Codeforces Round #756 (Div. 3)

A. Make Even

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

Polycarp has an integer n that doesn't contain the digit `0`. He can do the following operation with his number several (possibly zero) times:

- Reverse the prefix of length l (in other words, l leftmost digits) of n. So, the leftmost digit is swapped with the l-th digit from the left, the second digit from the left swapped with ($l-1$)-th left, etc. For example, if $n = 123456789$ and $l = 5$, then the new value of n will be 543216789.

Note that for different operations, the values of l can be different. The number l can be equal to the length of the number n — in this case, the whole number n is reversed.

Polycarp loves even numbers. Therefore, he wants to make his number even. At the same time, Polycarp is very impatient. He wants to do as few operations as possible.

Help Polycarp. Determine the minimum number of operations he needs to perform with the number n to make it even or determine that this is impossible.

You need to answer t independent test cases.

Input

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

Each of the following t lines contains one integer n ($1 \le n < 10^9$). It is guaranteed that the given number doesn't contain the digit `0`.

Output

Print t lines. On each line print one integer — the answer to the corresponding test case. If it is impossible to make an even number, print `-1`.

Example

input

```
4
3876
387
4489
3
```

output

```
0
2
1
-1
```

Note

In the first test case, $n = 3876$, which is already an even number. Polycarp doesn't need to do anything, so the answer is 0.

In the second test case, $n = 387$. Polycarp needs to do 2 operations:

1. Select $l = 2$ and reverse the prefix $\underline{38}7$. The number n becomes 837. This number is odd.
2. Select $l = 3$ and reverse the prefix $\underline{837}$. The number n becomes 738. This number is even.

It can be shown that 2 is the minimum possible number of operations that Polycarp needs to do with his number to make it even.

In the third test case, $n = 4489$. Polycarp can reverse the whole number (choose a prefix of length $l = 4$). It will become 9844 and this is an even number.

In the fourth test case, $n = 3$. No matter how hard Polycarp tried, he would not be able to make an even number.

B. Team Composition: Programmers and Mathematicians

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

The All-Berland Team Programming Contest will take place very soon. This year, teams of four are allowed to participate.

There are a programmers and b mathematicians at Berland State University. How many maximum teams can be made if:

- each team must consist of exactly 4 students,
- teams of 4 mathematicians or 4 programmers are unlikely to perform well, so the decision was made not to compose such teams.

Thus, each team must have at least one programmer **and** at least one mathematician.

Print the required maximum number of teams. Each person can be a member of no more than one team.

Input

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

This is followed by descriptions of t sets, one per line. Each set is given by two integers a and b ($0 \le a, b \le 10^9$).

Output

Print t lines. Each line must contain the answer to the corresponding set of input data — the required maximum number of teams.

Example

input

```
6
5 5
10 1
2 3
0 0
17 2
1000000000 1000000000
```

output

```
2
1
1
0
2
500000000
```

Note

In the first test case of the example, two teams can be composed. One way to compose two teams is to compose two teams of 2 programmers and 2 mathematicians.

In the second test case of the example, only one team can be composed: 3 programmers and 1 mathematician in the team.

C. Polycarp Recovers the Permutation

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

Polycarp wrote on a whiteboard an array p of length n, which is a permutation of numbers from 1 to n. In other words, in p each number from 1 to n occurs exactly once.

He also prepared a resulting array a, which is initially empty (that is, it has a length of 0).

After that, he did exactly n *steps*. Each *step* looked like this:

- Look at the leftmost and rightmost elements of p, and pick the smaller of the two.
- If you picked the leftmost element of p, append it to the left of a; otherwise, if you picked the rightmost element of p, append it to the right of a.
- The picked element is erased from p.

Note that on the last step, p has a length of 1 and its minimum element is both leftmost and rightmost. In this case, Polycarp can choose what role the minimum element plays. In other words, this element can be added to a both on the left and on the right (at the discretion of Polycarp).

Let's look at an example. Let $n = 4$, $p = [3, 1, 4, 2]$. Initially $a = []$. Then:

- During the first step, the minimum is on the right (with a value of 2), so after this step, $p = [3, 1, 4]$ and $a = [2]$ (he added the value 2 to the right).
- During the second step, the minimum is on the left (with a value of 3), so after this step, $p = [1, 4]$ and $a = [3, 2]$ (he added the value 3 to the left).
- During the third step, the minimum is on the left (with a value of 1), so after this step, $p = [4]$ and $a = [1, 3, 2]$ (he added the value 1 to the left).
- During the fourth step, the minimum is both left and right (this value is 4). Let's say Polycarp chose the right option. After this step, $p = []$ and $a = [1, 3, 2, 4]$ (he added the value 4 to the right).

