Codeforces Round #594 (Div. 2)

A. Integer Points

2 seconds, 512 megabytes

DLS and JLS are bored with a Math lesson. In order to entertain themselves, DLS took a sheet of paper and drew n distinct lines, given by equations $y = x + p_i$ for some distinct $p_1, p_2, \ldots, p_n$.

Then JLS drew on the same paper sheet m distinct lines given by equations $y = -x + q_i$ for some distinct $q_1, q_2, \ldots, q_m$.

DLS and JLS are interested in counting how many line pairs have **integer** intersection points, i.e. points with both coordinates that are integers. Unfortunately, the lesson will end up soon, so DLS and JLS are asking for your help.

Input

The first line contains one integer t ($1 \le t \le 1000$), the number of test cases in the input. Then follow the test case descriptions.

The first line of a test case contains an integer n ($1 \le n \le 10^5$), the number of lines drawn by DLS.

The second line of a test case contains n distinct integers p_i ($0 \le p_i \le 10^9$) describing the lines drawn by DLS. The integer p_i describes a line given by the equation $y = x + p_i$.

The third line of a test case contains an integer m ($1 \le m \le 10^5$), the number of lines drawn by JLS.

The fourth line of a test case contains m distinct integers q_i ($0 \le q_i \le 10^9$) describing the lines drawn by JLS. The integer q_i describes a line given by the equation $y = -x + q_i$.

The sum of the values of n over all test cases in the input does not exceed 10^5. Similarly, the sum of the values of m over all test cases in the input does not exceed 10^5.

In hacks it is allowed to use only one test case in the input, so $t = 1$ should be satisfied.

Output

For each test case in the input print a single integer — the number of line pairs with integer intersection points.

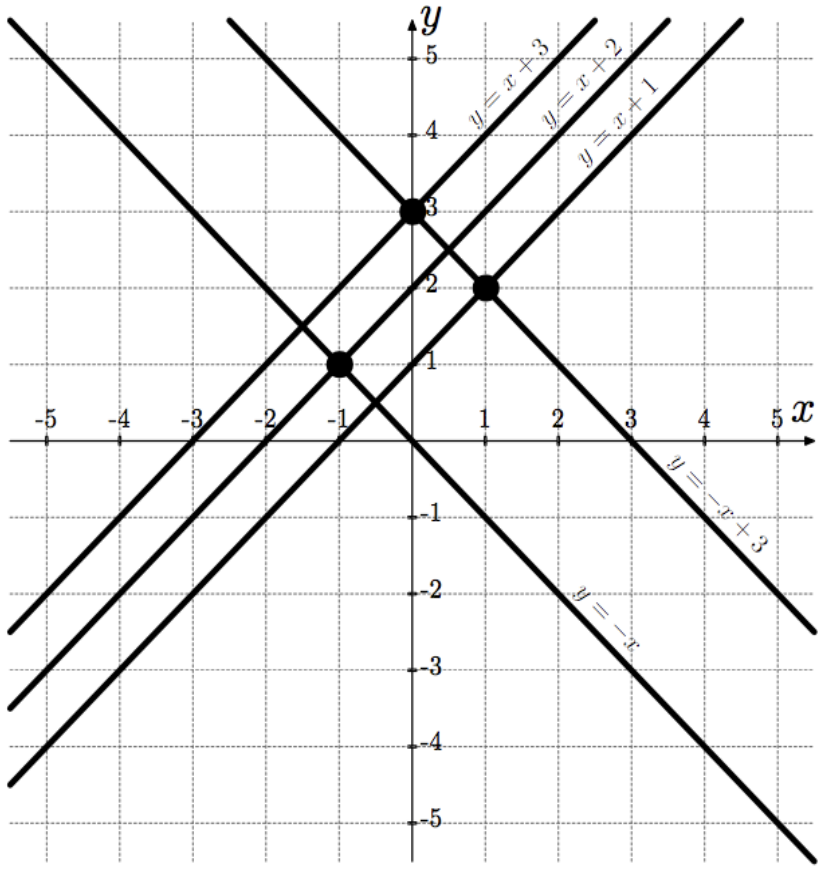
input

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

output

```
3
1
0
```

The picture shows the lines from the first test case of the example. Black circles denote intersection points with integer coordinates.



B. Grow The Tree

2 seconds, 512 megabytes

Gardener Alexey teaches competitive programming to high school students. To congratulate Alexey on the Teacher's Day, the students have gifted him a collection of wooden sticks, where every stick has an integer length. Now Alexey wants to grow a tree from them.

