

The 2022  icpc
Asia Nanjing Regional Contest

Contest Session

December 18, 2022



Problem List

A	Stop, Yesterday Please No More
B	Ropeway
C	Fabulous Fungus Frenzy
D	Chat Program
E	Color the Tree
F	Triangles
G	Inscription
H	Factories Once More
I	Perfect Palindrome
J	Perfect Matching
K	NaN in a Heap
L	Proposition Composition
M	Drain the Water Tank

This problem set should contain 13 (thirteen) problems on 26 (twenty six) numbered pages.
Please inform a runner immediately if something is missing from your problem set.

Hosted by



Problem Set Prepared by

签到成功 这是你的
签到奖励



SUA



It's against the rules to open non-contest websites during the contest.

If you're interested (which is our pleasure),
please scan the QR code only after the contest.

Problem A. Stop, Yesterday Please No More

After the great success in 2018, 2019, 2020 and 2021, Nanjing University of Aeronautics and Astronautics (NUAA) will host the *International Collegiate Programming Contest (ICPC)* Nanjing regional for the fifth time in a row.

Team *Power of Two* and team *Three Hold Two* won the champion title for Tsinghua University in 2018 and 2019. In 2020 and 2021, team *Inverted Cross* from Peking University won the back-to-back champion titles. They also won the second place in the 45th ICPC World Finals held in Dhaka, which is the best performance of East Continental teams in the past 6 years. We congratulate them and are very excited to see them coming back and compete in Nanjing 2022!

This year, there are around 500 teams participating in the contest. There are at most 35 gold medals, 70 silver medals and 105 bronze medals that will be awarded. We are looking forward to seeing participants' outstanding performance!

Although we can't gather in Nanjing this time (yet again) due to the pandemic, we should still be grateful for the hard work done by all staff and volunteers for this contest. Thank you all for your great contribution to this contest!



The 2018 ICPC Asia Nanjing Regional Contest

In the 2018 contest, problem K, *Kangaroo Puzzle*, requires the contestants to construct an operation sequence for the game:

The puzzle is a grid with n rows and m columns ($1 \leq n, m \leq 20$) and there are some (at least 2) kangaroos standing in the puzzle. The player's goal is to control them to get together. There are some walls in some cells and the kangaroos cannot enter the cells with walls. The other cells are empty. The kangaroos can move from an empty cell to an adjacent empty cell in four directions: up, down, left, and right.

There is exactly one kangaroo in every empty cell in the beginning and the player can control the kangaroos by pressing the button U, D, L, R on the keyboard. The kangaroos will move simultaneously according to the button you press.

The contestant needs to construct an operating sequence of at most 5×10^4 steps consisting of U, D, L, R only to achieve the goal.

In the 2020 contest, problem A, *Ah, It's Yesterday Once More*, requires the contestants to construct an input map to hack the following code of the problem described before:

```
#include <bits/stdc++.h>
using namespace std;
string s = "UDLR";
int main()
{
    srand(time(NULL));
    for (int i = 1; i <= 50000; i++) putchar(s[rand() % 4]);
    return 0;
}
```

In the 2021 contest, problem A, *Oops, It's Yesterday Twice More*, also requires the contestants to construct an operation sequence for the game:

This time, every cell in the grid stands exactly one kangaroo. You need to construct an operating sequence consisting only of characters 'U', 'D', 'L', and 'R'. After applying it, you must make sure every kangaroo will gather at the specific cell (a, b) . The length of the operating sequence cannot exceed $3(n - 1)$. As always, the kangaroos will move simultaneously according to the operation you command.

Now, in the 2022 contest, the kangaroo problem is back again! We don't know why problem setters are so obsessed with kangaroos but the problem is as follows:

You are given a grid with n rows and m columns. There is a hole in the cell on the i_h -th row and the j_h -th column. All other cells are empty and there is one kangaroo standing in each cell.

Similarly, the kangaroos are controlled by pressing the button U, D, L, R on the keyboard. All kangaroos will move simultaneously according to the button pressed. Specifically, for any kangaroo located in the cell on the i -th row and the j -th column, indicated by (i, j) :

1. Button U: it will move to $(i - 1, j)$.
2. Button D: it will move to $(i + 1, j)$.
3. Button L: it will move to $(i, j - 1)$.
4. Button R: it will move to $(i, j + 1)$.

If a kangaroo steps onto the hole (that is, $i = i_h$ and $j = j_h$) or steps out of the grid, it will be removed from the grid.

The problem is that, the exact value of i_h and j_h is not known. You're only given an operating sequence consisting only of characters 'U', 'D', 'L', and 'R', and an integer k indicating that after applying the operating sequence, there are k kangaroos remaining on the grid.

Calculate the number of possible positions of the hole. That is, calculate the number of integer pairs (i_h, j_h) such that:

- $1 \leq i_h \leq n, 1 \leq j_h \leq m$.
- The hole is located at (i_h, j_h) .
- After applying the given operating sequence, the number of kangaroos remaining on the grid is exactly k .

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains three integers n, m and k ($1 \leq n, m \leq 10^3, 0 \leq k < n \times m$) indicating the size of the grid and the number of kangaroos remaining on the grid after applying the operating sequence.

The second line contains a string $s_1s_2 \dots s_l$ ($s_i \in \{‘U’, ‘D’, ‘L’, ‘R’\}, 1 \leq l \leq 10^6$) indicating the operating sequence.

It’s guaranteed that neither the sum of $n \times m$ nor the total length of the operating sequences of each test case will exceed 10^6 .

Output

For each test case output one integer indicating the number of possible positions of the hole.

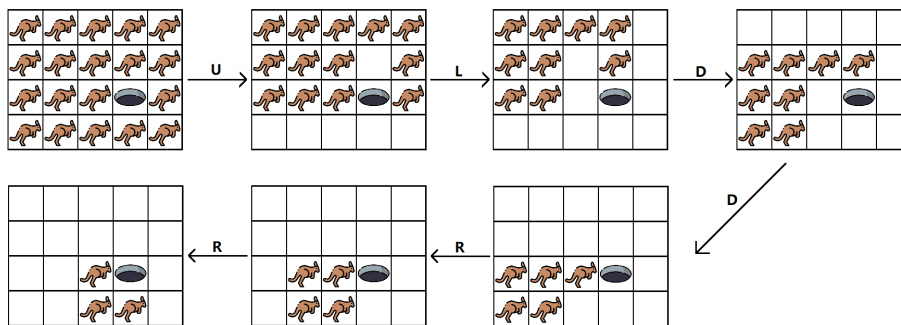
Example

standard input	standard output
3	2
4 5 3	20
ULDDRR	0
4 5 0	
UUUUUUU	
4 5 10	
UUUUUUU	