Thus, a possible value of a after n steps could be $a = [1, 3, 2, 4]$.

You are given the final value of the resulting array a. Find **any** possible initial value for p that can result the given a, or determine that there is no solution.

Input

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

Each test case consists of two lines. The first of them contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a. The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$) — the elements of the array a. All elements of the a array are distinct numbers.

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

Output

Print t lines, each of the lines must contain the answer to the corresponding set of input data: numbers $p_1, p_2, \dots, p_n$ — any of the possible initial values of the array p, which will lead to the given array a. All elements of p are distinct integers from 1 to n. Thus, if there are several solutions, print any. If there is no solution, then print -1 on the line.

Example

input

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

output

```
3 1 4 2
1
-1
2 3 1
```

Note

The first test case in the example is clarified in the main section of the problem statement. There may be other correct answers for this test set.

In the second test case, $n = 1$. Thus, there is only one permutation that can be the answer: $p = [1]$. Indeed, this is the answer to this test case.

In the third test case of the example, no matter what permutation you take as p, after applying the n steps, the result will differ from $a = [1, 3, 5, 4, 2]$.

D. Weights Assignment For Tree Edges

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

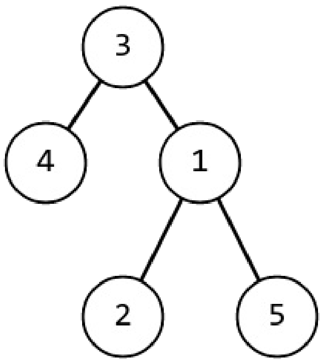
You are given a rooted tree consisting of n vertices. Vertices are numbered from 1 to n. Any vertex can be the root of a tree.

A *tree* is a connected undirected graph without cycles. A *rooted tree* is a tree with a selected vertex, which is called the *root*.

The tree is specified by an array of ancestors b containing n numbers: b_i is an ancestor of the vertex with the number i. The *ancestor* of a vertex u is a vertex that is the next vertex on a simple path from u to the root. For example, on the simple path from 5 to 3 (the root), the next vertex would be 1, so the ancestor of 5 is 1.

The root has no ancestor, so for it, the value of b_i is i (the root is the only vertex for which $b_i = i$).

For example, if $n = 5$ and $b = [3, 1, 3, 3, 1]$, then the tree looks like this.



An example of a rooted tree for $n = 5$, the root of the tree is a vertex number 3.

You are given an array p — a permutation of the vertices of the tree. If it is possible, assign any **positive** integer weights on the edges, so that the vertices sorted by distance from the root would form the given permutation p.

In other words, for a given permutation of vertices p, it is necessary to choose such edge weights so that the condition $dist[p_i] < dist[p_{i+1}]$ is true for each i from 1 to $n-1$. $dist[u]$ is a sum of the weights of the edges on the path from the root to u. In particular, $dist[u] = 0$ if the vertex u is the root of the tree.

For example, assume that $p = [3, 1, 2, 5, 4]$. In this case, the following edge weights satisfy this permutation:

- the edge $(3, 4)$ has a weight of 102;
- the edge $(3, 1)$ has weight of 1;
- the edge $(1, 2)$ has a weight of 10;
- the edge $(1, 5)$ has a weight of 100.

The array of distances from the root looks like: $dist = [1, 11, 0, 102, 101]$. The vertices sorted by increasing the distance from the root form the given permutation p.

Print the required edge weights or determine that there is no suitable way to assign weights. If there are several solutions, then print any of them.

Input

The first line of input data contains an integer t ($1 \le t \le 10^4$) — the number of input data sets in the test.

Each test case consists of three lines.

The first of them contains an integer n ($1 \le n \le 2 \cdot 10^5$). It is the number of vertices in the tree.

The second line contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le n$). It is guaranteed that the b array encodes some rooted tree.

The third line contains the given permutation p: n of different integers $p_1, p_2, \dots, p_n$ ($1 \le p_i \le n$).

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

Output

For each set of input data print the answer on a separate line.