The tree looks like a polyline on the plane, consisting of all sticks. The polyline starts at the point $(0, 0)$. While constructing the polyline, Alexey will attach sticks to it one by one in arbitrary order. Each stick must be either vertical or horizontal (that is, parallel to OX or OY axis). It is not allowed for two consecutive sticks to be aligned simultaneously horizontally or simultaneously vertically. See the images below for clarification.

Alexey wants to make a polyline in such a way that its end is as far as possible from $(0, 0)$. Please help him to grow the tree this way.

Note that the polyline defining the form of the tree may have self-intersections and self-touches, but it can be proved that the optimal answer does not contain any self-intersections or self-touches.

Input

The first line contains an integer n ($1 \le n \le 100\,000$) — the number of sticks Alexey got as a present.

The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10\,000$) — the lengths of the sticks.

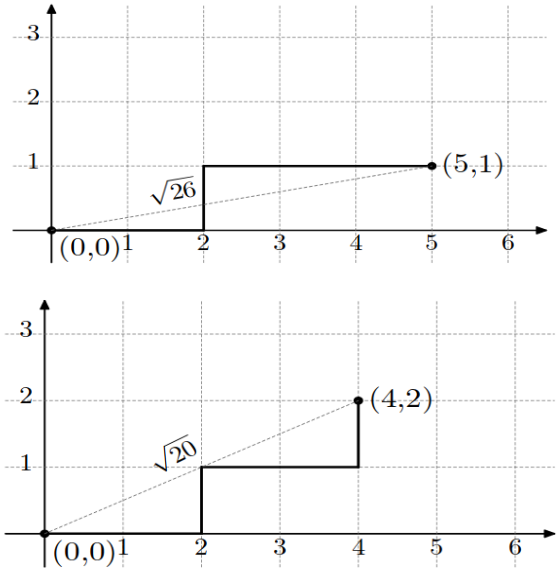
Output

Print one integer — the **square** of the largest possible distance from $(0, 0)$ to the tree end.

input
3 1 2 3
output
26

input
4 1 1 2 2
output
20

The following pictures show optimal trees for example tests. The squared distance in the first example equals $5 \cdot 5 + 1 \cdot 1 = 26$, and in the second example $4 \cdot 4 + 2 \cdot 2 = 20$.



C. Ivan the Fool and the Probability Theory

1 second, 512 megabytes

Recently Ivan the Fool decided to become smarter and study the probability theory. He thinks that he understands the subject fairly well, and so he began to behave like he already got PhD in that area.

To prove his skills, Ivan decided to demonstrate his friends a concept of random picture. A picture is a field of n rows and m columns, where each cell is either black or white. Ivan calls the picture random if for every cell it has **at most** one adjacent cell of the same color. Two cells are considered adjacent if they share a side.

Ivan's brothers spent some time trying to explain that it's not how the randomness usually works. Trying to convince Ivan, they want to count the number of different random (according to Ivan) pictures. Two pictures are considered different if at least one cell on those two picture is colored differently. Since the number of such pictures may be quite large, print it modulo $10^9 + 7$.

Input

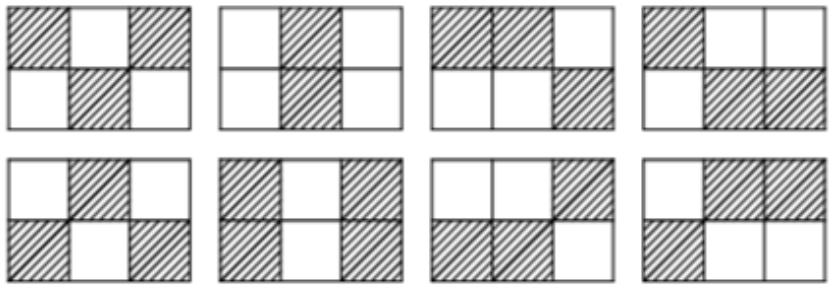
The only line contains two integers n and m ($1 \le n, m \le 100\,000$), the number of rows and the number of columns of the field.

Output

Print one integer, the number of random pictures modulo $10^9 + 7$.

input
2 3
output
8

The picture below shows all possible random pictures of size 2 by 3.



D1. The World Is Just a Programming Task (Easy Version)

1 second, 512 megabytes

This is an easier version of the problem. In this version, $n \le 500$.

Vasya is an experienced developer of programming competitions' problems. As all great minds at some time, Vasya faced a creative crisis. To improve the situation, Petya gifted him a string consisting of opening and closing brackets only. Petya believes, that the beauty of the bracket string is a number of its cyclical shifts, which form a correct bracket sequence.

