are multiple solutions, you may print any.

**Example**

| input |
| --- |
| 3<br>2<br>1 2<br>4<br>1 3<br>4 3<br>2 1<br>7<br>1 2<br>1 3<br>3 4<br>3 5<br>6 2<br>7 2 |
| **output** |
| 17<br>2 5 11<br>-1 |

## Note

For the first test case, there are only two paths having one edge each: $1 \to 2$ and $2 \to 1$, both having a weight of $17$, which is prime.




The second test case is described in the statement.

It can be proven that no such assignment exists for the third test case.

## D. Not Adding

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

You have an array $a_1, a_2, \ldots, a_n$ consisting of $n$ **distinct** integers. You are allowed to perform the following operation on it:

- Choose two elements from the array $a_i$ and $a_j$ ($i \neq j$) such that $\gcd(a_i, a_j)$ is not present in the array, and add $\gcd(a_i, a_j)$ to the end of the array. Here $\gcd(x, y)$ denotes greatest common divisor (GCD) of integers $x$ and $y$.

Note that the array changes after each operation, and the subsequent operations are performed on the new array.

What is the **maximum** number of times you can perform the operation on the array?

### Input
The first line consists of a single integer $n$ ($2 \leq n \leq 10^6$).

The second line consists of $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^6$). All $a_i$ are **distinct**.

### Output
Output a single line containing one integer — the maximum number of times the operation can be performed on the given array.

**Examples**

| input |
| --- |
| 5<br>4 20 1 25 30 |
| **output** |
| 3 |

| input |
| --- |
| 3<br>6 10 15 |
| **output** |
| 4 |

### Note
In the first example, one of the ways to perform maximum number of operations on the array is:

- Pick $i = 1, j = 5$ and add $\gcd(a_1, a_5) = \gcd(4, 30) = 2$ to the array.
- Pick $i = 2, j = 4$ and add $\gcd(a_2, a_4) = \gcd(20, 25) = 5$ to the array.
- Pick $i = 2, j = 5$ and add $\gcd(a_2, a_5) = \gcd(20, 30) = 10$ to the array.

It can be proved that there is no way to perform more than $3$ operations on the original array.

In the second example one can add $3$, then $1$, then $5$, and $2$.

## E. Not Escaping

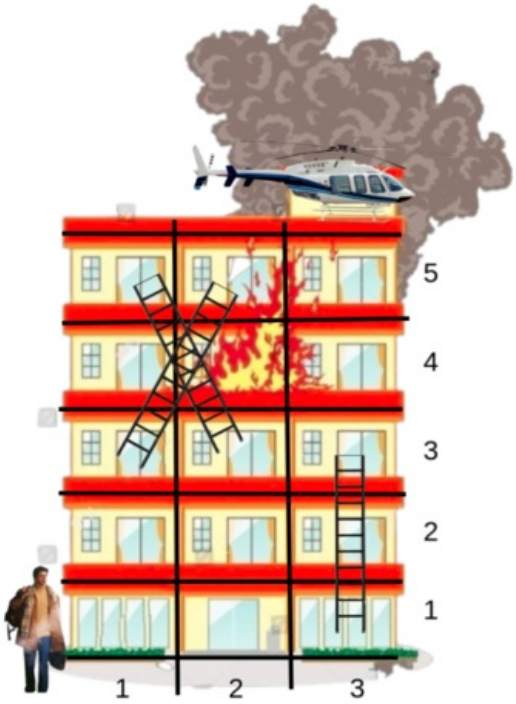
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Major Ram is being chased by his arch enemy Raghav. Ram must reach the top of the building to escape via helicopter. The building, however, is on fire. Ram must choose the optimal path to reach the top of the building to lose the minimum amount of health.

The building consists of $n$ floors, each with $m$ rooms each. Let $(i, j)$ represent the $j$-th room on the $i$-th floor. Additionally, there are $k$ ladders installed. The $i$-th ladder allows Ram to travel from $(a_i, b_i)$ to $(c_i, d_i)$, but **not in the other direction**. Ram also gains $h_i$ health points if he uses the ladder $i$. **It is guaranteed $a_i < c_i$ for all ladders.**

If Ram is on the $i$-th floor, he can move either left or right. Travelling across floors, however, is treacherous. If Ram travels from $(i, j)$ to $(i, k)$, he loses $|j - k| \cdot x_i$ health points.

Ram enters the building at $(1, 1)$ while his helicopter is waiting at $(n, m)$. What is the minimum amount of health Ram loses if he takes the most optimal path? Note this answer may be negative (in which case he gains health). Output "NO ESCAPE" if no matter what path Ram takes, he cannot escape the clutches of Raghav.




## Input

The first line of input contains $t$ ($1 \leq t \leq 5 \cdot 10^4$) — the number of test cases.

The first line of each test case consists of $3$ integers $n, m, k$ ($2 \leq n, m \leq 10^5$; $1 \leq k \leq 10^5$) — the number of floors, the number of rooms on each floor and the number of ladders respectively.

The second line of a test case consists of $n$ integers $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq 10^6$).

The next $k$ lines describe the ladders. Ladder $i$ is denoted by $a_i, b_i, c_i, d_i, h_i$ ($1 \leq a_i < c_i \leq n$; $1 \leq b_i, d_i \leq m$; $1 \leq h_i \leq 10^6$) — the rooms it connects and the health points gained from using it.

**It is guaranteed** $a_i < c_i$ **for all ladders** and there is **at most** one ladder between any 2 rooms in the building.

The sum of $n$, the sum of $m$, and the sum of $k$ over all test cases do not exceed $10^5$.

