A. Swap Columns and Find a Path

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

There is a matrix consisting of 2 rows and n columns. The rows are numbered from 1 to 2 from top to bottom; the columns are numbered from 1 to n from left to right. Let's denote the cell on the intersection of the i-th row and the j-th column as (i, j). Each cell contains an integer; initially, the integer in the cell (i, j) is $a_{i,j}$.

You can perform the following operation any number of times (possibly zero):

- choose two columns and swap them (i. e. choose two integers x and y such that $1 \le x < y \le n$, then swap $a_{1,x}$ with $a_{1,y}$, and then swap $a_{2,x}$ with $a_{2,y}$).

After performing the operations, you have to choose a path from the cell $(1, 1)$ to the cell $(2, n)$. For every cell (i, j) in the path except for the last, the next cell should be either $(i + 1, j)$ or $(i, j + 1)$. Obviously, the path cannot go outside the matrix.

The *cost* of the path is the sum of all integers in all $(n + 1)$ cells belonging to the path. You have to perform the operations and choose a path so that its cost is **maximum** possible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

Each test case consists of three lines:

- the first line contains one integer n ($1 \le n \le 5000$) — the number of columns in the matrix;
- the second line contains n integers $a_{1,1}, a_{1,2}, \ldots, a_{1,n}$ ($-10^5 \le a_{i,j} \le 10^5$) — the first row of the matrix;
- the third line contains n integers $a_{2,1}, a_{2,2}, \ldots, a_{2,n}$ ($-10^5 \le a_{i,j} \le 10^5$) — the second row of the matrix.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

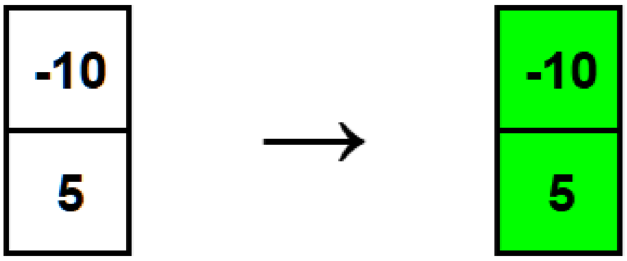
For each test case, print one integer — the maximum cost of a path you can obtain.