To digress from his problems, Vasya decided to select two positions of the string (**not necessarily distinct**) and swap characters located at this positions with each other. Vasya will apply this operation exactly once. He is curious what is the maximum possible beauty he can achieve this way. Please help him.

We remind that bracket sequence s is called correct if:

- s is empty;
- s is equal to "(t)", where t is correct bracket sequence;
- s is equal to $t_1 t_2$, i.e. concatenation of t_1 and t_2, where t_1 and t_2 are correct bracket sequences.

For example, "(()())", "()" are correct, while ")(" and "())" are not.

The cyclical shift of the string s of length n by k ($0 \leq k < n$) is a string formed by a concatenation of the last k symbols of the string s with the first $n - k$ symbols of string s. For example, the cyclical shift of string "(())()" by 2 equals "()(())".

Cyclical shifts i and j are considered different, if $i \neq j$.

Input

The first line contains an integer n ($1 \leq n \leq 500$), the length of the string.

The second line contains a string, consisting of exactly n characters, where each of the characters is either "(" or ")".

Output

The first line should contain a single integer — the largest beauty of the string, which can be achieved by swapping some two characters.

The second line should contain integers l and r ($1 \leq l, r \leq n$) — the indices of two characters, which should be swapped in order to maximize the string's beauty.

In case there are several possible swaps, print any of them.

input
10 ()()())(()
output
5 8 7

input
12)(()(()())()
output
4 5 10

input
6))()()
output
0 1 1

In the first example, we can swap 7-th and 8-th character, obtaining a string "()()()()()()". The cyclical shifts by $0, 2, 4, 6, 8$ of this string form a correct bracket sequence.

In the second example, after swapping 5-th and 10-th character, we obtain a string ")(())()()(())". The cyclical shifts by $11, 7, 5, 3$ of this string form a correct bracket sequence.

In the third example, swap of any two brackets results in 0 cyclical shifts being correct bracket sequences.

D2. The World Is Just a Programming Task (Hard Version)

1 second, 512 megabytes

This is a harder version of the problem. In this version, $n \le 300\,000$.

Vasya is an experienced developer of programming competitions' problems. As all great minds at some time, Vasya faced a creative crisis. To improve the situation, Petya gifted him a string consisting of opening and closing brackets only. Petya believes, that the beauty of the bracket string is a number of its cyclical shifts, which form a correct bracket sequence.

To digress from his problems, Vasya decided to select two positions of the string (**not necessarily distinct**) and swap characters located at this positions with each other. Vasya will apply this operation exactly once. He is curious what is the maximum possible beauty he can achieve this way. Please help him.

We remind that bracket sequence s is called correct if:

- s is empty;
- s is equal to "(t)", where t is correct bracket sequence;
- s is equal to $t_1 t_2$, i.e. concatenation of t_1 and t_2, where t_1 and t_2 are correct bracket sequences.

For example, "(()())", "()" are correct, while ")(" and "())" are not.

The cyclical shift of the string s of length n by k ($0 \le k < n$) is a string formed by a concatenation of the last k symbols of the string s with the first $n - k$ symbols of string s. For example, the cyclical shift of string "(())()" by 2 equals "()(())".

Cyclical shifts i and j are considered different, if $i \ne j$.

Input

The first line contains an integer n ($1 \le n \le 300\,000$), the length of the string.

The second line contains a string, consisting of exactly n characters, where each of the characters is either "(" or ")".

Output

The first line should contain a single integer — the largest beauty of the string, which can be achieved by swapping some two characters.

The second line should contain integers l and r ($1 \le l, r \le n$) — the indices of two characters, which should be swapped in order to maximize the string's beauty.

In case there are several possible swaps, print any of them.

input
10 ()()())(()
output
5 8 7

input
12)(()(()())()
output
4 5 10

input
6))()()
output
0 1 1

In the first example, we can swap 7-th and 8-th character, obtaining a string "()()()()()". The cyclical shifts by $0, 2, 4, 6, 8$ of this string form a correct bracket sequence.

In the second example, after swapping 5-th and 10-th character, we obtain a string ")(())()()(()". The cyclical shifts by $11, 7, 5, 3$ of this string form a correct bracket sequence.

In the third example, swap of any two brackets results in 0 cyclical shifts being correct bracket sequences.

E. Queue in the Train

1 second, 512 megabytes

There are n seats in the train's car and there is exactly one passenger occupying every seat. The seats are numbered from 1 to n from left to right. The trip is long, so each passenger will become hungry at some moment of time and will go to take boiled water for his noodles. The person at seat i ($1 \le i \le n$) will decide to go for boiled water at minute t_i.

