A. Shortest Path with Obstacle

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are three cells on an infinite 2-dimensional grid, labeled A, B, and F. Find the length of the shortest path from A to B if:

- in one move you can go to any of the four adjacent cells sharing a side;
- visiting the cell F is forbidden (it is an obstacle).

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow. Before each test case, there is an empty line.

Each test case contains three lines. The first one contains two integers x_A, y_A ($1 \le x_A, y_A \le 1000$) — coordinates of the start cell A. The second one contains two integers x_B, y_B ($1 \le x_B, y_B \le 1000$) — coordinates of the finish cell B. The third one contains two integers x_F, y_F ($1 \le x_F, y_F \le 1000$) — coordinates of the forbidden cell F. All cells are distinct.

Coordinate x corresponds to the column number and coordinate y corresponds to the row number (see the pictures below).

Output

Output t lines. The i-th line should contain the answer for the i-th test case: the length of the shortest path from the cell A to the cell B if the cell F is not allowed to be visited.

Example

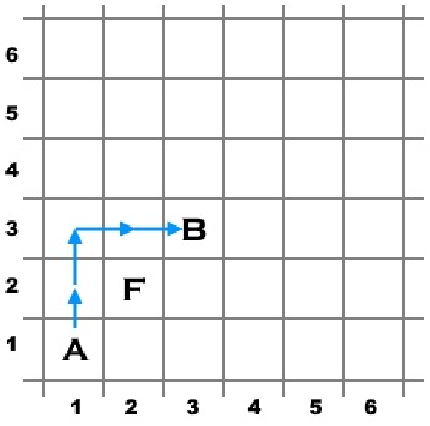
input

```
7

1 1
3 3
2 2

2 5
2 1
2 3

1000 42
1000 1
1000 1000

1 10
3 10
2 10

3 8
7 8
3 7

2 1
4 1
1 1

1 344
1 10
1 1
```

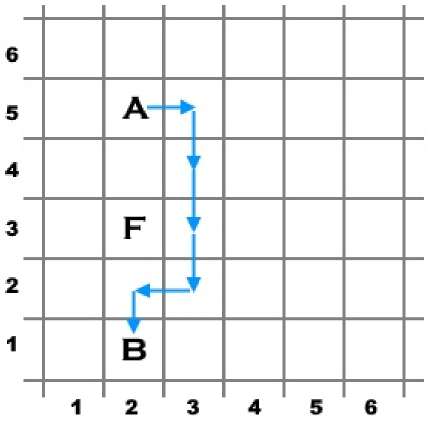
output

```
4
6
41
4
4
2
334
```

Note



An example of a possible shortest path for the first test case.



An example of a possible shortest path for the second test case.

B. Alphabetical Strings

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A string s of length n $(1 \leq n \leq 26)$ is called *alphabetical* if it can be obtained using the following algorithm:

- first, write an empty string to s (i.e. perform the assignment s `:= ""`);
- then perform the next step n times;
- at the i-th step take i-th lowercase letter of the Latin alphabet and write it either to the left of the string s or to the right of the string s (i.e. perform the assignment $s := c + s$ or $s := s + c$, where c is the i-th letter of the Latin alphabet).

In other words, iterate over the n first letters of the Latin alphabet starting from 'a' and etc. Each time we prepend a letter to the left of the string s or append a letter to the right of the string s. Strings that can be obtained in that way are alphabetical.

For example, the following strings are alphabetical: "`a`", "`ba`", "`ab`", "`bac`" and "`ihfcbadeg`". The following strings **are not** alphabetical: "`z`", "`aa`", "`ca`", "`acb`", "`xyz`" and "`ddcba`".

From the given string, determine if it is alphabetical.

Input

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

Each test case is written on a separate line that contains one string s. String s consists of lowercase letters of the Latin alphabet and has a length between 1 and 26, inclusive.

Output

Output t lines, each of them must contain the answer to the corresponding test case. Output YES if the given string s is alphabetical and NO otherwise.

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

Example

input

```
11
a
ba
ab
bac
ihfcbadeg
z
```

```
aa
ca
acb
xyz
ddcba
```

output

```
YES
YES
YES
YES
YES
NO
NO
NO
NO
NO
NO
```

Note

The example contains test cases from the main part of the condition.

C. Pair Programming

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Monocarp and Polycarp are learning new programming techniques. Now they decided to try pair programming.

It's known that they have worked together on the same file for $n + m$ minutes. Every minute exactly one of them made one change to the file. Before they started, there were already k lines written in the file.

Every minute exactly one of them does one of two actions: adds a new line to the end of the file or changes one of its lines.

