Lista AA 2022

A. Distance in Tree

3 seconds, 512 megabytes

A *tree* is a connected graph that doesn't contain any cycles.

The *distance* between two vertices of a tree is the length (in edges) of the shortest path between these vertices.

You are given a tree with n vertices and a positive number k. Find the number of distinct pairs of the vertices which have a distance of exactly k between them. Note that pairs (v, u) and (u, v) are considered to be the same pair.

Input

The first line contains two integers n and k ($1 \le n \le 50000$, $1 \le k \le 500$) — the number of vertices and the required distance between the vertices.

Next n - 1 lines describe the edges as "a_i b_i" (without the quotes) ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), where a_i and b_i are the vertices connected by the i-th edge. All given edges are different.

Output

Print a single integer — the number of distinct pairs of the tree's vertices which have a distance of exactly k between them.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

input
```
5 2
1 2
2 3
3 4
2 5
```
output
```
4
```

input
```
5 3
1 2
2 3
3 4
4 5
```
output
```
2
```

In the first sample the pairs of vertexes at distance 2 from each other are (1, 3), (1, 5), (3, 5) and (2, 4).

B. Longest Regular Bracket Sequence

2 seconds, 256 megabytes

This is yet another problem dealing with regular bracket sequences.

We should remind you that a bracket sequence is called regular, if by inserting «+» and «1» into it we can get a correct mathematical expression. For example, sequences «(())()», «()» and «(()(()))» are regular, while «)(», «(()» and «(()))(» are not.

You are given a string of «(» and «)» characters. You are to find its longest substring that is a regular bracket sequence. You are to find the number of such substrings as well.

Input

The first line of the input file contains a non-empty string, consisting of «(» and «)» characters. Its length does not exceed 10^6.

Output

Print the length of the longest substring that is a regular bracket sequence, and the number of such substrings. If there are no such substrings, write the only line containing "`0 1`".

input
```
)((())))(()())
```
output
```
6 2
```

input
```
))(
```
output
```
0 1
```

C. Dima and Lisa

1 second, 256 megabytes

Dima loves representing an odd number as the sum of multiple primes, and Lisa loves it when there are at most three primes. Help them to represent the given number as the sum of at most than three primes.

More formally, you are given an **odd** numer n. Find a set of numbers p_i ($1 \le i \le k$), such that

1. $1 \le k \le 3$
2. p_i is a prime
3. $\sum_{i=1}^{k} p_i = n$

The numbers p_i do not necessarily have to be distinct. It is guaranteed that at least one possible solution exists.

Input

The single line contains an odd number n ($3 \le n < 10^9$).

Output

In the first line print k ($1 \le k \le 3$), showing how many numbers are in the representation you found.

In the second line print numbers p_i in any order. If there are multiple possible solutions, you can print any of them.

input
```
27
```
output
```
3
5 11 11
```

A prime is an integer strictly larger than one that is divisible only by one and by itself.

D. Tic-tac-toe

1 second, 64 megabytes

Certainly, everyone is familiar with tic-tac-toe game. The rules are very simple indeed. Two players take turns marking the cells in a 3×3 grid (one player always draws crosses, the other — noughts). The player who succeeds first in placing three of his marks in a horizontal, vertical or diagonal line wins, and the game is finished. The player who draws crosses goes first. If the grid is filled, but neither Xs, nor 0s form the required line, a draw is announced.

You are given a 3×3 grid, each grid cell is empty, or occupied by a cross or a nought. You have to find the player (first or second), whose turn is next, or print one of the verdicts below:

- `illegal` — if the given board layout can't appear during a valid game;
- `the first player won` — if in the given board layout the first player has just won;
- `the second player won` — if in the given board layout the second player has just won;
- `draw` — if the given board layout has just let to a draw.

Input

The input consists of three lines, each of the lines contains characters "`.`", "`X`" or "`0`" (a period, a capital letter X, or a digit zero).

Output

Print one of the six verdicts: `first`, `second`, `illegal`, `the first player won`, `the second player won` or `draw`.

input

```
X0X
.0.
.X.
```

output

```
second
```

E. Multiple Testcases

2 seconds, 256 megabytes

So you decided to hold a contest on Codeforces. You prepared the problems: statements, solutions, checkers, validators, tests... Suddenly, your coordinator asks you to change all your tests to multiple testcases in the easiest problem!