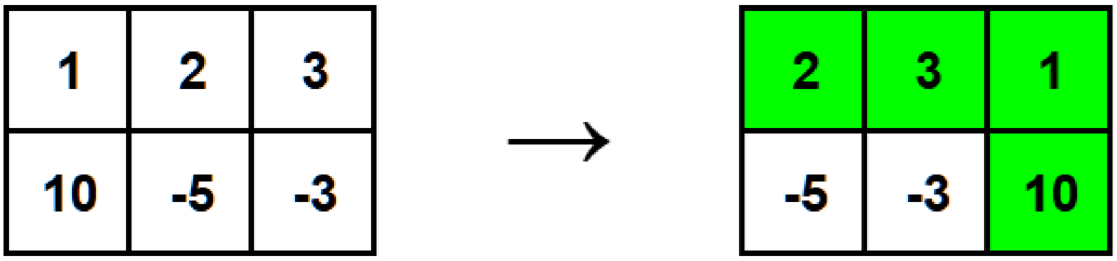
Standard Input	Standard Output

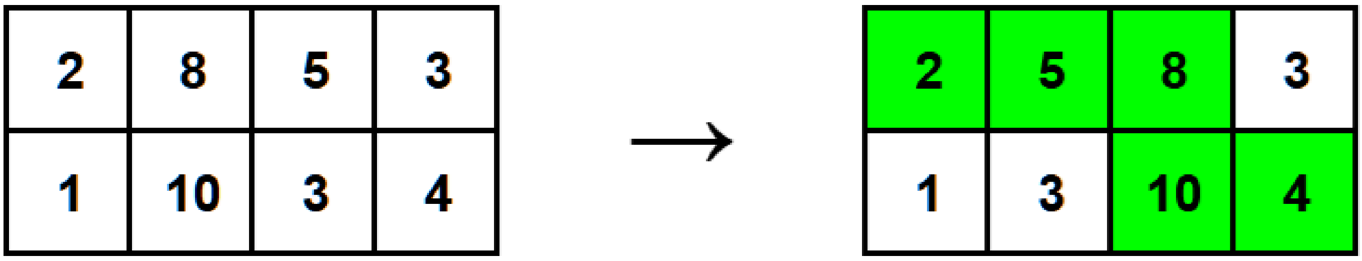
3
1
-10
5
3
1 2 3
10 -5 -3
4
2 8 5 3
1 10 3 4
	-5
	16
	29

Note

Here are the explanations of the first three test cases of the example. The left matrix is the matrix given in the input, the right one is the state of the matrix after several column swaps (possibly zero). The optimal path is highlighted in green.







,

B. Move Back at a Cost

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given an array of integers a of length n. You can perform the following operation zero or more times:

- In one operation choose an index i ($1 \le i \le n$), assign $a_i := a_i + 1$, and then move a_i to the back of the array (to the rightmost position). For example, if $a = [3, 5, 1, 9]$, and you choose $i = 2$, the array becomes $[3, 1, 9, 6]$.

Find the lexicographically smallest* array you can get by performing these operations.

*An array c is lexicographically smaller than an array d if and only if one of the following holds:

- c is a prefix of d, but $c \ne d$; or
- in the first position where c and d differ, the array c has a smaller element than the corresponding element in d.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line contains a single integer n ($1 \le n \le 10^5$), the length of the array.

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

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

Output

For each test case, print the lexicographically smallest array you can get.

Standard Input	Standard Output

3
3
2 1 3
5
1 2 2 1 4
6
1 2 3 6 5 4
	1 3 3
	1 1 3 3 5
	1 2 3 4 6 7

,

C. Adventurers

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

Once, four Roman merchants met in a Roman mansion to discuss their trading plans. They faced the following problem: they traded the same type of goods, and if they traded in the same city, they would inevitably incur losses. They decided to divide up the cities between them where they would trade.

The map of Rome can be represented in this problem as a plane with certain points marked — the cities of the Roman Empire.

The merchants decided to choose a certain *dividing point* (x_0, y_0). Then, in a city with coordinates (x_i, y_i),

- the first merchant sells goods if $x_0 \le x_i$ and $y_0 \le y_i$;
- the second merchant sells goods if $x_0 > x_i$ and $y_0 \le y_i$;
- the third merchant sells goods if $x_0 \le x_i$ and $y_0 > y_i$;
- the fourth merchant sells goods if $x_0 > x_i$ and $y_0 > y_i$.

The merchants want to choose (x_0, y_0) in such a way as to **maximize the smallest number of cities** that any of them gets (i. e., as fair as possible). Please find such a point for them.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($4 \le n \le 10^5$) — the number of cities on the map.

Each of the next n lines contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the cities.

Note that some points may coincide. This is because some cities may be so close that they cannot be distinguished on the map at the given scale.

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

Output

For each test case, in the first line, print a single integer k ($0 \le k \le \frac{n}{4}$) — the maximum possible number of cities that each merchant can get at a minimum.

In the second line, print two integers x_0 and y_0 ($|x_0|, |y_0| \le 10^9$) — the coordinates of the dividing point. If there are multiple suitable points, print any of them.

Standard Input	Standard Output

4
4
1 1
1 2
2 1
2 2
4
	1
	2 2
	0
	0 0
	2
	1 0

0 0
0 0
0 0
0 0
8
1 2
2 1
2 -1
1 -2
-1 -2
-2 -1
-2 1
-1 2
7
1 1
1 2
1 3
1 4
2 1
3 1
4 1
	0
	0 0

,

D. For the Emperor!

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

In Ancient Rome, a plan to defeat the barbarians was developed, but for its implementation, each city must be informed about it.

The northern part of the Roman Empire consists of n cities connected by m one-way roads. Initially, the i-th city has a_i messengers, and each messenger can freely move between cities following the existing roads. A messenger can carry a copy of the plan with him and inform the cities he visits, and can make unlimited copies for other messengers in the city he is currently in.

At the start, you will produce some number of plans and deliver them to messengers of your choice. Your goal is to make sure that every city is visited by a messenger with a plan. Find the smallest number of the plans you need to produce originally, so that the messengers will deliver them to every city, or determine that it is impossible to do so at all.

Input

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

The first line contains two integers n and m ($2 \le n \le 200$, $1 \le m \le 800$) — the number of cities and roads.

The second line contains n non-negative integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n$) — the initial number of messengers in each city.

Each of the following m lines contains two integers u and v ($1 \le u, v \le n, u \ne v$), indicating that there is a one-way road from city u to city v. The roads may repeat.

It is guaranteed that the sum of n over all test cases does not exceed 200. It is guaranteed that the sum of m over all test cases does not exceed 800.

Output

Output a single line containing a single integer — the smallest number of messengers you need to give a copy of the plan in the beginning, or -1 if it is not possible to inform all cities.

Standard Input	Standard Output

2
7 6
2 1 0 1 2 3 4
1 2
1 3
2 4
2 5
3 6
3 7
4 4
1 1 1 1
	2
	2

1 2
1 3
2 4
3 4	

,

E1. Cheops and a Contest (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

This is the easy version of the problem. The difference between the versions is that in this version, m equals 2. You can hack only if you solved all versions of this problem.

There is a problem-solving competition in Ancient Egypt with n participants, numbered from 1 to n. Each participant comes from a certain city; the cities are numbered from 1 to m. There is at least one participant from each city.

The i-th participant has strength a_i, specialization s_i, and wisdom b_i, so that $b_i \geq a_i$. Each problem in the competition will have a difficulty d and a unique topic t. The i-th participant will solve the problem if

- $a_i \geq d$, i.e., their strength is not less than the problem's difficulty, or
- $s_i = t$, and $b_i \geq d$, i.e., their specialization matches the problem's topic, and their wisdom is not less than the problem's difficulty.

Cheops wants to choose the problems in such a way that each participant from city i will solve strictly more problems than each participant from city j, for all $i < j$.

Please find a set of at most $5n$ problems, where the **topics of all problems are distinct**, so that Cheops' will is satisfied, or state that it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases T ($1 \leq T \leq 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n, m ($2{=}m \leq n \leq 3 \cdot 10^5$) — the number of participants and the number of cities.

The following n lines describe the participants. The i-th line contains three integers —a_i, b_i, s_i ($0 \leq a_i, b_i, s_i \leq 10^9$, $a_i \leq b_i$) — strength, wisdom, and specialization of the i-th participant, respectively.

The next m lines describe the cities. In the i-th line, the first number is an integer k_i ($1 \leq k_i \leq n$) — the number of participants from the i-th city. It is followed by k_i integers $q_{i,1}, q_{i,2}, \ldots, q_{i,k_i}$ — ($1 \leq q_{i,j} \leq n$, $1 \leq j \leq k_i$) — the indices of the participants from this city. It is guaranteed that each participant is mentioned exactly once.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, if there exists a set of problems that satisfies Cheops' conditions, then in the first line output a single integer p ($1 \leq p \leq 5n$) — the number of problems in your solution.

Then output p lines, each containing two integers d and t ($0 \leq d, t \leq 10^9$) — the difficulty and topic of the respective problem. The topics must be distinct.

If there is no set of problems that meets Cheops' wishes, print -1 instead.

