Kotlin Heroes: Episode 7

A. Travel to Bertown

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

Vika has decided to go on a trip to Bertown. She comes to Bertown on the k-th day. Vika does not want to spend money on hotels, fortunately she has n friends who live in Bertown and allow Vika to stay at their houses. The i-th friend told Vika that she could stay at their house from the l_i-th day till the r_i-th day, inclusive.

Vika decided that she would stay at no more than one friend's house. Help Vika determine the maximum number of days that she will be able to stay in Bertown.

Input

The first line contains a single integer t ($1 \le t \le 500$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n, k \le 100$) — the number of Vika's friends and the day when Vika comes to Bertown.

Then follow n lines, each containing two integers l_i and r_i ($1 \le l_i \le r_i \le 100$) denoting the period when Vika can stay at the i-th friend's house.

Output

For each test case, output a single integer — the maximum number of days that Vika can stay in Bertown (or 0 if none of Vika's friends can host her on day k).

Example

input

```
3
3 3
1 4
2 6
4 10
2 4
2 3
5 8
2 4
4 4
1 3
```

output

```
4
0
1
```

Note

In the first example, Vika can stay at the 2-nd friend's house from 3-rd to 6-th day.

In the second example, none of Vika's friends can host her on the 4-th day.

In the third example, Vika can stay at the 1-st friend's house, but only for the 4-th day.

B. Nearest Point Function

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

Your friend has created a nearest point function. For a given array of integer points x (**sorted** in ascending order, without any duplicates) and a point y it can find the nearest point from x to the point y. In other words, it will find such a point x_i that $|y - x_i|$ is the minimum possible, where $|a|$ is the absolute value of a.

For example, if $x = [1, 2, 5, 7, 9, 11]$, the answer for $y = 3$ will be 2, the answer for $y = 5$ will be 5 and the answer for $y = 100$ will be 11.

This function is a bit buggy, though. If there are several nearest points to the given one, the function crashes. For example, if $x = [1, 2, 5, 7, 9, 11]$ (as above) and $y = 6$, the function will crash, since points 5 and 7 are both the nearest points for 6.

Your task is, for a given array of integer points x (**sorted** in ascending order, without any duplicates), determine if it is possible to cause the function to crash by choosing some **integer** point y.

You have to answer t independent test cases.

Input

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

The first line of the test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of points in the array x.

The second line of the test case contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^9$), where x_i is the i-th point in the array x.

All points in the array x are distinct, and the array x is sorted in ascending order (in other words, $x_1 < x_2 < \ldots < x_n$).

The sum of n over the test cases in the input does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print YES if it is possible to find some **integer** point y that will crash the function and NO otherwise.

Example

input

```
7
2
1 3
2
1 100
3
1 50 101
2
1 1000000000
2
1 999999999
6
1 2 5 7 9 11
6
1 2 5 8 9 12
```

output

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

Note

In the first test case of the example, the function crashes if $y = 2$ is chosen.

In the fifth test case of the example, the function crashes if $y = 500000000$ is chosen.

In the sixth test case of the example, the function crashes if $y = 10$ is chosen.

C. Sweets

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Anya came to her friend's birthday party. There are n delicious sweets on a circle table (for convenience, we will number them from 1 to n in clockwise direction). For each of the sweets, it is known whether Anya likes it or not. Anya decided that she should eat all the sweets that are on the table, and she likes them.

However, eating all the sweets in some random order is too boring. Therefore, Anya came up with a game that will make the process of eating sweets more interesting.

The game is played according to the following rules:

- if there are no sweets that Anya likes left on the table, then the game ends;
- at the very beginning of the game, if there is at least one sweet on the table that Anya wants to eat, she eats the sweet number 1;
- after Anya has eaten some sweet, she counts k sweets clockwise, starting from the next sweet in the circle, and eats the k-th sweet in the count (if there are less than k sweets in the circle, some sweets can participate in the count more than once, and the last sweet in the count is picked). Obviously, the sweets that have already been eaten do not participate in the count.

For example, let 6 sweets be arranged in a circle, Anya likes sweets 4, 5 and 6, $k = 4$. Then the game goes as follows:

1. initially there are sweets $[1, 2, 3, 4, 5, 6]$ on the table, Anya chooses the sweet number 1.
2. Anya eats the sweet number 1, after that, there are sweets $[2, 3, 4, 5, 6]$ on the table. Anya counts 4 sweets, starting with the 2-nd sweet, and stops at the 5-th sweet.