Tank with a boiled water is located to the left of the 1-st seat. In case too many passengers will go for boiled water simultaneously, they will form a queue, since there can be only one passenger using the tank at each particular moment of time. Each passenger uses the tank for exactly p minutes. We assume that the time it takes passengers to go from their seat to the tank is negligibly small.

Nobody likes to stand in a queue. So when the passenger occupying the i-th seat wants to go for a boiled water, he will first take a look on all seats from 1 to $i - 1$. In case at least one of those seats is empty, he assumes that those people are standing in a queue right now, so he would be better seating for the time being. However, at the very first moment he observes that all seats with numbers smaller than i are busy, he will go to the tank.

There is an unspoken rule, that in case at some moment several people can go to the tank, than only the leftmost of them (that is, seating on the seat with smallest number) will go to the tank, while all others will wait for the next moment.

Your goal is to find for each passenger, when he will receive the boiled water for his noodles.

Input

The first line contains integers n and p ($1 \le n \le 100\,000$, $1 \le p \le 10^9$) — the number of people and the amount of time one person uses the tank.

The second line contains n integers $t_1, t_2, \ldots, t_n$ ($0 \le t_i \le 10^9$) — the moments when the corresponding passenger will go for the boiled water.

Output

Print n integers, where i-th of them is the time moment the passenger on i-th seat will receive his boiled water.

input
5 314 0 310 942 628 0
output
314 628 1256 942 1570

Consider the example.

At the 0-th minute there were two passengers willing to go for a water, passenger 1 and 5, so the first passenger has gone first, and returned at the 314-th minute. At this moment the passenger 2 was already willing to go for the water, so the passenger 2 has gone next, and so on. In the end, 5-th passenger was last to receive the boiled water.

F. Catowice City

2 seconds, 512 megabytes

In the Catowice city next weekend the cat contest will be held. However, the jury members and the contestants haven't been selected yet. There are n residents and n cats in the Catowice, and each resident has exactly one cat living in his house. The residents and cats are numbered with integers from 1 to n, where the i-th cat is living in the house of i-th resident.

Each Catowice resident is in friendship with several cats, including the one living in his house. In order to conduct a contest, at least one jury member is needed and at least one cat contestant is needed. Of course, every jury member should know none of the contestants. For the contest to be successful, it's also needed that the number of jury members plus the number of contestants is equal to n.

Please help Catowice residents to select the jury and the contestants for the upcoming competition, or determine that it's impossible to do.

Input

The first line contains an integer t ($1 \le t \le 100\,000$), the number of test cases. Then description of t test cases follow, where each description is as follows:

The first line contains integers n and m ($1 \le n \le m \le 10^6$), the number of Catowice residents and the number of friendship pairs between residents and cats.

Each of the next m lines contains integers a_i and b_i ($1 \le a_i, b_i \le n$), denoting that a_i-th resident is acquaintances with b_i-th cat. It's guaranteed that each pair of some resident and some cat is listed at most once.

It's guaranteed, that for every i there exists a pair between i-th resident and i-th cat.

Different test cases are separated with an empty line.

It's guaranteed, that the sum of n over all test cases is at most 10^6 and that the sum of m over all test cases is at most 10^6.

Output

For every test case print:

- "No", if it's impossible to select the jury and contestants.
- Otherwise print "Yes".
  In the second line print two integers j and p ($1 \le j$, $1 \le p$, $j + p = n$) — the number of jury members and the number of contest participants.

  In the third line print j distinct integers from 1 to n, the indices of the residents forming a jury.

  In the fourth line print p distinct integers from 1 to n, the indices of the cats, which will participate in the contest.

  In case there are several correct answers, print any of them.

input

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

3 7
1 1
1 2
1 3
2 2
3 1
3 2
3 3

1 1
1 1

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

output

```
Yes
2 1
1 3
2
Yes
1 2
2
1 3
No
No
```

In the first test case, we can select the first and the third resident as a jury. Both of them are not acquaintances with a second cat, so we can select it as a contestant.

In the second test case, we can select the second resident as a jury. He is not an acquaintances with a first and a third cat, so they can be selected as contestants.

In the third test case, the only resident is acquaintances with the only cat, so they can't be in the contest together. So it's not possible to make a contest with at least one jury and at least one cat.

In the fourth test case, each resident is acquaintances with every cat, so it's again not possible to make a contest with at least one jury and at least one cat.