Standard Input	Standard Output

2
5 2
5 7 1
6 7 2
3 9 2
5 10 3
4 4 1
2 1 2
3 3 4 5
2 2
1 2 1
1 2 1
1 2
1 1
	7
	6 4
	6 5
	5 6
	5 7
	4 8
	4 9
	7 1
	-1

,

E2. Cheops and a Contest (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version, m is arbitrary. You can hack only if you solved all versions of this problem.

There is a problem-solving competition in Ancient Egypt with n participants, numbered from 1 to n. Each participant comes from a certain city; the cities are numbered from 1 to m. There is at least one participant from each city.

The i-th participant has strength a_i, specialization s_i, and wisdom b_i, so that $b_i \geq a_i$. Each problem in the competition will have a difficulty d and a unique topic t. The i-th participant will solve the problem if

- $a_i \geq d$, i.e., their strength is not less than the problem's difficulty, or
- $s_i = t$, and $b_i \geq d$, i.e., their specialization matches the problem's topic, and their wisdom is not less than the problem's difficulty.

Cheops wants to choose the problems in such a way that each participant from city i will solve strictly more problems than each participant from city j, for all $i < j$.

Please find a set of at most $5n$ problems, where the **topics of all problems are distinct**, so that Cheops' will is satisfied, or state that it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases T ($1 \leq T \leq 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n, m ($2 \leq m \leq n \leq 3 \cdot 10^5$) — the number of participants and the number of cities.

The following n lines describe the participants. The i-th line contains three integers —a_i, b_i, s_i ($0 \leq a_i, b_i, s_i \leq 10^9$, $a_i \leq b_i$) — strength, wisdom, and specialization of the i-th participant, respectively.

The next m lines describe the cities. In the i-th line, the first number is an integer k_i ($1 \leq k_i \leq n$) — the number of participants from the i-th city. It is followed by k_i integers $q_{i,1}, q_{i,2}, \ldots, q_{i,k_i}$ — ($1 \leq q_{i,j} \leq n$, $1 \leq j \leq k_i$) — the indices of the participants from this city. It is guaranteed that each participant is mentioned exactly once.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, if there exists a set of problems that satisfies Cheops' conditions, then in the first line output a single integer p ($1 \leq p \leq 5n$) — the number of problems in your solution.

Then output p lines, each containing two integers d and t ($0 \leq d, t \leq 10^9$) — the difficulty and topic of the respective problem. The topics must be distinct.

If there is no set of problems that meets Cheops' wishes, print -1 instead.

Standard Input	Standard Output
2	7
5 2	6 4
5 7 1	6 5
6 7 2	5 6
3 9 2	5 7
5 10 3	4 8
4 4 1	4 9
2 1 2	7 1
3 3 4 5	-1
2 2	
1 2 1	
1 2 1	
1 2	
1 1	

,

F1. Yandex Cuneiform (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the easy version of the problem. The difference between the versions is that in this version, there are no question marks. You can hack only if you solved all versions of this problem.

For a long time, no one could decipher Sumerian cuneiform. However, it has finally succumbed to pressure! Today, you have the chance to decipher Yandex cuneiform.

Yandex cuneiform is defined by the following rules:

1. An empty string is a Yandex cuneiform.
2. If you insert exactly one copy of each of the three letters 'Y', 'D', and 'X' into a Yandex cuneiform in such a way that no two adjacent letters become equal after the operation, you obtain a Yandex cuneiform.
3. If a string can't be obtained using the above rules, it is not a Yandex cuneiform.

You are given a template. A template is a string consisting of the characters 'Y', 'D', 'X', and '?'.

You need to check whether there exists a way to replace each question mark with 'Y', 'D', or 'X' to obtain a Yandex cuneiform, and if it exists, output any of the matching options, as well as a sequence of insertion operations to obtain the resulting cuneiform.

In this version of the problem, there are **no question marks** in the template.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^4$). The description of the test cases follows.