Initially, each test in that problem is just an array. The maximum size of an array is $$$k$$$. For simplicity, the contents of arrays don't matter. You have $$$n$$$ tests — the $$$i$$$-th test is an array of size $$$m_i$$$ ($$$1 \le m_i \le k$$$).

Your coordinator asks you to distribute all of your arrays into multiple testcases. Each testcase can include multiple arrays. However, each testcase should include no more than $$$c_1$$$ arrays of size **greater than or equal to $$$1$$$** ($$$\ge 1$$$), no more than $$$c_2$$$ arrays of size **greater than or equal to $$$2$$$**, $$$\dots$$$, no more than $$$c_k$$$ arrays of size **greater than or equal to $$$k$$$**. Also, $$$c_1 \ge c_2 \ge \dots \ge c_k$$$.

So now your goal is to create the new testcases in such a way that:

- each of the initial arrays appears in **exactly one** testcase;
- for each testcase the given conditions hold;
- the number of testcases is minimum possible.

Print the minimum possible number of testcases you can achieve and the sizes of arrays included in each testcase.

Input

The first line contains two integers $$$n$$$ and $$$k$$$ ($$$1 \le n, k \le 2 \cdot 10^5$$$) — the number of initial tests and the limit for the size of each array.

The second line contains $$$n$$$ integers $$$m_1, m_2, \dots, m_n$$$ ($$$1 \le m_i \le k$$$) — the sizes of the arrays in the original tests.

The third line contains $$$k$$$ integers $$$c_1, c_2, \dots, c_k$$$ ($$$n \ge c_1 \ge c_2 \ge \dots \ge c_k \ge 1$$$); $$$c_i$$$ is the maximum number of arrays of size greater than or equal to $$$i$$$ you can have in a single testcase.

Output

In the first line print a single integer $$$ans$$$ ($$$1 \le ans \le n$$$) — the minimum number of testcases you can achieve.

Each of the next $$$ans$$$ lines should contain the description of a testcase in the following format:

$$$t$$$ $$$a_1$$$ $$$a_2$$$ $$$\dots$$$ $$$a_{t}$$$ ($$$1 \le t\le n$$$) — the testcase includes $$$t$$$ arrays, $$$a_i$$$ is the size of the $$$i$$$-th array in that testcase.

Each of the initial arrays should appear in **exactly one** testcase. In particular, it implies that the sum of $$$t$$$ over all $$$ans$$$ testcases should be equal to $$$n$$$.

Note that the answer always exists due to $$$c_k \ge 1$$$ (and therefore $$$c_1 \ge 1$$$).

If there are multiple answers, you can output any one of them.

input

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

output

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

input

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

output

```
2
3 8 5 7
3 10 8 1
```

input

```
5 1
1 1 1 1 1
5
```

output

```
1
5 1 1 1 1 1
```

input

```
5 1
1 1 1 1 1
1
```

output

```
5
1 1
1 1
1 1
1 1
1 1
```

In the first example there is no way to distribute the tests into less than $$$3$$$ testcases. The given answer satisfies the conditions: each of the testcases includes no more than $$$4$$$ arrays of size greater than or equal to $$$1$$$ and no more than $$$1$$$ array of sizes greater than or equal to $$$2$$$ and $$$3$$$.

Note that there are multiple valid answers for this test. For example, testcases with sizes $$$[[2], [1, 2], [3]]$$$ would also be correct.

However, testcases with sizes $$$[[1, 2], [2, 3]]$$$ would be incorrect because there are $$$2$$$ arrays of size greater than or equal to $$$2$$$ in the second testcase.

Note the difference between the third and the fourth examples. You can include up to $$$5$$$ arrays of size greater than or equal to $$$1$$$ in the third example, so you can put all arrays into a single testcase. And you can have only up to $$$1$$$ array in the fourth example. Thus, every array should be included in a separate testcase.

F. Array Differentiation

1 second, 256 megabytes

You are given a sequence of $$$n$$$ integers $$$a_1, \, a_2, \, \dots, \, a_n$$$.

Does there exist a sequence of $$$n$$$ integers $$$b_1, \, b_2, \, \dots, \, b_n$$$ such that the following property holds?

- For each $$$1 \le i \le n$$$, there exist two (not necessarily distinct) indices $$$j$$$ and $$$k$$$ ($$$1 \le j, \, k \le n$$$) such that $$$a_i = b_j - b_k$$$.