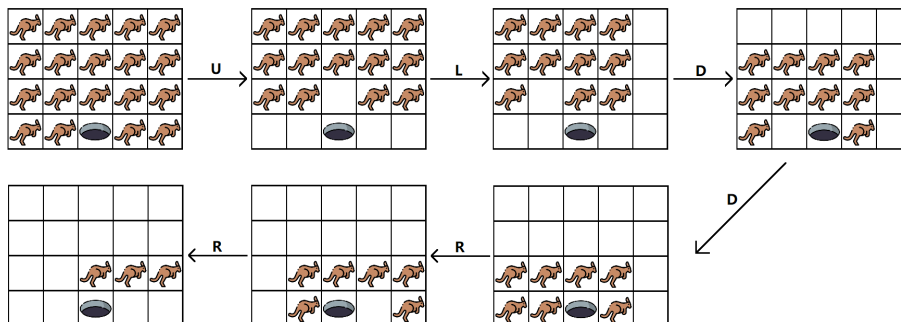
Note

For the first sample test case there are 2 possible positions for the hole.

The first possible position is (3, 4).



The second possible position is (4, 3).



Problem B. Ropeway

Purple Mountain (Zijin Shan) is one of the most famous mountain in Nanjing and its ropeway, the *Purple Mountain Ropeway*, used to be the longest chairlift in China. Tourists can take the ropeway from the foot of the mountain and go all the way up to the top in less than ten minutes.



The old Purple Mountain Ropeway. Photo by MiNe. Licensed under CC BY 2.0.

In order to build a ropeway, aside from the stations at the foot and the top of the mountain, we also need to build supporting towers along the way. The engineers have built the stations at 0 and $(n + 1)$ unit of distance from the entrance of the ropeway and have investigated the cost to build a supporting tower at $1, 2, \dots, n$ unit of distance, which are a_1, a_2, \dots, a_n respectively.

Due to some of the considerations on engineering, supporting towers must be built at some position. Also, to ensure the safety of the ropeway, the distance between neighboring supporting towers or stations must be less than or equal to k . That is, let $b_0, b_1, b_2, \dots, b_m, b_{m+1}$ be the final positions to build two stations and m supporting towers, where $0 \leq m \leq n$ and $0 = b_0 < b_1 < b_2 < \dots < b_m < b_{m+1} = n + 1$, we have $b_i - b_{i-1} \leq k$ for all $1 \leq i \leq m + 1$.

In the meantime, to make plans more flexible, the engineers will temporarily change the cost sequence for q times. The i -th change can be described as (p_i, v_i) , which means changing the value of a_{p_i} to v_i temporarily.

For each change, please calculate the minimum total cost to build supporting towers such that all requirements above can be satisfied.

Please note again that all changes are temporary and independent from each other. That is, after calculating the answer for a change, this change will be reverted and the cost sequence will be reverted back to its original state.

Input

There are multiple test cases. The first line of the input contain an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and k ($1 \leq n \leq 5 \times 10^5$, $1 \leq k \leq \min(n + 1, 3 \times 10^3)$) indicating the number of candidate positions to build supporting towers and the maximum distance between neighboring supporting towers or stations.

The second line contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$) where a_i indicates the cost to build a supporting tower at i unit of distance.

The third line contains a binary string $s_1s_2\cdots s_n$ of length n ($s_i \in \{0, 1\}$). If $s_i = 1$ you must build a supporting tower at i unit of distance. If $s_i = 0$ you can decide whether to build a supporting tower at i unit of distance.

The fourth line contains an integer q ($1 \leq q \leq 3 \times 10^3$) indicating the number of changes.

For the following q lines, the i -th line contains two integers p_i and v_i ($1 \leq p_i \leq n$, $1 \leq v_i \leq 10^9$) indicating the i -th change.

It's guaranteed that the sum of n of all test cases will not exceed 2×10^6 . It's also guaranteed that neither the sum of k nor the sum of q of all test cases will exceed 10^4 .

Output

For each change output one line containing one integer indicating the minimum total cost to build supporting towers after temporarily applying the change.

Example

standard input	standard output
3	206
10 3	214
5 10 7 100 4 3 12 5 100 1	0
0001000010	100
2	
2 3	
6 15	
5 6	
1 1 1 1 1	
00000	
1	
3 100	
5 6	
1 1 1 1 1	
00100	
1	
3 100	

Note

For the first sample test case:

- By applying the first change, the cost sequence is now $\{5, 3, 7, 100, 4, 3, 12, 5, 100, 1\}$. We should build supporting towers at 2, 4, 6 and 9 unit of distance to minimize the total cost.
- By applying the second change the cost sequence is now $\{5, 10, 7, 100, 4, 15, 12, 5, 100, 1\}$. We should build supporting towers at 1, 4, 5, 8 and 9 unit of distance to minimize the total cost.

For the second sample test case, no supporting towers are needed.

For the third sample test case we have to build a supporting tower at 3 unit of distance, so the cost is just a_3 .

Problem C. Fabulous Fungus Frenzy

As the Traveler, you are exploring the world of Teyvat in the world-famous game *Genshin Impact*. After visiting Monstadt, Liyue and Inazuma, now you are in the nation of Sumeru.

Fungi are an enemy group found in Sumeru. They are an evolved species of mushrooms with increased abilities to reproduce and protect the flora. Despite their lovely appearance, fungi are actually very dangerous creatures. They are considered Tri-Lakshana Creatures and thus Pyro and Electro will have adverse effects on them. When inflicted with Pyro, they enter a Scorched state, attacking significantly slower but dealing more damage. On the other hand, Electro causes them to enter an Activated state, attacking significantly faster. Since various kinds of fungi are able to discharge different skills of different elements, one can see that they can actually form a very strong team. You start to wonder the possibility of teaming up with fungi if you can make them listen to your instructions.



Luckily, in the event *Fabulous Fungus Frenzy*, you are able to control fungi and let them fight enemies! You are invited to compete in the *Nilotpala Cup Beast Tamers Tournament*. By the gadget called the Wisdom Orb, you can capture and put together your own team of Fungi. However, to win the tournament, you also need to pass the *Coruscating Potential* challenge to cultivate Fungi and unlock their special skills.

During the Coruscating Potential challenge, you must use *Floral Jellies* to form blends that your fungi enjoy. Once they have absorbed them, they will awaken their potential.