3. Anya eats the 5-th sweet, after that, there are sweets $[2, 3, 4, 6]$ on the table. Anya counts 4 sweets, starting with the 6-th sweet, and stops at the 4-th sweet.
4. Anya eats the sweet number 4, after that, there are sweets $[2, 3, 6]$ on the table. Anya counts 4 sweets, starting with the 6-th sweet, and stops at the 6-th sweet.
5. Anya eats the sweet number 6, after that, there are sweets $[2, 3]$ on the table. There are no sweets that Anya likes on the table, so the game ends.

Your task is to calculate the number of sweets that Anya will eat.

Input

The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 5000$) — the number of sweets and the parameter k.

The next line contains the string s, where $s_i = 1$ if Anya likes i-th sweet, and $s_i = 0$ otherwise.

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 number of sweets that Anya will eat.

Example

input

```
4
6 4
000111
7 3
0000100
3 2
000
5 1
10011
```

output

```
4
4
0
5
```

Note

The first test case of the example is described in the statement.

D. String Searching

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array s consisting of n different strings. Each string consists of m lowercase Latin letters.

You have to respond to q queries. Each query contains a string t of length $m+1$. Count the number of indices i, such that the string t can be obtained from the string s_i, if it is allowed to insert one letter in an arbitrary position.

Input

The first line contains two integers n and m ($1 \le n \le 10^5$; $1 \le m \le 10$) — the number of strings in the array and the length of each string.

The following n lines contain strings s_i. All of the given strings are different.

The next line contains a single integer q ($1 \le q \le 10^5$) — the number of queries.

The following q lines contain query strings t of length $m+1$.

Output

For each query, print the number of indices i, such that a string from the query can be obtained from the string s_i, if it is allowed to insert one letter in an arbitrary position.

Examples

input

```
2 1
a
c
4
aa
ca
mm
cf
```

output

```
1
2
0
1
```

input

```
6 3
dba
abd
cbb
ada
add
bdd
5
ccbb
abdd
adba
bada
dddd
```

output

```
1
3
2
1
0
```

Note

Explanation of the first test of the example:

- the string a can be transformed into aa by inserting one letter;
- both strings a and c can be transformed into ca by inserting one letter;
- neither a nor c can be transformed into mm by inserting one letter;
- c can be transformed into cf by inserting one letter.

E. Chess Team Forming

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

Polycarp is coaching a team for an upcoming game at the chess tournament. A complete team for the tournament should consist of $n+1$ members.

There are n members in his team, the i-th member's skill value is a_i. Polycarp is yet to choose the final member for the team.

The opposing team has $n+1$ members, the j-th member's skill value is b_j.

Polycarp has m options for the final player of the team. The k-th of them has a skill value c_k.

Before the game starts, Polycarp pairs up the members of his team with the members of the opposing team. Every member of both teams is in exactly one pair. The difficulty of a game for a certain player is the difference between his paired opponent's skill and his own skill. So if the i-th player of the Polycarp's team is paired up with the j-th member of the opposing team, then the difficulty is equal to $b_j - a_i$. The difficulty of the game for a team is the maximum difficulty of all its players.

So, before the game starts, Polycarp wants to pair up the players in such a way that the difficulty of the game for his team is minimized.

For each of the m options for the final player print the smallest difficulty of the game for the team Polycarp can achieve.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of members in the Polycarp's team.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the skill value of the i-th player of the Polycarp's team.

The third line contains $n+1$ integers $b_1, b_2, \dots, b_{n+1}$ ($1 \le b_j \le 10^9$), where b_j is the skill value of the j-th player of the opposing team.

The fourth line contains a single integer m ($1 \le m \le 2 \cdot 10^5$) — the number of options for the final player.

The fifth line contains m integers $c_1, c_2, \dots, c_m$ ($1 \le c_k \le 10^9$), where c_k is the skill value of the k-th of the options for the final player of the Polycarp's team.

Output

Print m integers — the k-th of them should be equal to the smallest difficulty of the game for the team Polycarp can achieve if he picks the k-th option player as the final one.

Examples

input

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

output

```
4 2 3 4 2
```

input

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

output

```
3 3 3 3 3 2 2 2 1 0
```

input

```
1
10
10 5
2
5 15
```

output

```
0 -5
```

Note

In the first example the optimal pairings for the first three options are the following. Note that there might be multiple valid pairing for the minimum answer.

First option:

Polycarp's team: `6 1 3 1 10 4`

Opposing team: `9 4 2 5  9 8`

The respective difficulties of the game for each player are: `3, 3, -1, 4, -1, 4`. The maximum is 4, thus it's the difficulty of the game for the team.