Input

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

The first line of each test case contains one integer $$$n$$$ ($$$1 \le n \le 10$$$).

The second line of each test case contains the $$$n$$$ integers $$$a_1, \, \dots, \, a_n$$$ ($$$-10^5 \le a_i \le 10^5$$$).

Output

For each test case, output a line containing `YES` if a sequence $$$b_1, \, \dots, \, b_n$$$ satisfying the required property exists, and `NO` otherwise.

input

```
5
5
4 -7 -1 5 10
1
0
3
1 10 100
4
-3 2 10 2
9
25 -171 250 174 152 242 100 -205 -258
```

output

```
YES
YES
NO
YES
YES
```

In the **first test case**, the sequence $$$b = [-9, \, 2, \, 1, \, 3, \, -2]$$$ satisfies the property. Indeed, the following holds:

- $$$a_1 = 4 = 2 - (-2) = b_2 - b_5$$$;
- $$$a_2 = -7 = -9 - (-2) = b_1 - b_5$$$;
- $$$a_3 = -1 = 1 - 2 = b_3 - b_2$$$;
- $$$a_4 = 5 = 3 - (-2) = b_4 - b_5$$$;
- $$$a_5 = 10 = 1 - (-9) = b_3 - b_1$$$.

In the **second test case**, it is sufficient to choose $$$b = [0]$$$, since $$$a_1 = 0 = 0 - 0 = b_1 - b_1$$$.

In the **third test case**, it is possible to show that no sequence $$$b$$$ of length $$$3$$$ satisfies the property.

G. The Treasure of The Segments

3 seconds, 256 megabytes

Polycarp found $$$n$$$ segments on the street. A segment with the index $$$i$$$ is described by two integers $$$l_i$$$ and $$$r_i$$$ — coordinates of the beginning and end of the segment, respectively. Polycarp realized that he didn't need all the segments, so he wanted to delete some of them.

Polycarp believes that a set of $$$k$$$ segments is good if there is a segment $$$[l_i, r_i]$$$ ($$$1 \leq i \leq k$$$) from the set, such that it intersects every segment from the set (the intersection must be a **point or segment**). For example, a set of $$$3$$$ segments $$$[[1, 4], [2, 3], [3, 6]]$$$ is good, since the segment $$$[2, 3]$$$ intersects each segment from the set. Set of $$$4$$$ segments $$$[[1, 2], [2, 3], [3, 5], [4, 5]]$$$ is not good.

Polycarp wonders, what is the minimum number of segments he has to delete so that the remaining segments form a good set?

Input

The first line contains a single integer $$$t$$$ ($$$1 \leq t \leq 2 \cdot 10^5$$$) — number of test cases. Then $$$t$$$ test cases follow.

The first line of each test case contains a single integer $$$n$$$ ($$$1 \leq n \leq 2 \cdot 10^5$$$) — the number of segments. This is followed by $$$n$$$ lines describing the segments.

Each segment is described by two integers $$$l$$$ and $$$r$$$ ($$$1 \leq l \leq r \leq 10^9$$$) — coordinates of the beginning and end of the segment, respectively.

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

Output

For each test case, output a single integer — the minimum number of segments that need to be deleted in order for the set of remaining segments to become good.

input

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

output

```
0
1
2
0
```

H. Minimax Problem

5 seconds, 512 megabytes

You are given $$$n$$$ arrays $$$a_1$$$, $$$a_2$$$, ..., $$$a_n$$$; each array consists of exactly $$$m$$$ integers. We denote the $$$y$$$-th element of the $$$x$$$-th array as $$$a_{x, y}$$$.

You have to choose two arrays $$$a_i$$$ and $$$a_j$$$ ($$$1 \le i, j \le n$$$, it is possible that $$$i = j$$$). After that, you will obtain a new array $$$b$$$ consisting of $$$m$$$ integers, such that for every $$$k \in [1, m]$$$ $$$b_k = \max(a_{i, k}, a_{j, k})$$$.

Your goal is to choose $$$i$$$ and $$$j$$$ so that the value of $$$\min \limits_{k = 1}^{m} b_k$$$ is maximum possible.

Input

The first line contains two integers $$$n$$$ and $$$m$$$ ($$$1 \le n \le 3 \cdot 10^5$$$, $$$1 \le m \le 8$$$) — the number of arrays and the number of elements in each array, respectively.