## Output

Output the minimum health Ram loses on the optimal path from $(1, 1)$ to $(n, m)$. If Ram cannot escape the clutches of Raghav regardless of the path he takes, output "NO ESCAPE" (all uppercase, without quotes).

**Example**

| input |
| --- |
| 4<br>5 3 3<br>5 17 8 1 4<br>1 3 3 3 4<br>3 1 5 2 5<br>3 2 5 1 6<br>6 3 3<br>5 17 8 1 4 2<br>1 3 3 3 4<br>3 1 5 2 5<br>3 2 5 1 6<br>5 3 1<br>5 17 8 1 4<br>1 3 5 3 100<br>5 5 5<br>3 2 3 7 5<br>3 5 4 2 1<br>2 2 5 4 5<br>4 4 5 2 3<br>1 2 4 2 2<br>3 3 5 2 4 |
| **output** |
| 16<br>NO ESCAPE<br>-90<br>27 |

## Note

The figure for the first test case is in the statement. There are only $2$ possible paths to $(n, m)$:

- Ram travels to $(1, 3)$, takes the ladder to $(3, 3)$, travels to $(3, 2)$, takes the ladder to $(5, 1)$, travels to $(5, 3)$ where he finally escapes via helicopter. The health lost would be

$$\begin{aligned} & x_1 \cdot |1-3| - h_1 + x_3 \cdot |3-2| - h_3 + x_5 \cdot |1-3| \\ = \; & 5 \cdot 2 - 4 + 8 \cdot 1 - 6 + 4 \cdot 2 \\ = \; & 16. \end{aligned}$$

- Ram travels to $(1, 3)$, takes the ladder to $(3, 3)$, travels to $(3, 1)$, takes the ladder to $(5, 2)$, travels to $(5, 3)$ where he finally escapes via helicopter. The health lost would be

$$\begin{aligned} & x_1 \cdot |1-3| - h_1 + x_3 \cdot |3-1| - h_2 + a_5 \cdot |2-3| \\ = \; & 5 \cdot 2 - 4 + 8 \cdot 2 - 5 + 4 \cdot 1 \\ = \; & 21. \end{aligned}$$

Therefore, the minimum health lost would be $16$.
In the second test case, there is no path to $(n, m)$.

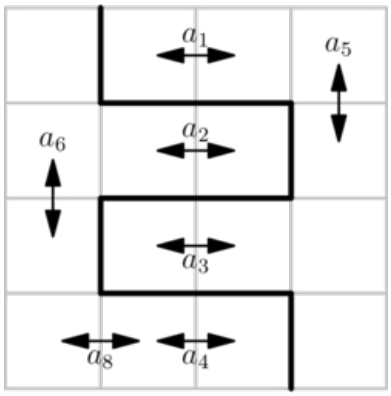
In the third case case, Ram travels to $(1, 3)$ and takes the only ladder to $(5, 3)$. He loses $5 \cdot 2$ health points and gains $h_1 = 100$ health points. Therefore the total loss is $10 - 100 = -90$ (negative implies he gains health after the path).

## F. Not Splitting

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There is a $k \times k$ grid, where $k$ is even. The square in row $r$ and column $c$ is denoted by $(r, c)$. Two squares $(r_1, c_1)$ and $(r_2, c_2)$ are considered *adjacent* if $|r_1 - r_2| + |c_1 - c_2| = 1$.

An array of adjacent pairs of squares is called *strong* if it is possible to cut the grid along grid lines into two connected, congruent pieces so that each pair is part of the **same** piece. Two pieces are congruent if one can be matched with the other by translation, rotation, and reflection, or a combination of these.




The picture above represents the first test case. Arrows indicate pairs of squares, and the thick black line represents the cut.
You are given an array $a$ of $n$ pairs of adjacent squares. Find the size of the largest strong subsequence of $a$. An array $p$ is a subsequence of an array $q$ if $p$ can be obtained from $q$ by deletion of several (possibly, zero or all) elements.

### Input

The input consists of multiple test cases. The first line contains an integer $t$ ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two space-separated integers $n$ and $k$ ($1 \le n \le 10^5$; $2 \le k \le 500$, $k$ **is even**) — the length of $a$ and the size of the grid, respectively.

Then $n$ lines follow. The $i$-th of these lines contains four space-separated integers $r_{i,1}$, $c_{i,1}$, $r_{i,2}$, and $c_{i,2}$ ($1 \le r_{i,1}, c_{i,1}, r_{i,2}, c_{i,2} \le k$) — the $i$-th element of $a$, represented by the row and column of the first square $(r_{i,1}, c_{i,1})$ and the row and column of the second square $(r_{i,2}, c_{i,2})$. **These squares are adjacent.**

It is guaranteed that the sum of $n$ over all test cases does not exceed $10^5$, and the sum of $k$ over all test cases does not exceed $500$.

### Output

For each test case, output a single integer — the size of the largest strong subsequence of $a$.

**Example**

| input |
| --- |
| 3<br>8 4<br>1 2 1 3<br>2 2 2 3<br>3 2 3 3<br>4 2 4 3<br>1 4 2 4<br>2 1 3 1 |

```
2 2 3 2
4 1 4 2
7 2
1 1 1 2
2 1 2 2
1 1 1 2
1 1 2 1
1 2 2 2
1 1 2 1
1 2 2 2
1 6
3 3 3 4
```

**output**

```
7
4
1
```

## Note

In the first test case, the array $a$ is not good, but if we take the subsequence $[a_1, a_2, a_3, a_4, a_5, a_6, a_8]$, then the square can be split as shown in the statement.

In the second test case, we can take the subsequence consisting of the last four elements of $a$ and cut the square with a horizontal line through its center.