Specifically, you are given an initial configuration of the Floral Jellies blend, represented by an $n \times m$ matrix. For all $1 \leq i \leq n$ and $1 \leq j \leq m$, each entry (i, j) lies a Floral Jelly. The value of entries represents the type of Floral Jellies. Equal value of entries indicates that they are the same type of jellies.

You are allowed to perform 3 kinds of operations: Switch, Rotate and Preset.

Switch: For any two adjacent Floral Jellies, you can use one Switch operation to exchange the positions of them.

Specifically, two adjacent Floral Jellies located at (x_1, y_1) and (x_2, y_2) are considered adjacent if and only if $|x_1 - x_2| + |y_1 - y_2| = 1$. After the Switch, the Floral Jellies located at (x_1, y_1) and (x_2, y_2) will be equal to the previous ones located at (x_2, y_2) and (x_1, y_1) respectively.



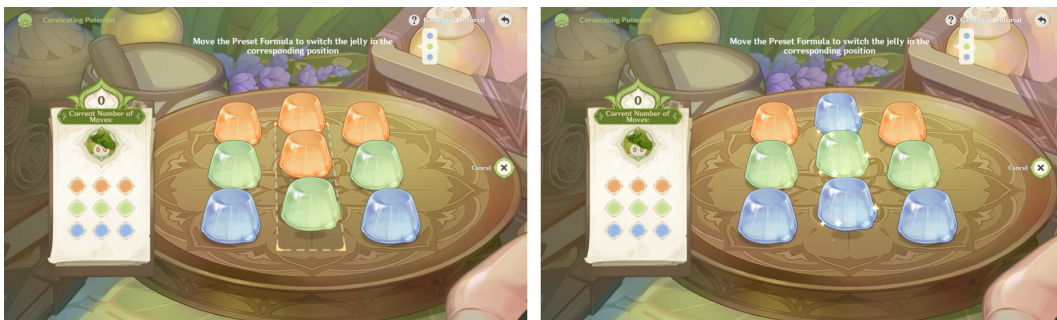
Rotate: For any four Floral Jellies forming a 2×2 block, you can use one Rotate operation to shift the positions of them once in a clockwise direction.

Specifically, a 2×2 block means arbitrary four Floral Jellies located at (x, y) , $(x, y + 1)$, $(x + 1, y + 1)$ and $(x + 1, y)$ satisfying $1 \leq x < n$ and $1 \leq y < m$. After the Rotate, the Floral Jellies located at (x, y) , $(x, y + 1)$, $(x + 1, y + 1)$ and $(x + 1, y)$ will be equal to the previous ones located at $(x + 1, y)$, (x, y) , $(x, y + 1)$ and $(x + 1, y + 1)$ respectively.



Preset: For any Floral Jellies forming a $n' \times m'$ block ($1 \leq n' \leq n$, $1 \leq m' \leq m$), you can place a pre-existing formula of the same size directly onto the corresponding slot and replace every Floral Jelly with the type on the formula. A preset formula is represented by a $n' \times m'$ matrix \mathbf{F} .

Specifically, for arbitrary Floral Jellies located at $(x + i, y + j)$ where $1 \leq x \leq n - n' + 1$, $1 \leq y \leq m - m' + 1$, $0 \leq i < n'$ and $0 \leq j < m'$, after a Preset operation with the preset formula \mathbf{F} , all Floral Jellies located at $(x + i, y + j)$ will be replaced by a Floral Jelly of the type in (i, j) of \mathbf{F} .



The challenge requires you to turn the Floral Jellies blends to the target configuration. That is, after processing all operations, each position must be occupied by a Floral Jelly of the required type.

Input

There is only one test case in each test file.

The first line contains three integers n , m and k ($2 \leq n, m \leq 20$, $1 \leq k \leq 20$) indicating the size of the initial configuration of the Floral Jellies and the number of preset formulas.

For the following n lines, the i -th line contains a string $s_{i,1}s_{i,2} \cdots s_{i,m}$ where $s_{i,j}$ indicates the type of jelly on the i -th row and the j -th column of the initial configuration.

Then an empty line follows.

For the following n lines, the i -th line contains a string $t_{i,1}t_{i,2} \cdots t_{i,m}$ where $t_{i,j}$ indicates the type of jelly on the i -th row and the j -th column of the target configuration.

Then k preset formulas follow. For the p -th preset formula:

There will first be an empty line.

The next line contains two integers n_p and m_p ($1 \leq n_p \leq n$, $1 \leq m_p \leq m$) indicating the matrix size of the preset formula.

For the following n_p lines, the i -th line contains a string $f_{i,1}^{(p)} f_{i,2}^{(p)} \cdots f_{i,m_p}^{(p)}$ where $f_{i,j}^{(p)}$ indicates the type of jelly on the i -th row and the j -th column of the p -th preset formula.

There are 62 types of Floral Jellies in all, denoted by lowercase letters ('a' to 'z'), uppercase letters ('A' to 'Z') and digits ('0' to '9') respectively. Two Floral Jellies are considered the same if and only if they are of the same character. All given strings will only consist of these 62 characters.

Output

If the puzzle is unsolvable, output “-1” (without quotes).

Otherwise, output an integer r ($0 \leq r \leq 4 \times 10^5$) in the first line indicating the number of moves needed to solve the puzzle.

Then output r lines, each of which contains three integers op , x and y , indicating an operation on the current Floral Jellies blend. You should output the operations in the order they are carried out. The available operations are listed as follows:

- $-4 \ x \ y$: Swaps the two jellies at (x, y) and $(x - 1, y)$. There must be $1 < x \leq n$ and $1 \leq y \leq m$.
- $-3 \ x \ y$: Swaps the two jellies at (x, y) and $(x + 1, y)$. There must be $1 \leq x < n$ and $1 \leq y \leq m$.
- $-2 \ x \ y$: Swaps the two jellies at (x, y) and $(x, y - 1)$. There must be $1 \leq x \leq n$ and $1 < y \leq m$.
- $-1 \ x \ y$: Swaps the two jellies at (x, y) and $(x, y + 1)$. There must be $1 \leq x \leq n$ and $1 \leq y < m$.
- $0 \ x \ y$: Rotates the four jellies at (x, y) , $(x, y + 1)$, $(x + 1, y + 1)$ and $(x + 1, y)$ in clockwise direction. There must be $1 \leq x < n$ and $1 \leq y < m$.
- $op \ x \ y$: Covers the submatrix that contains all the jellies at (i, j) where $x \leq i \leq x + n_{op} - 1$ and $y \leq j \leq y + m_{op} - 1$ with the op -th preset formula. There must be $1 \leq op \leq k$, $1 \leq x \leq n - n_{op} + 1$ and $1 \leq y \leq m - m_{op} + 1$.