Monocarp worked in total for n minutes and performed the sequence of actions $[a_1, a_2, \dots, a_n]$. If $a_i = 0$, then he adds a new line to the end of the file. If $a_i > 0$, then he changes the line with the number a_i. Monocarp performed actions strictly in this order: a_1, then a_2, ..., a_n.

Polycarp worked in total for m minutes and performed the sequence of actions $[b_1, b_2, \dots, b_m]$. If $b_j = 0$, then he adds a new line to the end of the file. If $b_j > 0$, then he changes the line with the number b_j. Polycarp performed actions strictly in this order: b_1, then b_2, ..., b_m.

Restore their common sequence of actions of length $n + m$ such that all actions would be correct — there should be no changes to lines that do not yet exist. Keep in mind that in the common sequence Monocarp's actions should form the subsequence $[a_1, a_2, \dots, a_n]$ and Polycarp's — subsequence $[b_1, b_2, \dots, b_m]$. They can replace each other at the computer any number of times.

Let's look at an example. Suppose $k = 3$. Monocarp first changed the line with the number 2 and then added a new line (thus, $n = 2, \ a = [2, 0]$). Polycarp first added a new line and then changed the line with the number 5 (thus, $m = 2, \ b = [0, 5]$).

Since the initial length of the file was 3, in order for Polycarp to change line number 5 two new lines must be added beforehand. Examples of correct sequences of changes, in this case, would be $[0, 2, 0, 5]$ and $[2, 0, 0, 5]$. Changes $[0, 0, 5, 2]$ (wrong order of actions) and $[0, 5, 2, 0]$ (line 5 cannot be edited yet) are not correct.

Input

The first line contains an integer t ($1 \le t \le 1000$). Then t test cases follow. Before each test case, there is an empty line.

Each test case contains three lines. The first line contains three integers k, n, m ($0 \le k \le 100$, $1 \le n, m \le 100$) — the initial number of lines in file and lengths of Monocarp's and Polycarp's sequences of changes respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 300$).

The third line contains m integers $b_1, b_2, \dots, b_m$ ($0 \le b_j \le 300$).

Output

For each test case print any correct common sequence of Monocarp's and Polycarp's actions of length $n + m$ or -1 if such sequence doesn't exist.

Example

input

```
5

3 2 2
2 0
0 5

4 3 2
2 0 5
0 6
```

```
0 2 2
1 0
2 3

5 4 4
6 0 8 0
0 7 0 9

5 4 1
8 7 8 0
0
```

output

```
2 0 0 5
0 2 0 6 5
-1
0 6 0 7 0 8 0 9
-1
```

D. Co-growing Sequence

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A sequence of non-negative integers $a_1, a_2, \dots, a_n$ is called *growing* if for all i from 1 to $n-1$ all ones (of binary representation) in a_i are in the places of ones (of binary representation) in a_{i+1} (in other words, $a_i \,\&\, a_{i+1} = a_i$, where $\&$ denotes bitwise AND). If $n = 1$ then the sequence is considered *growing* as well.

For example, the following four sequences are growing:

- $[2, 3, 15, 175]$ — in binary it's $[10_2, 11_2, 1111_2, 10101111_2]$;
- $[5]$ — in binary it's $[101_2]$;
- $[1, 3, 7, 15]$ — in binary it's $[1_2, 11_2, 111_2, 1111_2]$;
- $[0, 0, 0]$ — in binary it's $[0_2, 0_2, 0_2]$.

The following three sequences are non-growing:

- $[3, 4, 5]$ — in binary it's $[11_2, 100_2, 101_2]$;
- $[5, 4, 3]$ — in binary it's $[101_2, 100_2, 011_2]$;
- $[1, 2, 4, 8]$ — in binary it's $[0001_2, 0010_2, 0100_2, 1000_2]$.

Consider two sequences of non-negative integers $x_1, x_2, \dots, x_n$ and $y_1, y_2, \dots, y_n$. Let's call this pair of sequences *co-growing* if the sequence $x_1 \oplus y_1, x_2 \oplus y_2, \dots, x_n \oplus y_n$ is *growing* where $\oplus$ denotes bitwise XOR.

You are given a sequence of integers $x_1, x_2, \dots, x_n$. Find the lexicographically minimal sequence $y_1, y_2, \dots, y_n$ such that sequences x_i and y_i are co-growing.

The sequence $a_1, a_2, \dots, a_n$ is lexicographically smaller than the sequence $b_1, b_2, \dots, b_n$ if there exists $1 \le k \le n$ such that $a_i = b_i$ for any $1 \le i < k$ but $a_k < b_k$.

Input

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

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

The second line contains n integers $x_1, x_2, \dots, x_n$ ($0 \le x_i < 2^{30}$) — elements of the sequence x_i.