Second option:

Polycarp's team: `10 4 1 3 7 6`

Opposing team: `  9 4 2 5 9 8`

Third option:

Polycarp's team: `6 3 1 4 10 6`

Opposing team: `9 4 2 5  9 8`

F. Binary String Partition

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

Let's call a string t consisting of characters `0` and/or `1` *beautiful*, if either the number of occurrences of character `0` in this string does not exceed k, or the number of occurrences of characters `1` in this string does not exceed k (or both). For example, if $k = 3$, the strings `101010`, `111`, `0`, `00000`, `1111111000` are *beautiful*, and the strings `1111110000`, `01010101` are not *beautiful*.

You are given a string s. You have to divide it into the minimum possible number of *beautiful* strings, i. e., find a sequence of strings $t_1, t_2, \dots, t_m$ such that every t_i is *beautiful*, $t_1 + t_2 + \dots + t_m = s$ (where $+$ is the concatenation operator), and m is minimum possible.

For every k from 1 to $|s|$, find the minimum possible number of strings such that s can be divided into them (i. e. the minimum possible m in the partition).

Input

The only line contains one string s ($1 \le |s| \le 2 \cdot 10^5$). Each character of s is either `0` or `1`.

Output

Print $|s|$ integers. The i-th integer should be equal to the minimum number of strings in the partition of s, when $k = i$.

Examples

input

```
00100010
```

output

```
2 1 1 1 1 1 1 1
```

input

```
1001011100
```

output

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

G. Biome Map

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

Polycarp decided to generate a biome map for his game. A map is a matrix divided into cells 1×1. Each cell of the map must contain one of the available biomes.

Each biome is defined by two parameters: temperature (an integer from 1 to n) and humidity (an integer from 1 to m). But not for every temperature/humidity combination, a corresponding biome is available.

The biome map should be generated according to the following rules:

- each cell of the map belongs to exactly one biome;
- each available biome has at least one cell on the map;
- if two cells of the map are adjacent by the side and they belong to biomes with parameters (t_1, h_1) and (t_2, h_2), respectively, then the equality $|t_1 - t_2| + |h_1 - h_2| = 1$ holds;
- let the number of available biomes be equal to k, then the number of rows and columns of the map (separately) should not exceed k.

Help Polycarp generate a biome map that meets all the conditions described above (or report that this is impossible).

Input

The first line contains a single integer t ($1 \le t \le 20$) — the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n, m \le 10$) — maximum temperature and humidity parameters.

The following n lines contain m integers each $a_{i,1}, a_{i,2}, \dots, a_{i,m}$ ($0 \le a_{i,j} \le 100$), where $a_{i,j}$ — the biome identifier with the parameters (i, j), if $a_{i,j} \ne 0$, otherwise the biome with such parameters is not available.

All biome identifiers are different, and there are at least two biomes available.

Output

For each test case, print the answer in the following format:

- print -1 in a single line if there is no map that meets all the conditions;
- otherwise, in the first line, print two integers h and w — the number of rows and columns of the map, respectively. In the following h lines, print w integers — the identifiers of the biomes in the corresponding cells of the map.

Example

input

```
4
2 3
0 2 5
0 1 0
3 5
0 3 4 9 11
1 5 0 10 12
0 6 7 0 0
2 2
2 0
0 5
1 2
13 37
```

output

```
1 3
5 2 1
2 8
11 9 4 3 5 1 5 6
12 10 9 4 3 5 6 7
-1
1 2
```

```
13 37
```

H. Submatrices

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given matrix s that contains n rows and m columns. Each element of a matrix is one of the 5 first Latin letters (in upper case).

For each k ($1 \le k \le 5$) calculate the number of submatrices that contain exactly k different letters. Recall that a submatrix of matrix s is a matrix that can be obtained from s after removing several (possibly zero) first rows, several (possibly zero) last rows, several (possibly zero) first columns, and several (possibly zero) last columns. If some submatrix can be obtained from s in two or more ways, you have to count this submatrix the corresponding number of times.

Input

The first line contains two integers n and m ($1 \le n, m \le 800$).

Then n lines follow, each containing the string s_i ($|s_i| = m$) — i-th row of the matrix.

Output

For each k ($1 \le k \le 5$) print one integer — the number of submatrices that contain exactly k different letters.