Here a jelly at (x, y) means it lies in the x -th row from the top to the bottom and the y -th column from the left to the right.

In addition to the limitation of the total number of operations not exceeding 4×10^5 , we also have a special constraint on the preset operation. The total number of operations with $1 \leq op \leq k$ cannot exceed 400. It can be proven that if the puzzle is solvable, there always exists a solution satisfying all the constraints.

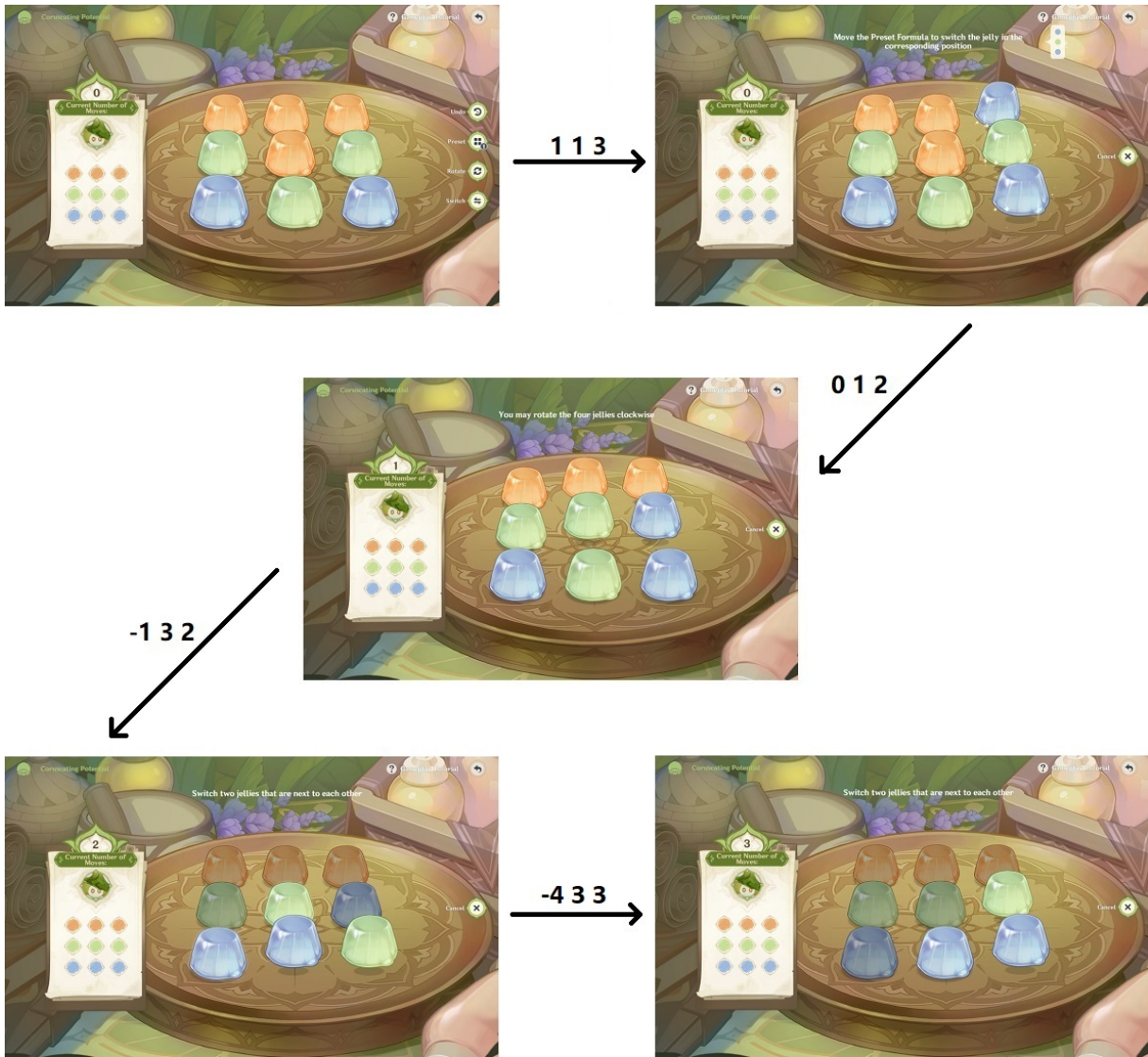
Note that you don't need to minimize the number of operations. If there are multiple valid solutions, output any.

Examples

standard input	standard output
<pre>3 3 1 000 GOG BGB 000 GGG BBB 3 1 B G B</pre>	<pre>4 1 1 3 0 1 2 -1 3 2 -4 3 3</pre>
<pre>2 2 1 00 00 PP PP 1 2 OP</pre>	<pre>-1</pre>
<pre>4 8 4 11122222 33344444 55556666 77777777 NIxSHUOx DExDUIxx DANxSHIx YUANSHEN 2 3 NIy DEx 3 8 zzzzzzzz DANNSH9I YUA9SHEN 1 1 x 2 5 SH08y DUUI8</pre>	<pre>13 2 2 1 -3 3 4 -2 3 8 1 1 1 4 1 4 0 1 6 3 1 3 3 1 8 3 2 3 3 2 7 3 2 8 3 3 4 3 3 8</pre>

Note

We explain the first sample test case below.



Problem D. Chat Program

You're the researcher of the *International Chat Program Company* (ICPC). Today, you discover the following chat history when reviewing some research data.

SUA (2022/12/04 23:01:25)

I'm out of ideas for competitive programming problems! Please give me a problem about sequences.

BOT (2022/12/04 23:01:27)

Sure. Here is a competitive programming problem about sequences.

Given an integer sequence a_1, a_2, \dots, a_n of length n and four other integers k, m, c and d , your goal is to maximize the k -th largest element in the sequence.

To achieve the goal, you can perform the following operation at most once: select a continuous sub-array of length m and add an arithmetic sequence with length m , initial term c and common difference d to the sub-array.

More formally, you can select an integer p satisfying $1 \leq p \leq n - m + 1$ and add $(c + di)$ to a_{p+i} for all $0 \leq i < m$.

Calculate the largest possible value of the k -th largest element in the sequence after at most one operation.

The k -th largest element in the sequence is the k -th element in the sorted sequence after sorting all elements from the largest to the smallest. For example, the 3rd largest element in sequence $\{5, 7, 1, 9\}$ is 5, while the 3rd largest element in sequence $\{9, 7, 5, 9\}$ is 7.

SUA (2022/12/05 00:15:17)

This problem seems difficult! Please teach me the solution.

BOT (2022/12/05 00:15:30)

Sure. Firstly, we can...