It is guaranteed that the sum of n overall all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print n integers $y_1, y_2, \dots, y_n$ ($0 \le y_i < 2^{30}$) — lexicographically minimal sequence such that such that it's co-growing with given sequence x_i.

Example

input

```
5
4
1 3 7 15
4
1 2 4 8
5
1 2 3 4 5
4
11 13 15 1
1
0
```

output

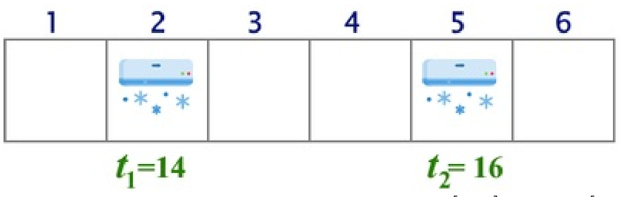
```
0 0 0 0
0 1 3 7
0 1 0 3 2
0 2 0 14
0
```

E. Air Conditioners

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

On a strip of land of length n there are k air conditioners: the i-th air conditioner is placed in cell a_i ($1 \le a_i \le n$). Two or more air conditioners cannot be placed in the same cell (i.e. all a_i are distinct).

Each air conditioner is characterized by one parameter: temperature. The i-th air conditioner is set to the temperature t_i.



Example of strip of length $n = 6$, where $k = 2$, $a = [2, 5]$ and $t = [14, 16]$.

For each cell i ($1 \le i \le n$) find it's temperature, that can be calculated by the formula

$$\min_{1 \le j \le k}(t_j + |a_j - i|),$$

where $|a_j - i|$ denotes absolute value of the difference $a_j - i$.

In other words, the temperature in cell i is equal to the minimum among the temperatures of air conditioners, increased by the distance from it to the cell i.

Let's look at an example. Consider that $n = 6, k = 2$, the first air conditioner is placed in cell $a_1 = 2$ and is set to the temperature $t_1 = 14$ and the second air conditioner is placed in cell $a_2 = 5$ and is set to the temperature $t_2 = 16$. In that case temperatures in cells are:

1. temperature in cell 1 is: $\min(14 + |2 - 1|, 16 + |5 - 1|) = \min(14 + 1, 16 + 4) = \min(15, 20) = 15$;
2. temperature in cell 2 is: $\min(14 + |2 - 2|, 16 + |5 - 2|) = \min(14 + 0, 16 + 3) = \min(14, 19) = 14$;
3. temperature in cell 3 is: $\min(14 + |2 - 3|, 16 + |5 - 3|) = \min(14 + 1, 16 + 2) = \min(15, 18) = 15$;
4. temperature in cell 4 is: $\min(14 + |2 - 4|, 16 + |5 - 4|) = \min(14 + 2, 16 + 1) = \min(16, 17) = 16$;
5. temperature in cell 5 is: $\min(14 + |2 - 5|, 16 + |5 - 5|) = \min(14 + 3, 16 + 0) = \min(17, 16) = 16$;
6. temperature in cell 6 is: $\min(14 + |2 - 6|, 16 + |5 - 6|) = \min(14 + 4, 16 + 1) = \min(18, 17) = 17$.

For each cell from 1 to n find the temperature in it.

Input

The first line contains one integer q ($1 \le q \le 10^4$) — the number of test cases in the input. Then test cases follow. Before each test case, there is an empty line.

Each test case contains three lines. The first line contains two integers n ($1 \le n \le 3 \cdot 10^5$) and k ($1 \le k \le n$) — the length of the strip of land and the number of air conditioners respectively.

The second line contains k integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n$) — positions of air conditioners on the strip of land.

The third line contains k integers $t_1, t_2, \ldots, t_k$ ($1 \le t_i \le 10^9$) — temperatures of air conditioners.

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

Output

For each test case output n integers separated by space: temperatures of air in cells.

Example

input

```
5

6 2
2 5
14 16

10 1
7
30

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