Examples

input

```
2 3
ABB
ABA
```

output

```
9 9 0 0 0
```

input

```
6 6
EDCECE
EDDCEB
ACCECC
BAEEDC
DDDDEC
DDAEAD
```

output

```
56 94 131 103 57
```

input

```
3 10
AEAAEEEEEC
CEEAAEEEEE
CEEEEAACAA
```

output

```
78 153 99 0 0
```

I. Excursions

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

Irina works in an excursion company in Saratov. Today, she is going to organize excursions in the cities of Saratov and Engels.

There are n_1 sights in Saratov and n_2 sights in Engels. The cities are separated by a river, but there are m bus routes that go along the bridges and allow tourists to go from Saratov to Engels and vice versa. The i-th bus route goes from the x_i-th sight in Saratov to the y_i-th sight in Engels, and in the opposite direction as well.

Irina wants to plan excursions for the current day. The excursion trips start in Saratov in the morning, continue in Engels in the afternoon, and finish in Saratov in the evening.

Each tourist starts their excursion day at some sight in Saratov, k_i tourists start at the i-th sight. Then the tour guides lead them to Engels: at each sight in Saratov, a tour guide chooses a bus route leading from this sight to Engels, and **all the tourists starting from this sight** transfer to Engels along this bus route. After the excursions in Engels are finished, the same thing happens: at each sight in Engels, a tour guide chooses a bus route leading from this sight to Saratov, and **all the tourists at this sight** transfer to Saratov along this bus route.

This process can lead to a situation such that some tourists return to the same sight in Saratov where they started in the morning. Obviously, tourists don't like it; so Irina wants to choose where the tour guides take the tourists (both on the way from Saratov to Engels and on the way from Engels to Saratov), so that the minimum possible number of tourists return to the same sight where they started. Help Irina to find an optimal plan!

Input

The first line contains three integers n_1, n_2 and m ($1 \le n_1, n_2 \le 100$; $\max(n_1, n_2) \le m \le n_1 \cdot n_2$) — the number of sights in Saratov, the number of sights in Engels, and the number of bus routes, respectively.

The second line contains n_1 integers $k_1, k_2, \dots, k_{n_1}$ ($1 \le k_i \le 10^6$), where k_i is the number of tourists starting at the i-th sight in Saratov.

Then m lines follow, each describing a bus route: the i-th line contains two integers x_i and y_i ($1 \le x_i \le n_1$; $1 \le y_i \le n_2$) meaning that the i-th bus route connects the x_i-th sight in Saratov and the y_i-th sight in Engels. All these bus routes are distinct, and each sight has at least one bus route leading to/from it.

Output

Print one integer — the minimum possible number of tourists that will return to the same sight where they started.

Examples

input

```
2 1 2
10 20
1 1
2 1
```

output

```
10
```

input

```
3 3 6
10 20 30
1 3
3 1
2 3
2 1
3 2
1 2
```

output

```
0
```

J. Pawns

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is an infinite chessboard, divided into cells. The cell (x, y) is the cell on the intersection of the x_i-th row and y_i-th column. n black pawns are placed on the board, the i-th black pawn occupies the cell (x_i, y_i).

You want to capture all black pawns. In order to do so, you may perform the following actions:

- place a white pawn into any empty cell (any cell having integer coordinates can be chosen, as long as it doesn't contain any pawns);
- make a move with one of the white pawns according to the chess rules.

Recall that when you make a move with a white pawn in the cell (x, y), the chess rules allow you to choose exactly one of these actions:

- if there is a black pawn in the cell $(x+1, y-1)$, you can capture it — the black pawn is removed from the board, and the white pawn moves to $(x+1, y-1)$;
- if there is a black pawn in the cell $(x+1, y+1)$, you can capture it — the black pawn is removed from the board, and the white pawn moves to $(x+1, y+1)$;
- if the cell $(x+1, y)$ is empty, you can move the white pawn to that cell.

You may perform any finite sequence of actions (placing white pawns and moving them). You want to capture all of the black pawns, and it can be shown that it is always possible; and you want to do it placing as few white pawns as possible.

What is the minimum number of white pawns you have to place to capture all n black pawns?

Input

The first line contains one integer n ($1 \le n \le 5 \cdot 10^5$) — the number of black pawns.

Then n lines follow. The i-th line contains two integers x_i and y_i ($1 \le x_i, y_i \le 5 \cdot 10^5$) denoting a black pawn in the cell (x_i, y_i). No cell is occupied by two or more black pawns.

Output

Print one integer — the minimum number of white pawns you have to place to capture all n black pawns.

Examples

input

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

output

```
1
```

input

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

output

```
2
```

input

```
2
1 1
2 2
```

output

```
1
```