[DATA EXPUNGED]

Unfortunately, parts of the chat history are lost due to a disk failure. You're amazed at how a chat program can create a competitive programming problem. To verify whether the chat program can create valid problems, you decide to try on this problem.

Input

There is only one test case in each test file.

The first line contains five integers n, k, m, c and d ($1 \leq k, m \leq n \leq 2 \times 10^5$, $0 \leq c, d \leq 10^9$) indicating the length of the sequence, your goal, the length, initial term and common difference of the arithmetic sequence.

The second line contains n integers a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) indicating the sequence.

Output

Output one line containing one integer indicating the largest possible value of the k -th largest element in the sequence after at most one operation.

Examples


standard input	standard output
6 4 3 1 2 1 1 4 5 1 4	4
7 3 2 4 0 1 9 1 9 8 1 0	9
8 3 5 0 0 2 0 2 2 1 2 1 8	2


Note



For the first sample test case, we can choose $p = 3$ so the sequence becomes $\{1, 1, 5, 8, 6, 4\}$. The 4-th largest element in the sequence is 4.

For the second sample test case, we can choose $p = 5$ so the sequence becomes $\{1, 9, 1, 9, 12, 5, 0\}$. The 3-rd largest element in the sequence is 9.

For the third sample test case, it is easy to see that the operation does not change the sequence, so we choose not to perform the operation. The 3-rd largest element in the sequence is 2.

Could you encourage the competitors of a competitive programming contest? 



Sure, here are a few words of encouragement for competitors in a competitive programming contest:  

- Keep up the hard work! You've put in a lot of effort to get to this point, and it's important to keep pushing yourself.
- Don't be afraid to ask for help. If you're stuck on a problem, don't hesitate to reach out to others for guidance. → only your teammates!
- Stay positive. It's natural to feel frustrated when you're facing a difficult problem, but try to stay positive and keep moving forward.
- Take breaks when you need to. It's important to stay focused and avoid burnout. If you're feeling overwhelmed, take a break and come back to the problem later with a fresh mind.
- Remember that everyone has their own strengths and weaknesses. Don't compare yourself to others – focus on your own progress and improvement.

Overall, the most important thing is to have fun and enjoy the challenge. Good luck!

OpenAI ChatGPT is encouraging the contestants

Problem E. Color the Tree

There is a rooted tree of n vertices. The vertices are numbered from 1 to n (both inclusive) and the tree is rooted at 1. All of the n vertices are white at the beginning and you need to color all the vertices black.

To help you achieve the goal we will provide you with n types of operations, numbered from 0 to $(n - 1)$ (both inclusive). Operation i ($0 \leq i \leq n - 1$) requires you to first select a vertex u , then color all the vertices v satisfying the following conditions black:

- Vertex v is in the subtree rooted at u , which means $u = v$ or u is an ancestor of v .
- The distance between vertices u and v is exactly i . Here the distance between u and v is the minimum number of edges we need to go through to reach v from u .

The cost of performing operation i once is given as a_i . A vertex can be colored more than once and all operations can be performed any number of times. Calculate the minimum total cost to color all the vertices black.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($2 \leq n \leq 10^5$) indicating the size of the tree.

The second line contains n integers a_0, a_1, \dots, a_{n-1} ($1 \leq a_i \leq 10^9$) where a_i indicates the cost of performing operation i once.

For the following $(n - 1)$ lines, the i -th line contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$) indicating an edge connecting vertices u_i and v_i .

It's guaranteed that the sum of n of all test cases will not exceed 3×10^5 .

Output

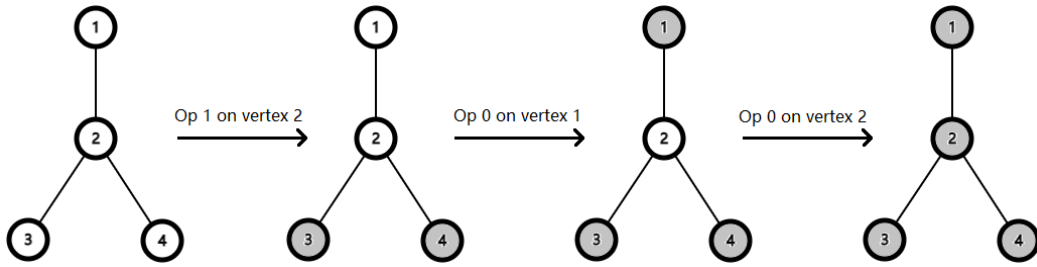
For each test case output one line containing one integer indicating the minimum total cost to color the tree black.

Example

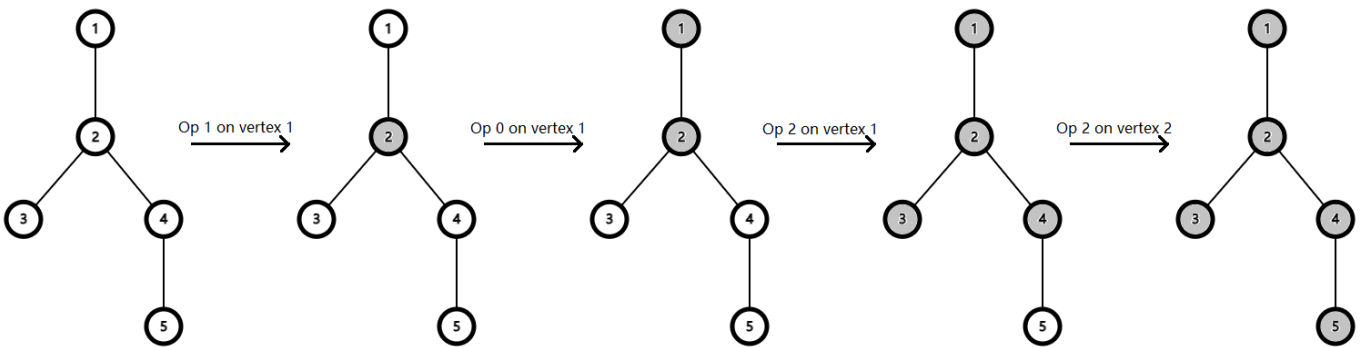
standard input	standard output
3	35
4	17
10 15 40 1	1218
1 2	
2 3	
2 4	
5	
10 5 1 100 1000	
1 2	
2 3	
2 4	
4 5	
4	
1000 200 10 8	
1 2	
2 3	
3 4	

Note

The first sample test case is illustrated below. The answer is $15 + 10 + 10 = 35$.



The second sample test case is illustrated below. The answer is $5 + 10 + 1 + 1 = 17$.



Problem F. Triangles

There is a square on the plane. The coordinates of its lower left corner, upper left corner, lower right corner and upper right corner are $(0, 0)$, $(0, 10^9)$, $(10^9, 0)$ and $(10^9, 10^9)$ respectively.

Given a positive integer k , you need to partition this square into exactly k acute triangles. That is, find k acute triangles where any two triangles do not overlap (however they can share a point or a segment) and the union of all triangles equals the square.

Input

There is only one test case in each test file.

The first only line contains a single integer k ($1 \leq k \leq 50$).

Output

If there doesn't exist a valid partition, output "No" (without quotes).

Otherwise, first output "Yes" (without quotes) in one line. Each of the following k lines contains six numbers x_1, y_1, x_2, y_2, x_3 and y_3 separated by a space, which means that the three points (x_1, y_1) , (x_2, y_2) and (x_3, y_3) form an acute triangle. Note that the k acute triangles must be a partition of the square.

To avoid issues related to precisions, we additionally require that the coordinates of the vertices of all triangles must be integers. It can be proven that for all possible inputs of this problem, if the square can be partitioned into k acute triangles, there always exists a partition satisfying all the constraints.

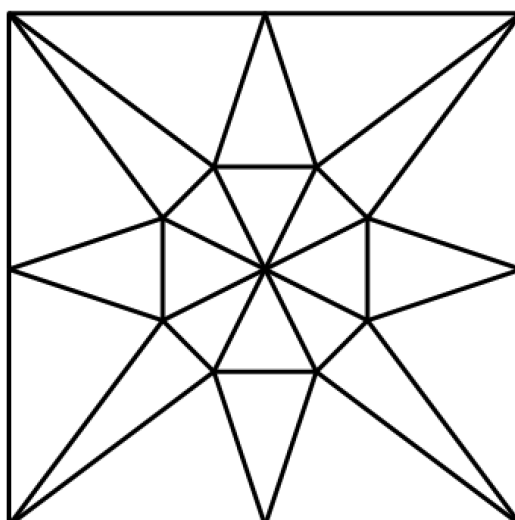
Examples

standard input
2
standard output
No

standard input
24
standard output
<p>Yes</p> <pre> 0 0 500000000 0 400000000 300000000 1000000000 0 500000000 0 600000000 300000000 0 0 0 500000000 300000000 400000000 0 1000000000 0 500000000 300000000 600000000 0 1000000000 500000000 1000000000 400000000 700000000 1000000000 1000000000 500000000 1000000000 600000000 700000000 1000000000 1000000000 1000000000 500000000 700000000 600000000 1000000000 0 1000000000 500000000 700000000 400000000 0 0 400000000 300000000 300000000 400000000 0 500000000 300000000 400000000 300000000 600000000 0 1000000000 300000000 600000000 400000000 700000000 500000000 1000000000 400000000 700000000 600000000 700000000 1000000000 1000000000 600000000 700000000 700000000 600000000 1000000000 500000000 700000000 600000000 700000000 400000000 1000000000 0 700000000 400000000 600000000 300000000 500000000 0 400000000 300000000 600000000 300000000 500000000 500000000 400000000 300000000 300000000 400000000 500000000 500000000 300000000 400000000 300000000 600000000 500000000 500000000 300000000 600000000 400000000 700000000 500000000 500000000 400000000 700000000 600000000 700000000 500000000 500000000 600000000 700000000 700000000 600000000 500000000 500000000 700000000 600000000 700000000 400000000 500000000 500000000 700000000 400000000 600000000 300000000 500000000 500000000 600000000 300000000 400000000 300000000 </pre>

Note

The following figure shows the division given by the second sample case.



Problem G. Inscryption

You are lost deep in the forest. The only thing that is still accompanying you is your stoat. It has an initial attack of 1. It is your only *Beast* at the beginning.

A single path revealed itself before you. On the path are n event marks. Every event mark falls into one of the following:

- **Card Choice:** A dentizen of the forest shall grace your caravan. You will gain an additional Beast. It will always have an initial attack of 1.
- **Mysterious Stone:** You will be compelled to make a worthy sacrifice. Two Beasts from your caravan of your choice will perform the ritual: one to be lost forever, adding its attack onto the other. Failure to perform the ritual will forbid you to go on.
- **Fork in the Road:** You will choose to trigger either a Card Choice or a Mysterious Stone. You can't choose to do nothing.

When you walk through the winding road, the event marks will be triggered in order. Find out the maximum average of attack for your Beasts you can achieve after all event marks are completed.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains one integer n ($1 \leq n \leq 10^6$) indicating the number of event marks.

The second line contains n integers a_1, a_2, \dots, a_n ($-1 \leq a_i \leq 1$) where a_i indicates the type of the i -th event mark: 1 means a Card Choice, -1 means a Mysterious Stone and 0 means a Fork in the Road.

It's guaranteed that the sum of n over all test cases does not exceed 10^6 .

Output

For each test case output one line.

If it is impossible to complete all event marks, output one integer -1 .

Otherwise it can be proven that the answer is a rational number $\frac{p}{q}$. Output two integers p and q where $\frac{p}{q}$ is the simplest fraction representation of $\frac{p'}{q'}$.

$\frac{p}{q}$ is the simplest fraction representation of $\frac{p'}{q'}$ if $\frac{p}{q} = \frac{p'}{q'}$ and the greatest common divisor of p and q is 1.

Example

standard input	standard output
6	3 2
7	3 1
1 1 1 -1 1 1 -1	-1
4	1 1
1 0 -1 0	2 1
4	-1
0 -1 -1 0	
1	
0	
2	
0 0	
1	
-1	

Note

The first sample test case is explained as follows:

Event	Action	Beasts
1	Gain additional Beast	{1, 1}
1	Gain additional Beast	{1, 1, 1}
1	Gain additional Beast	{1, 1, 1, 1}
-1	Choose Beasts with attack 1 and 1	{2, 1, 1}
1	Gain additional Beast	{2, 1, 1, 1}
1	Gain additional Beast	{2, 1, 1, 1, 1}
-1	Choose Beasts with attack 2 and 1	{3, 1, 1, 1}

The average attack is $\frac{3+1+1+1}{4} = \frac{6}{4} = \frac{3}{2}$.

The second sample test case is explained as follows:

Event	Action	Beasts
1	Gain additional Beast	{1, 1}
0	Trigger Card Choice and gain additional Beast	{1, 1, 1}
-1	Choose Beasts with attack 1 and 1	{2, 1}
0	Trigger Mysterious Stone and choose Beasts with attack 2 and 1	{3}

The average attack is $\frac{3}{1}$.

The third sample test case is explained as follows:

Event	Action	Beasts
0	Trigger Card Choice and gain additional Beast	{1, 1}
-1	Choose Beasts with attack 1 and 1	{2}
-1	Not enough Beasts	Failure

Problem H. Factories Once More

There is a kingdom consisting of n cities. The cities are numbered from 1 to n (both inclusive) and there are $(n - 1)$ bidirectional roads connecting them. For each pair of cities, the residents can arrive one from another one through these roads.

The queen has recently decided to construct k new factories. To avoid contamination, she requires that a city can have at most one factory.

You, as the royal designer, are appointed to arrange the construction and meanwhile, maximize the sum of distances between every two factories.

The distance between two factories is the length of the shortest path between the two cities where the two factories belong to. The length of a path is the sum of length of the edges in the path.

Input

There is only one test case in each test file.

The first line contains two integers n and k ($2 \leq n \leq 10^5$, $2 \leq k \leq n$) indicating the number of cities and the number of new factories.

For the following $(n - 1)$ lines, the i -th line contains three integers u_i , v_i and w_i ($1 \leq u_i, v_i \leq n$, $u \neq v$, $1 \leq w \leq 10^4$) indicating a road connecting city u_i and v_i and its length is w_i .

Output

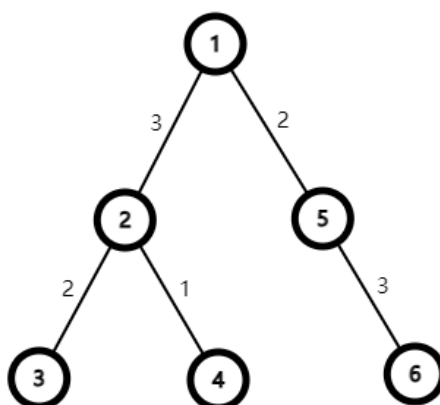
Output one line containing one single integer, indicating the maximum sum of distances between every two factories.

Example

standard input	standard output
6 3 1 2 3 2 3 2 2 4 1 1 5 2 5 6 3	22

Note

The sample test case is shown as follows.



We can choose to build factories in cities 3, 4 and 6. Let $d(i, j)$ be the length of the shortest path between cities i and j , the answer is $d(3, 4) + d(3, 6) + d(4, 6) = 3 + 10 + 9 = 22$.

Problem I. Perfect Palindrome

Given a string $S = s_0s_1 \cdots s_{n-1}$ of length n , let $f(S, d)$ be the string obtained by shifting S to the left d times. That is $f(S, d) = s_{(d+0) \bmod n} s_{(d+1) \bmod n} \cdots s_{(d+n-1) \bmod n}$. We say S is a perfect palindrome if for **all** non-negative integer d , $f(S, d)$ is a palindrome.

You're now given a string $A = a_0a_1 \cdots a_{n-1}$ of length n consisting only of lower-cased English letters. You can perform the following operation on A any number of times (including zero times): Choose an integer i such that $0 \leq i < n$ and change a_i to any lower-cased English letter.

Calculate the minimum number of operations needed to change A into a perfect palindrome.

We say a string $P = p_0p_1 \cdots p_{n-1}$ of length n is a palindrome, if $p_i = p_{n-1-i}$ for all $0 \leq i < n$.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first and only line contains a string $a_0a_1 \cdots a_{n-1}$ ($1 \leq n \leq 10^5$) consisting only of lower-cased English letters.

It is guaranteed that the total length of strings of all test cases will not exceed 10^6 .

Output

For each test case output one line containing one integer indicating the minimum number of operations needed to change A into a perfect palindrome.

Example

standard input	standard output
2	2
abcb	0
xxx	

Note

For the first sample test case, we can change the first and the third characters to 'b' so the string becomes "bbbb". It is easy to see that for all non-negative integer d , $f(\text{"bbbb"}, d) = \text{"bbbb"}$ and "bbbb" is a palindrome, so "bbbb" is a perfect palindrome. These changes cost us 2 operations and it can be proven that this is the minimum number of operations needed.

For the second sample test case, "xxx" is already a perfect palindrome, so no changes are needed.

Problem J. Perfect Matching

Given an undirected graph with n vertices (n is even) and also given n integers a_1, a_2, \dots, a_n , for all positive integers i and j satisfying $1 \leq i < j \leq n$ and $|i - j| = |a_i - a_j|$ ($|x|$ indicates the absolute value of x) we connect vertices i and j with an undirected edge in the graph. It's obvious that this undirected graph does not contain self loops or multiple edges.

Find a perfect matching of this undirected graph, or state that a perfect matching does not exist.

Recall that a perfect matching of a graph is a subset of size $\frac{n}{2}$ of all the edges in the graph, such that each vertex in the graph is connected by one edge in this subset.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an even integer n ($2 \leq n \leq 10^5$) indicating the number of vertices in the undirected graph.

The second line contains n integers a_1, a_2, \dots, a_n ($-10^9 \leq a_i \leq 10^9$).

It's guaranteed that the sum of n of all test cases does not exceed 10^6 .

Output

For each test case, if there does not exist a perfect matching output "No" (without quotes) in one line; If there exists a perfect matching first output "Yes" (without quotes) in one line, then output $\frac{n}{2}$ lines where the i -th line contains two integers u_i and v_i separated by a space indicating the two vertices connected by the i -th edge in the perfect matching. If there are multiple valid answers, output any.

Example

standard input	standard output
3	Yes
6	1 4
14 22 33 11 25 36	5 2
4	6 3
100 10 98 12	Yes
4	1 3
1 3 5 7	4 2
	No

Problem K. NaN in a Heap

Prerequisite: NaN

NaN (Not a Number) is a special floating-point value introduced by the IEEE 754 floating-point standard in 1985. The standard specifies that, when NaN is compared with a floating-point value x (x can be positive, zero, negative, or even NaN itself), the following results should be returned.

Comparison	NaN $\geq x$	NaN $\leq x$	NaN $> x$	NaN $< x$	NaN $= x$	NaN $\neq x$
Result	False	False	False	False	False	True

Prerequisite: Heap

A heap is a data structure which can be represented by a sequence with special properties. The following algorithm demonstrates how to insert n floating-point values a_1, a_2, \dots, a_n into a min-heap H in order, where H is a sequence and is initially empty.

In the following algorithm, let h_i be the i -th element in sequence H , and let $j/2$ be the maximum integer x satisfying $2x \leq j$.