```
7 1
1
1000000000

6 3
6 1 3
5 5 5
```

output

```
15 14 15 16 16 17
36 35 34 33 32 31 30 31 32 33
1 2 3 4 5
1000000000 1000000001 1000000002 1000000003 1000000004 1000000005 1000000006
5 6 5 6 6 5
```

F. Array Stabilization (GCD version)

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array of positive integers $a = [a_0, a_1, \dots, a_{n-1}]$ ($n \ge 2$).

In one step, the array a is replaced with another array of length n, in which each element is the greatest common divisor (GCD) of two neighboring elements (the element itself and its right neighbor; consider that the right neighbor of the $(n-1)$-th element is the 0-th element).

Formally speaking, a new array $b = [b_0, b_1, \dots, b_{n-1}]$ is being built from array $a = [a_0, a_1, \dots, a_{n-1}]$ such that $b_i = \gcd(a_i, a_{(i+1) \bmod n})$, where $\gcd(x, y)$ is the greatest common divisor of x and y, and $x \bmod y$ is the remainder of x dividing by y. In one step the array b is built and then the array a is replaced with b (that is, the assignment a := b is taking place).

For example, if $a = [16, 24, 10, 5]$ then $b = [\gcd(16, 24)$, $\gcd(24, 10)$, $\gcd(10, 5)$, $\gcd(5, 16)] = [8, 2, 5, 1]$. Thus, after one step the array $a = [16, 24, 10, 5]$ will be equal to $[8, 2, 5, 1]$.

For a given array a, find the minimum number of steps after which all values a_i become equal (that is, $a_0 = a_1 = \dots = a_{n-1}$). If the original array a consists of identical elements then consider the number of steps is equal to 0.

Input

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

Each test case contains two lines. The first line contains an integer n ($2 \le n \le 2 \cdot 10^5$) — length of the sequence a. The second line contains n integers $a_0, a_1, \dots, a_{n-1}$ ($1 \le a_i \le 10^6$).

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

Output

Print t numbers — answers for each test case.

Example

input

```
5
4
16 24 10 5
4
42 42 42 42
3
4 6 4
5
1 2 3 4 5
6
9 9 27 9 9 63
```

output

```
3
0
2
1
1
```

G. How Many Paths?

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a directed graph G which can contain loops (edges from a vertex to itself). Multi-edges are absent in G which means that for all ordered pairs (u, v) exists at most one edge from u to v. Vertices are numbered from 1 to n.

A path from u to v is a sequence of edges such that:

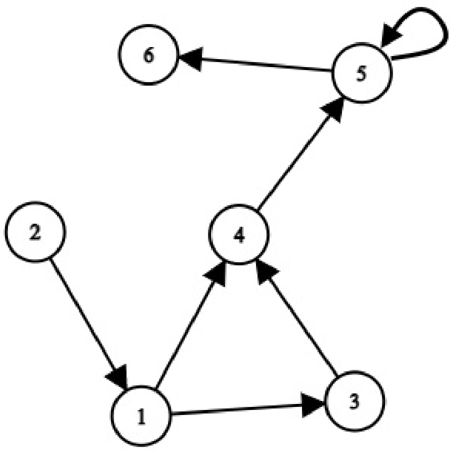
- vertex u is the start of the first edge in the path;
- vertex v is the end of the last edge in the path;
- for all pairs of adjacent edges next edge starts at the vertex that the previous edge ends on.

We will assume that the empty sequence of edges is a path from u to u.

For each vertex v output one of four values:

- 0, if there are no paths from 1 to v;
- 1, if there is only one path from 1 to v;
- 2, if there is more than one path from 1 to v and the number of paths is finite;
- -1, if the number of paths from 1 to v is infinite.

Let's look at the example shown in the figure.



Then:

- the answer for vertex 1 is 1: there is only one path from 1 to 1 (path with length 0);
- the answer for vertex 2 is 0: there are no paths from 1 to 2;
- the answer for vertex 3 is 1: there is only one path from 1 to 3 (it is the edge $(1, 3)$);
- the answer for vertex 4 is 2: there are more than one paths from 1 to 4 and the number of paths are finite (two paths: $[(1, 3), (3, 4)]$ and $[(1, 4)]$);
- the answer for vertex 5 is -1: the number of paths from 1 to 5 is infinite (the loop can be used in a path many times);
- the answer for vertex 6 is -1: the number of paths from 1 to 6 is infinite (the loop can be used in a path many times).

Input

The first contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow. Before each test case, there is an empty line.

The first line of the test case contains two integers n and m ($1 \le n \le 4 \cdot 10^5, 0 \le m \le 4 \cdot 10^5$) — numbers of vertices and edges in graph respectively. The next m lines contain edges descriptions. Each line contains two integers a_i, b_i ($1 \le a_i, b_i \le n$) — the start and the end of the i-th edge. The vertices of the graph are numbered from 1 to n. The given graph can contain loops (it is possible that $a_i = b_i$), but cannot contain multi-edges (it is not possible that $a_i = a_j$ and $b_i = b_j$ for $i \ne j$).

The sum of n over all test cases does not exceed $4 \cdot 10^5$. Similarly, the sum of m over all test cases does not exceed $4 \cdot 10^5$.

Output

Output t lines. The i-th line should contain an answer for the i-th test case: a sequence of n integers from -1 to 2.

Example

input

```
5

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

1 0

3 3
1 2
```

```
2 3
3 1

5 0

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

output

```
1 0 1 2 -1 -1
1
-1 -1 -1
1 0 0 0 0
1 1 2 1
```

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform