A. Office Keys

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

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 $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — positions in which people are located initially. The positions are given in arbitrary order.

The third line contains k distinct integers $b_1, b_2, \ldots, b_k$ $(1 \le b_j \le 10^9)$ — 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.

Examples

input

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

output

```
50
```

input

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

output

```
7
```

Note

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.

B. Cards Sorting

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

Vasily has a deck of cards consisting of n cards. There is an integer on each of the cards, this integer is between 1 and $100\,000$, inclusive. It is possible that some cards have the same integers on them.

Vasily decided to sort the cards. To do this, he repeatedly takes the top card from the deck, and if the number on it equals the minimum number written on the cards in the deck, then he places the card away. Otherwise, he puts it under the deck and takes the next card from the top, and so on. The process ends as soon as there are no cards in the deck. You can assume that Vasily always knows the minimum number written on some card in the remaining deck, but doesn't know where this card (or these cards) is.

You are to determine the total number of times Vasily takes the top card from the deck.

Input

The first line contains single integer n $(1 \le n \le 100\,000)$ — the number of cards in the deck.

The second line contains a sequence of n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 100\,000)$, where a_i is the number written on the i-th from top card in the deck.

Output

Print the total number of times Vasily takes the top card from the deck.

Examples

input

```
4
6 3 1 2
```

output

```
7
```

input

```
1
1000
```

output

```
1
```

input

```
7
3 3 3 3 3 3 3
```

output

```
7
```

Note

In the first example Vasily at first looks at the card with number 6 on it, puts it under the deck, then on the card with number 3, puts it under the deck, and then on the card with number 1. He places away the card with 1, because the number written on it is the minimum among the remaining cards. After that the cards from top to bottom are $[2, 6, 3]$. Then Vasily looks at the top card with number 2 and puts it away. After that the cards from top to bottom are $[6, 3]$. Then Vasily looks at card 6, puts it under the deck, then at card 3 and puts it away. Then there is only one card with number 6 on it, and Vasily looks at it and puts it away. Thus, in total Vasily looks at 7 cards.

C. Bamboo Partition

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

Vladimir wants to modernize partitions in his office. To make the office more comfortable he decided to remove a partition and plant several bamboos in a row. He thinks it would be nice if there are n bamboos in a row, and the i-th from the left is a_i meters high.

Vladimir has just planted n bamboos in a row, each of which has height 0 meters right now, but they grow 1 meter each day. In order to make the partition nice Vladimir can cut each bamboo once at any height (no greater that the height of the bamboo), and then the bamboo will stop growing.

Vladimir wants to check the bamboos each d days (i.e. d days after he planted, then after $2d$ days and so on), and cut the bamboos that reached the required height. Vladimir wants the total length of bamboo parts he will cut off to be no greater than k meters.

What is the maximum value d he can choose so that he can achieve what he wants without cutting off more than k meters of bamboo?

Input

The first line contains two integers n and k ($1 \le n \le 100$, $1 \le k \le 10^{11}$) — the number of bamboos and the maximum total length of cut parts, in meters.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the required heights of bamboos, in meters.

Output

Print a single integer — the maximum value of d such that Vladimir can reach his goal.

Examples

input

```
3 4
1 3 5
```

output

```
3
```

input

```
3 40
10 30 50
```

output

```
32
```

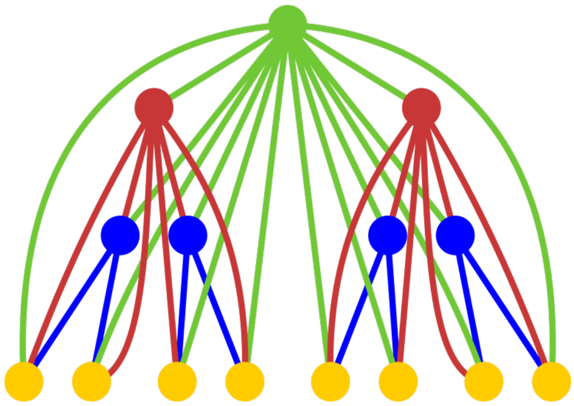
Note

In the first example Vladimir can check bamboos each 3 days. Then he will cut the first and the second bamboos after 3 days, and the third bamboo after 6 days. The total length of cut parts is $2 + 0 + 1 = 3$ meters.