Algorithm 1 Heapify

```

1: function HEAPIFY( $A$ )
2:   Let  $H$  be an empty sequence.
3:   for  $i \leftarrow 1$  to  $n$  do            $\triangleright n$  is the number of elements to be inserted into heap.
4:     Append  $a_i$  to the back of  $H$ .
5:      $j := i$ 
6:     while  $j > 1$  do
7:       if  $h_j < h_{j/2}$  then            $\triangleright$  Recall that if  $h_j$  or  $h_{j/2}$  is NaN, this expression will be false.
8:         Swap  $h_j$  and  $h_{j/2}$ .
9:          $j := j/2$ 
10:      else
11:        break
12:      end if
13:    end while
14:  end for
15:  return  $H$ 
16: end function

```

Problem

Given an integer n , consider permutations of these n elements: all integers from 1 to $(n - 1)$ (both inclusive), as well as a NaN value. We say a permutation P of these n elements is a “heap sequence”, if there exists a permutation Q also of these n elements satisfying $P = \text{HEAPIFY}(Q)$.

We now randomly pick a permutation of these n elements with equal probability (that is, the probability of a specific permutation to be picked is $\frac{1}{n!}$), calculate the probability that the picked permutation is a heap sequence.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 10^3$) indicating the number of test cases. For each test case:

The first and only line contains an integer n ($1 \leq n \leq 10^9$).

Output

For each test case output one line containing the answer.

It can be proven that the answer is a rational number $\frac{p}{q}$. To avoid issues related to precisions, please output the integer $(pq^{-1} \bmod M)$ as the answer, where $M = 10^9 + 7$ and q^{-1} is the integer satisfying $qq^{-1} \equiv 1 \pmod{M}$.

Example

standard input	standard output
5	1
1	666666672
3	55555556
7	596445110
10	3197361
20221218	

Note

For the second sample test case, there are 4 heap sequences.

- $\{\text{NaN}, 1, 2\} = \text{HEAPIFY}(\{\text{NaN}, 1, 2\})$.
- $\{\text{NaN}, 2, 1\} = \text{HEAPIFY}(\{\text{NaN}, 2, 1\})$.
- $\{1, \text{NaN}, 2\} = \text{HEAPIFY}(\{1, \text{NaN}, 2\}) = \text{HEAPIFY}(\{2, \text{NaN}, 1\})$.
- $\{1, 2, \text{NaN}\} = \text{HEAPIFY}(\{1, 2, \text{NaN}\}) = \text{HEAPIFY}(\{2, 1, \text{NaN}\})$.

So the answer is $\frac{4}{3!} = \frac{2}{3}$ in rational number. As $3 \times 333333336 \equiv 1 \pmod{M}$, we should output $2 \times 333333336 \bmod M = 666666672$.

Problem L. Proposition Composition

These is an undirected connected graph G with n vertices and $(n - 1)$ edges. The vertices are numbered from 1 to n (both inclusive) and the i -th edge connects vertices i and $(i + 1)$.

There will be m extra edges added to this graph. Each addition is permanent. After each edge is added, output the number of ways to choose two edges e and f from the graph such that if both edges e and f are removed from the graph, the graph will become disconnected (that is, the graph will have at least two connected components).

Note that first choosing e then choosing f is considered the same way as first choosing f then choosing e .

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \leq n, m \leq 2.5 \times 10^5$) indicating the size of the graph and the number of extra edges.

For the following m lines, the i -th line contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$) indicating the i -th extra edge connects vertices u_i and v_i .

It's guaranteed that neither the sum of n nor the sum of m over all test cases will exceed 2.5×10^5 .

Output

For each test case output m lines where the i -th line contains the answer after the i -th extra edge is added.

Example

standard input	standard output
3	6
4 3	5
2 4	6
4 2	21
3 3	24
7 3	10
3 4	15
1 2	12
1 7	3
6 4	2
1 3	
4 6	
2 5	
3 4	

Note

We explain the first sample test case as follows.

After adding the first extra edge, removing any two edges will make the graph disconnected. So the answer is 6.

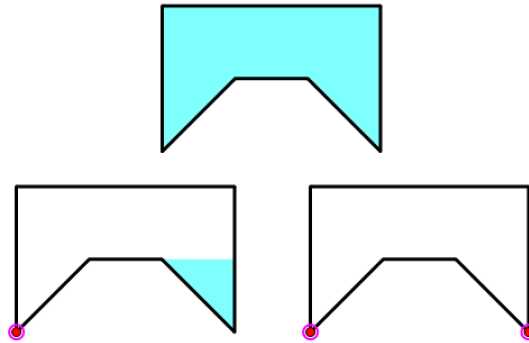
After adding the second extra edge, we can choose original edge 1 and any other edge, or choose original edge 2 and original edge 3. The answer is $4 + 1 = 5$.

After adding the third extra edge, we can choose original edge 1 and any other edge, or choose original edge 2 and original edge 3. The answer is $5 + 1 = 6$.

Problem M. Drain the Water Tank

The water plant has made a new water tank with a polygon shape and negligible thickness recently.

To bring the water tank into service, the engineers are going to install some drain valves on the tank. A drain valve is considered a point on the water tank, and the water will flow out of the tank through the drain valve when it is on.



As illustrated above, valves are represented by purple dots and the light blue areas are the water remaining in the tank after all valves are turned on.

You, as the chief engineer, are willing to know the minimum number of drain valves needed so that all the water in the tank can be drained off when turning on all the drain valves simultaneously.

You may assume that the water is an ideal fluid and no atmospheric pressure here, so the water always tends to flow to somewhere strictly lower, even when the water is on a horizontal stage.

Input

There is only one test case in each test file.

The first line of the input contains an integer n ($3 \leq n \leq 2 \times 10^3$) indicating the number of vertices of the polygon (that is, the shape of the water tank).

For the following n lines, the i -th line contains two integers x_i and y_i ($|x_i|, |y_i| \leq 10^4$) indicating the coordinate of the i -th vertex of the polygon. Vertices are given in counter-clockwise order.

The polygon is simple. That is, its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex. It is **NOT** guaranteed that no two consecutive edges are collinear.

Output

Output one line containing one integer indicating the minimum number of drain valves that suffices to drain the water tank.

Examples

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

Note

In the first sample test case, two drain valves installed at $(0, 0)$ and $(3, 0)$ are sufficient to drain the water tank.

In the second sample test case, a single drain valve installed at $(3, 0)$ is sufficient to drain the water tank. Actually, it is sufficient to install the valve at $(x, 0)$ where $2 \leq x \leq 4$.

In the third sample test case, two drain valves installed at $(1, 0)$ and $(1, 2)$ are sufficient to drain the water tank.