If the solution exists, print an array of n integers $w_1, w_2, \dots, w_n$, where w_i is the weight of the edge that leads from b_i to i. For the root there is no such edge, so use the value $w_i = 0$. For all other vertices, the values of w_i must satisfy the inequality $1 \le w_i \le 10^9$. There can be equal numbers among w_i values, but all sums of weights of edges from the root to vertices must be different and satisfy the given permutation.

If there are several solutions, output any of them.

If no solution exists, output -1.

Example

input

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

output

```
1 10 0 102 100
-1
0 3 100 1 1 2 4
```

```
6 5 10 0 2 3
```

Note

The first set of input data of the example is analyzed in the main part of the statement.

In the second set of input data of the example, it is impossible to assign the positive weights to obtain a given permutation of vertices.

E1. Escape The Maze (easy version)

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

The only difference with E2 is the question of the problem..

Vlad built a maze out of n rooms and $n-1$ bidirectional corridors. From any room u any other room v can be reached through a sequence of corridors. Thus, the room system forms an undirected tree.

Vlad invited k friends to play a game with them.

Vlad starts the game in the room 1 and wins if he reaches a room other than 1, into which exactly one corridor leads.

Friends are placed in the maze: the friend with number i is in the room x_i, and no two friends are in the same room (that is, $x_i \neq x_j$ for all $i \neq j$). Friends win if one of them meets Vlad in any room or corridor before he wins.

For one unit of time, each participant of the game can go through one corridor. All participants move at the same time. Participants may not move. Each room can fit all participants at the same time.

Friends know the plan of a maze and intend to win. Vlad is a bit afraid of their ardor. Determine if he can guarantee victory (i.e. can he win in any way friends play).

In other words, determine if there is such a sequence of Vlad's moves that lets Vlad win in any way friends play.

Input

The first line of the input contains an integer t ($1 \leq t \leq 10^4$) — the number of test cases in the input. The input contains an empty string before each test case.

The first line of the test case contains two numbers n and k ($1 \leq k < n \leq 2 \cdot 10^5$) — the number of rooms and friends, respectively.

The next line of the test case contains k integers $x_1, x_2, \ldots, x_k$ ($2 \leq x_i \leq n$) — numbers of rooms with friends. All x_i are different.

The next $n-1$ lines contain descriptions of the corridors, two numbers per line v_j and u_j ($1 \leq u_j, v_j \leq n$) — numbers of rooms that connect the j corridor. All corridors are bidirectional. From any room, you can go to any other by moving along the corridors.

It is guaranteed that the sum of the values n over all test cases in the test is not greater than $2 \cdot 10^5$.

Output

Print t lines, each line containing the answer to the corresponding test case. The answer to a test case should be "YES" if Vlad can guarantee himself a victory and "NO" otherwise.

You may print every letter in any case you want (so, for example, the strings "yEs", "yes", "Yes" and "YES" will all be recognized as positive answers).

Example

input

```
4

8 2
5 3
4 7
2 5
1 6
3 6
7 2
1 7
6 8

3 1
2
1 2
2 3

3 1
2
1 2
1 3

3 2
2 3
```

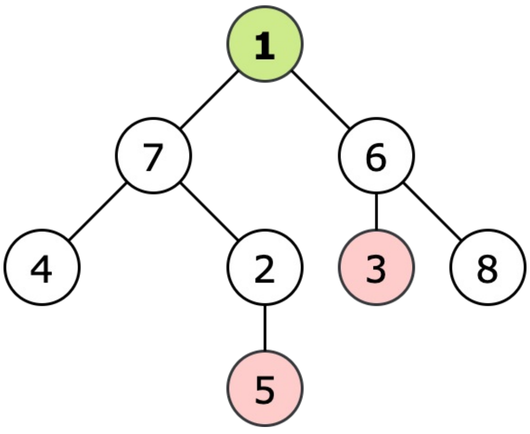
```
3 1
1 2
```

output

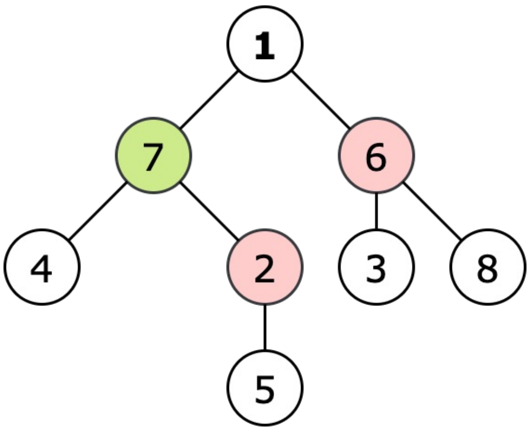
```
YES
NO
YES
NO
```

Note

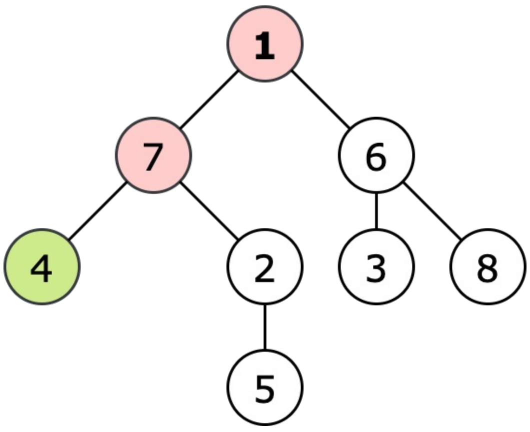
In the first test case, regardless of the strategy of his friends, Vlad can win by going to room 4. The game may look like this:



The original locations of Vlad and friends. Vlad is marked in green, friends — in red.



Locations after one unit of time.



End of the game.

Note that if Vlad tries to reach the exit at the room 8, then a friend from the 3 room will be able to catch him.

E2. Escape The Maze (hard version)

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

The only difference with E1 is the question of the problem.

Vlad built a maze out of n rooms and $n-1$ bidirectional corridors. From any room u any other room v can be reached through a sequence of corridors. Thus, the room system forms an undirected tree.

Vlad invited k friends to play a game with them.

Vlad starts the game in the room 1 and wins if he reaches a room other than 1, into which exactly one corridor leads. Friends are placed in the maze: the friend with number i is in the room x_i, and no two friends are in the same room (that is, $x_i \neq x_j$ for all $i \neq j$). Friends win if one of them meets Vlad in any room or corridor before he wins.

For one unit of time, each participant of the game can go through one corridor. All participants move at the same time. Participants may not move. Each room can fit all participants at the same time.

Friends know the plan of a maze and intend to win. They don't want to waste too much energy. They ask you to determine if they can win and if they can, what **minimum** number of friends must remain in the maze so that they can always catch Vlad.

In other words, you need to determine the size of the minimum (by the number of elements) subset of friends who can catch Vlad or say that such a subset does not exist.

Input

The first line of the input contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. The input contains an empty string before each test case.

The first line of the test case contains two numbers n and k ($1 \le k < n \le 2 \cdot 10^5$) — the number of rooms and friends, respectively.

The next line of the test case contains k integers $x_1, x_2, \dots, x_k$ ($2 \le x_i \le n$) — numbers of rooms with friends. All x_i are different.

The next $n-1$ lines contain descriptions of the corridors, two numbers per line v_j and u_j ($1 \le u_j, v_j \le n$) — numbers of rooms that connect the j corridor. All corridors are bidirectional. From any room, you can go to any other by moving along the corridors.

It is guaranteed that the sum of the values n over all test cases in the test is not greater than $2 \cdot 10^5$.

Output

Print t lines, each line containing the answer to the corresponding test case. The answer to a test case should be -1 if Vlad wins anyway and a minimal number of friends otherwise.

Example

input

```
4

8 2
5 3
4 7
2 5
1 6
3 6
7 2
1 7
6 8

8 4
6 5 7 3
4 7
2 5
1 6
3 6
7 2
1 7
6 8

3 1
2
1 2
2 3

3 2
2 3
3 1
1 2
```

output

```
-1
2
1
2
```

Note

In the first set of inputs, even if all the friends stay in the maze, Vlad can still win. Therefore, the answer is "-1".

In the second set of inputs it is enough to leave friends from rooms 6 and 7. Then Vlad will not be able to win. The answer is "2".

In the third and fourth sets of inputs Vlad cannot win only if all his friends stay in the maze. Therefore the answers are "1" and "2".

F. ATM and Students

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Polycarp started working at a bank. He was assigned to monitor the ATM. The ATM initially contains s rubles.

A queue of n students lined up to him. Each student wants to either withdraw a certain amount of money or deposit it into an account. If a_i is positive, then the student credits that amount of money via ATM. Otherwise, the student withdraws $|a_i|$ rubles.