D. Singer House

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

It is known that passages in Singer house are complex and intertwined. Let's define a Singer k-house as a graph built by the following process: take complete binary tree of height k and add edges from each vertex to all its successors, if they are not yet present.



Singer 4-house

Count the number of non-empty paths in Singer k-house which do not pass the same vertex twice. Two paths are distinct if the sets or the orders of visited vertices are different. Since the answer can be large, output it modulo $10^9 + 7$.

Input

The only line contains single integer k ($1 \le k \le 400$).

Output

Print single integer — the answer for the task modulo $10^9 + 7$.

Examples

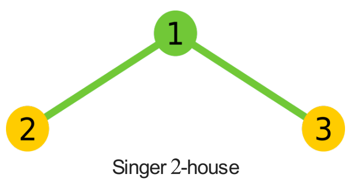
input
2
output
9

input
3
output
245

input
20
output
550384565

Note

There are 9 paths in the first example (the vertices are numbered on the picture below): `1`, `2`, `3`, `1-2`, `2-1`, `1-3`, `3-1`, `2-1-3`, `3-1-2`.



Singer 2-house

E. Perpetual Motion Machine

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

Developer Petr thinks that he invented a perpetual motion machine. Namely, he has a lot of *elements*, which work in the following way.

Each element has one controller that can be set to any non-negative real value. If a controller is set on some value x, then the controller consumes x^2 energy units per second. At the same time, any two elements connected by a wire produce $y \cdot z$ energy units per second, where y and z are the values set on their controllers.

Petr has only a limited number of wires, so he has already built some scheme of elements and wires, and is now interested if it's possible to set the controllers in such a way that the system produces **at least as much** power as it consumes, and at least one controller is set on the value different from 0. Help him check this, and if it's possible, find the required **integer** values that should be set.

It is guaranteed that if there exist controllers' settings satisfying the above conditions, then there exist required integer values not greater than 10^6.

Input

There are several (at least one) test cases in the input. The first line contains single integer — the number of test cases.

There is an empty line before each test case. The first line of test case contains two integers n and m ($1 \le n \le 10^5$, $0 \le m \le 10^5$) — the number of elements in the scheme and the number of wires.

After that, m lines follow, each of them contains two integers a and b ($1 \le a, b \le n$) — two elements connected by a wire. No element is connected with itself, no two elements are connected by more than one wire.

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

For hacks you can only use tests with one test case.

Output

Print answer for each test case.

For each test case print "YES" if it's possible to set the controllers in such a way that the consumed power is not greater than the power produced, and the required values on the next line. The settings should be integers from 0 to 10^6, inclusive, and at least one value should be different from 0. If there are multiple answers, print any of them.

If it's not possible to set the controllers in the required way, print one line "NO".

Example

input

```
4

4 4
1 2
2 3
3 4
4 2

3 2
2 3
3 1

4 6
1 2
3 4
4 2
1 4
1 3
3 2

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

output

```
YES
1 2 2 1
NO
YES
1 1 1 1
YES
1 5 1 1 1 1 1 1 1 1
```

Note

In the first example it's possible to set the controllers in the required way, for example, in the following way: set 1 on the first element, set 2 on the second and on the third, set 1 on the fourth. The consumed power is then equal to $1^2 + 2^2 + 2^2 + 1^2 = 10$ energy units per second, the produced power is equal to $1 \cdot 2 + 2 \cdot 2 + 2 \cdot 1 + 2 \cdot 1 = 10$ energy units per second. Thus the answer is "YES".

In the second test case it's not possible to set the controllers in the required way. For example, if we set all controllers to 0.5, then the consumed powers equals 0.75 energy units per second, while produced power equals 0.5 energy units per second.