Each test case consists of a single line containing a template of length n ($3 \le n < 2 \cdot 10^5$, $n \bmod 3 = 0$), consisting only of characters 'Y', 'D', 'X'.

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 a single line containing 'NO' if it is not possible to obtain a cuneiform from the given template.

Otherwise, output 'YES' on the first line, and on the second line, any obtainable cuneiform. After that, you need to output the sequence of operations that leads to the cuneiform you printed.

A sequence of operations is described by $\frac{n}{3}$ triples of pairs. A pair has the form `c p`, where c is one of the letters 'Y', 'D', or 'X', and p is the position at which the letter c should be inserted. The insertion position is the number of letters to skip from the beginning of the string for the insertion. For example, after inserting the character 'D' into the string "YDX" with $p = 3$, the result is "YDXD", and with $p = 0$, it is "DYDX". Note that the index cannot exceed the current length of the string.

The operations are applied from top to bottom, left to right. After inserting each triple to the string, there should be no two adjacent identical characters.

Standard Input	Standard Output
`4`	`YES`
`YDX`	`YDX`
`YDXDYX`	`X 0 D 0 Y 0`
`YDX`	`YES`
`DYYDXYXYX`	`YDXDYX`
	`X 0 Y 0 D 1`
	`X 2 D 3 Y 4`
	`YES`
	`YDX`
	`Y 0 D 1 X 2`
	`NO`

Note

In the second example, the string is transformed like this: ""$\to$ `YDX` $\to$ `YDXDYX`.

,

F2. Yandex Cuneiform (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version, there is no restriction on the number of question marks. You can hack only if you solved all versions of this problem.

For a long time, no one could decipher Sumerian cuneiform. However, it has finally succumbed to pressure! Today, you have the chance to decipher Yandex cuneiform.

Yandex cuneiform is defined by the following rules:

1. An empty string is a Yandex cuneiform.
2. If you insert exactly one copy of each of the three letters 'Y', 'D', and 'X' into a Yandex cuneiform in such a way that no two adjacent letters become equal after the operation, you obtain a Yandex cuneiform.
3. If a string can't be obtained using the above rules, it is not a Yandex cuneiform.

You are given a template. A template is a string consisting of the characters 'Y', 'D', 'X', and '?'.

You need to check whether there exists a way to replace each question mark with 'Y', 'D', or 'X' to obtain a Yandex cuneiform, and if it exists, output any of the matching options, as well as a sequence of insertion operations to obtain the resulting cuneiform.

In this version of the problem, the number of question marks in the template can be arbitrary.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^4$). The description of the test cases follows.

Each test case consists of a single line containing a template of length n ($3 \le n < 2 \cdot 10^5$, $n \bmod 3 = 0$), consisting only of characters 'Y', 'D', 'X', and '?'.

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 a single line containing 'NO' if it is not possible to obtain a cuneiform from the given template.

Otherwise, output 'YES' on the first line, and on the second line, any obtainable cuneiform. After that, you need to output the sequence of operations that leads to the cuneiform you printed.

A sequence of operations is described by $\frac{n}{3}$ triples of pairs. A pair has the form `c p`, where c is one of the letters 'Y', 'D', or 'X', and p is the position at which the letter c should be inserted. The insertion position is the number of letters to skip from the beginning of the string for the insertion. For example, after inserting the character 'D' into the string "YDX" with $p = 3$, the result is "YDXD", and with $p = 0$, it is "DYDX". Note that the index cannot exceed the current length of the string.

The operations are applied from top to bottom, left to right. After inserting each triple to the string, there should be no two adjacent identical characters.

Standard Input	Standard Output
4 ??? Y??D?X ??? D??DXYXYX	YES YDX X 0 D 0 Y 0 YES YDXDYX X 0 Y 0 D 1 X 2 D 3 Y 4 YES YDX Y 0 D 1 X 2 NO

Note

In the second example, the string is transformed like this: ””$\to$ `YDX` $\to$ `YDXDYX`.