In the beginning, the ATM is turned off and an arbitrary number of students are not served. At some point, Polycarp turns on the ATM, which has an initial amount of s rubles. Then, the remaining students start queueing at the ATM. If at some point in time there is less money in the ATM than the student wants to withdraw, then the student is not served and Polycarp turns off the ATM and does not turn it on anymore.

More formally, the students that are served are forming a **contiguous subsequence**.

Polycarp wants the ATM to serve the **maximum** number of students. Help him in this matter. Print the numbers of the first and last student, or determine that he will not be able to serve anyone.

In other words, find such a longest continuous segment of students that, starting with the sum of s at the ATM, all these students will be served. ATM serves students consistently (i.e. one after another in the order of the queue).

Input

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

Each test case consists of two lines. The first one contains integers n and s ($1 \le n \le 2 \cdot 10^5$; $0 \le s \le 10^9$) — the length of the a array and the initial amount of rubles in the ATM. The second contains n integers $a_1, a_2, \dots, a_n$ ($-10^9 \le a_i \le 10^9$) — elements of the a array. Note that a_i can be zero.

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

Output

Print t lines, each line must contain the answer to the corresponding set of input data: if the answer exists, print the numbers of the first and last served student. If there is no solution, then print -1 on the line.

If there are several possible answers, print any.

Example

input

```
3
4 10
-16 2 -6 8
3 1000
-100000 -100000 -100000
6 0
2 6 -164 1 -1 -6543
```

output

```
2 4
-1
1 2
```

Note

In the first test case, the only correct answer is 2 4, since when serving students, the number of rubles at the ATM does not become negative, and this is the maximum number of students that can be served.

In the second test case, the answer is -1, as there is not enough money for any student at the ATM.

In the third test case, the answer can be either 1 2 or 4 5.

G. Robot and Candies

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

Polycarp has a rectangular field of $n \times m$ cells (the size of the $n \cdot m$ field does not exceed 10^6 cells, $m \ge 2$), in each cell of which there can be candy. There are n rows and m columns in the field.

Let's denote a cell with coordinates x vertically and y horizontally by (x, y). Then the top-left cell will be denoted as $(1, 1)$, and the bottom-right cell will be denoted as (n, m).

If there is candy in the cell, then the cell is marked with the symbol '1', otherwise — with the symbol '0'.

Polycarp made a Robot that can collect candy. The Robot can move from (x, y) either to $(x + 1, y + 1)$, or to $(x + 1, y - 1)$. If the Robot is in a cell that contains candy, it takes it.

While there is at least one candy on the field, the following procedure is executed:

- Polycarp puts the Robot in an arbitrary cell on the **topmost row** of the field. He himself chooses in which cell to place the Robot. It is allowed to put the Robot in the same cell multiple times.

- The Robot moves across the field and collects candies. He controls the Robot.
- When the Robot leaves the field, Polycarp takes it. If there are still candies left, Polycarp repeats the procedure.

Find the **minimum** number of times Polycarp needs to put the Robot on the topmost row of the field in order to collect all the candies. It is guaranteed that Polycarp can always collect all the candies.

Input

The first line of input data contains an integer t ($1 \leq t \leq 10^4$) — the number of input data sets in the test.

Before each input data, there is a blank line in the test. Next is a line that contains integers n and m ($2 \leq m$, $2 \leq n \cdot m \leq 10^6$) — field sizes. This is followed by n lines, i-th of which describes the i-th line of the field. Each of them is a string of size m characters: the symbol '1' corresponds to a cell with candy, the symbol '0' — an empty cell.

It is guaranteed that the sum of $n \cdot m$ values for all input data sets in the test does not exceed 10^6.

Output

Print t lines, each line should contain the answer to the corresponding set of input data: the minimum number of times Polycarpus needs to put the Robot on the topmost row of the field in order to collect all the candies.

Example

input

```
4

2 2
00
00

3 3
100
000
101

4 5
01000
00001
00010
10000

3 3
111
111
111
```

output

```
0
2
2
4
```

Note

In the first set Polycarp may not put the Robot on the field at all, so the answer "0"

In the second set, Polycarp will need to place the robot on the field twice. The Robot can collect candies like this: for the first time Polycarp puts the Robot in the cell $(1, 1)$ and collects candies at the positions $(1, 1)$ and $(3, 3)$. The second time Polycarp can again put the Robot in $(1, 1)$, and then the Robot will move first to $(2, 2)$, then to $(3, 1)$ and collect the last candy.

In the fourth set, you can show that the Robot cannot collect all the candies in three passes.