Then $$$n$$$ lines follow, the $$$x$$$-th line contains the array $$$a_x$$$ represented by $$$m$$$ integers $$$a_{x, 1}$$$, $$$a_{x, 2}$$$, ..., $$$a_{x, m}$$$ ($$$0 \le a_{x, y} \le 10^9$$$).

Output

Print two integers $$$i$$$ and $$$j$$$ ($$$1 \le i, j \le n$$$, **it is possible that $$$i = j$$$**) — the indices of the two arrays you have to choose so that the value of $$$\min \limits_{k = 1}^{m} b_k$$$ is maximum possible. If there are multiple answers, print any of them.

input

```
6 5
5 0 3 1 2
1 8 9 1 3
1 2 3 4 5
9 1 0 3 7
2 3 0 6 3
6 4 1 7 0
```

output

```
1 5
```

I. Flood Fill

2 seconds, 256 megabytes

You are given a line of $$$n$$$ colored squares in a row, numbered from $$$1$$$ to $$$n$$$ from left to right. The $$$i$$$-th square initially has the color $$$c_i$$$.

Let's say, that two squares $$$i$$$ and $$$j$$$ belong to the same connected component if $$$c_i = c_j$$$, and $$$c_i = c_k$$$ for all $$$k$$$ satisfying $$$i < k < j$$$. In other words, all squares on the segment from $$$i$$$ to $$$j$$$ should have the same color.

For example, the line $$$[3, 3, 3]$$$ has $$$1$$$ connected component, while the line $$$[5, 2, 4, 4]$$$ has $$$3$$$ connected components.

The game "flood fill" is played on the given line as follows:

- At the start of the game you pick any starting square (this is not counted as a turn).
- Then, in each game turn, change the color of the connected component containing the starting square to any other color.

Find the minimum number of turns needed for the entire line to be changed into a single color.

Input

The first line contains a single integer $$$n$$$ ($$$1 \le n \le 5000$$$) — the number of squares.

The second line contains integers $$$c_1, c_2, \ldots, c_n$$$ ($$$1 \le c_i \le 5000$$$) — the initial colors of the squares.

Output

Print a single integer — the minimum number of the turns needed.

input

```
4
5 2 2 1
```

output

```
2
```

input

```
8
4 5 2 2 1 3 5 5
```

output

```
4
```

input

```
1
4
```

output

```
0
```

In the first example, a possible way to achieve an optimal answer is to pick square with index $$$2$$$ as the starting square and then play as follows:

- $$$[5, 2, 2, 1]$$$
- $$$[5, 5, 5, 1]$$$
- $$$[1, 1, 1, 1]$$$

In the second example, a possible way to achieve an optimal answer is to pick square with index $$$5$$$ as the starting square and then perform recoloring into colors $$$2, 3, 5, 4$$$ in that order.

In the third example, the line already consists of one color only.

J. Office Keys

2 seconds, 256 megabytes

There are n people and k keys on a straight line. Every person wants to get to the office which is located on the line as well. To do that, he needs to reach some point with a key, take the key and then go to the office. Once a key is taken by somebody, it couldn't be taken by anybody else.

You are to determine the minimum time needed for all n people to get to the office with keys. Assume that people move a unit distance per 1 second. If two people reach a key at the same time, only one of them can take the key. A person can pass through a point with a key without taking it.

Input

The first line contains three integers n, k and p ($1 \le n \le 1\,000$, $n \le k \le 2\,000$, $1 \le p \le 10^9$) — the number of people, the number of keys and the office location.

The second line contains n distinct integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$) — positions in which people are located initially. The positions are given in arbitrary order.

The third line contains k distinct integers $b_1, b_2, ..., b_k$ ($1 \le b_j \le 10^9$) — positions of the keys. The positions are given in arbitrary order.

Note that there can't be more than one person or more than one key in the same point. A person and a key can be located in the same point.

Output

Print the minimum time (in seconds) needed for all n to reach the office with keys.

input

```
2 4 50
20 100
60 10 40 80
```

output

```
50
```

input

```
1 2 10
11
15 7
```

output

```
7
```

In the first example the person located at point 20 should take the key located at point 40 and go with it to the office located at point 50. He spends 30 seconds. The person located at point 100 can take the key located at point 80 and go to the office with it. He spends 50 seconds. Thus, after 50 seconds everybody is in office with keys.