DATA STRUCTURES AND ALGORITHMS

ASSIGNMENT-5
Language allowed: C

February 5, 2020

A. Trees and Dusty Cupboards

While cleaning your room, you find a large undirected rooted tree (rooted at 0) in your cupboard. You are now intrigued by it and wish the *maximum* distance between any two nodes. Write a simple program that will aid you solve this problem.

Input

The first line of input contains a single integer N ($2 \leq$ N ≤ 1500) denoting the number of vertices in the tree. Each of the following N-1 lines contain two space-separated integers X_i and Y_i ($0 \leq X_i, Y_i \leq$ N-1) denoting an undirected edge between the vertices X_i and Y_i.

Output

In the first line, print a single integer L, denoting the maximum distance between any two nodes. In the second line, print two space-separated integers U_i and V_i ($U_i < V_i$) denoting the nodes that are L units apart. If multiple such pairs are present, print the pair with that has minimum U_i. If U_i values are also same, print the pair with maximum V_i.

---

input
```
14
0 1
0 2
0 3
4 1
5 1
4 6
4 12
6 10
6 11
13 9
9 2
8 7
7 3
```

output
```
7
8 11
```

B. Last Question

After an outstanding performance of your class 3E in the final exams, board director Gakuho Asuno has decided to award you all special prizes. But before the ceremony, he has one last question for you all. He gives you all an array of integers and asks you to find the sum of the *bit-differences* of every *ordered-pair* of integers in it. *Bit-difference* of two integers X and Y is the number of positions at which bits of X and Y differ (If required, you may have to pad zeroes on the left of the binary representation of smaller number of the pair). He challenges you all to solve this problem as soon as possible. Finally, it's up to you, Nagisa, to keep up Koro-sensei's name and solve this problem. This question is expected to be solved in *O(N)*.

Input

The first line of input contains a single integer N ($2 \leq N \leq 3.10^5$) denoting the number of elements in the array A. The next line contains N space-separated integers ($0 \leq A_i \leq 2^{32}$) denoting the array itself.

Output

Output a single integer M, denoting the sum of *bit-differences* of every *ordered-pair* of integers in that given array. As the sum can be very large, print it to modulo ($10^9 + 7$);

---

input
```
10
68764 8 12 964 199 49965 9410 100000 1685 357
```

output
```
596
```

C. Fruit Loops

Howard "Fruit Loops" Wolowitz is getting ready for his first spacewalk outside the ISS. The pieces of space-suit he needs to wear is numbered from 1 serially. He needs to wear the pieces of the suit in a certain order, i.e, he can wear a certain piece (say the helmet) only after he has worn some other given piece (say the upper torso). Help Wolowitz get the right order in which he should wear the suit.

Input

The first line of input contains two space-separated integers N ($1 \leq N \leq 1000$) and M ($0 \leq M \leq \frac{N(N-1)}{2}$) denoting the number of pieces in the suit and number of dependencies respectively (A dependency indicates the order in which two given pieces of the suit can be worn). Each of the next M lines contains two space-separated integers X_i and Y_i ($1 \leq X_i$, Y_i $\leq$ N) denoting that the piece X_i can only be worn after the piece Y_i is worn. It is guaranteed that there is atleast one piece X_i that can be worn at any time, i.e., does not require any piece to be worn before it.

Output

Print a sequence of N integers, with the i^{th} integer denoting the piece that has to be worn at the i^{th} instant. If there are multiple answers, print any. If there is a cyclic dependency in the sequence in which the pieces should be worn, then print "NOT POSSIBLE".

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

output
1 4 5 3 2
```

D. Find those triples!

In this question, you are given a sorted array A containing unique elements. Your task is to find the number of groups of three elements (also termed as triples) such that the difference between the minimum and maximum of each triple is *atmost* K.

Input

The first line of input contains two space-separated integers N ($1 \leq N \leq 10^5$) and K ($1 \leq K \leq 10^9$) denoting the number of elements in the array and the given number K. The last line contains N space-separated integers ($-10^9 \leq A_i \leq 10^9$) denoting the sorted array A.

Output

Print a single integer M, denoting the number of triples.

```
input
5 19
1 10 20 30 50

output
1
```

```
input
4 3
1 2 3 4

output
4
```

E. Okabe and the Toll Gates

The cities in Japan lie on a straight line numbered from 0 serially and adjacent cities are unit distance apart. Okabe Rintaro lives in city U has been invited to give a talk on his time-machine theory in city V. He plans to rent a car from his city to the destination. The road from city U to V has a few toll gates. Each toll gate has a gas station. All the gas stations surprisingly sell gas in fixed capacity containers (in litres). A litre of gas costs K yen. The car Okabe rents runs according to the following mileage: *Z litres of gas lets him drive AZ+B units.* On his drive, Okabe plans to stop at every *immediate* toll gate (not anywhere in between), empty any gas currently in the tank and refill it again from the gas station at the toll gate. Help Okabe spend as *minimum* money as possible on the gas. It is guaranteed that cities U and V will always have tolls gates. Assume that the car had no gas before Okabe rented it. Note that, the use of inbuilt *qsort* function cannot be made to solve this problem.

Input

The first line of input contains three space-separated N ($1 \leq$ N $\leq 10^5$), M ($1 \leq$ M $\leq$ N) and L ($1 \leq$ L $\leq 10^5$) denoting the number of cities in Japan, the number of cities that have toll gates and the number of different gas containers sold at each gas station. The next line contains five space-separated integers U ($0 \leq$ U $\leq$ N - 1), V (U $\leq$ V $\leq$ N - 1), A, B ($1 \leq$ A, B $\leq 10^3$) and K ($1 \leq$ K $\leq 10^3$), denoting the starting and destination city, mileage coefficients and the rate of gas respectively. The following line has M space-separated integers ($0 \leq T_i \leq$ N - 1) denoting the cities that have toll gates. The last line of input contains L space-separated integers ($1 \leq C_i \leq 10^6$) denoting the various gas container sizes sold in the gas stations.

Output

Print a single integer P, denoting the *minimum* money Okabe should spend on gas. If Okabe cannot make it to the city V using the above strategy, print "NOT POSSIBLE".

```
input
100 15 7
11 92 2 3 10
52 81 76 36 5 23 50 90 17 46 3 82 11 92 83
1 8 11 7 2 5 9

output
330
```

F. Lab Evals

After the evaluation of the fourth assignment of CS F211, the TA wished to distribute sweets to the students based on their marks. However, he did not know how many sweets to buy. He wanted it such that the student with strictly higher (and not equal) marks should get more sweets than his/her peer(s) with immediate lesser marks. However, he also wanted to make sure that each student got at least one sweet and he buys the least number of sweets. Unfortunately, the TA could not solve such a simple problem. Since you are a student of DSA, help your incompetent TA so that you get your sweet. Also, assume that a single sweet is not shared among multiple students.
Note that, the use of inbuilt *qsort* function cannot be made to solve this problem.

Input

The first line of input contains a single integer N ($1 \leq \text{N} \leq 10^6$) denoting the number of students registered to the course. The next line contains N space-separated integers ($0 \leq M_i \leq 10^9$) denoting the marks of the students (the i^{th} integer denotes the marks of the i^{th} student).

Output

Print a single number X, denoting the number of sweets the TA should purchase.

```
input
4
1 5 2 1

output
7

explanation
The TA can distribute the sweets as 1, 3, 2, 1 respectively.
```

G. Super Bit String

A bit string is a string of bits that have values of either 1 or 0. A super bit string is a string made by flipping none or at least one of the 0s in the bit string to 1. Given a decimal integer K, convert it to a bit string that has a given length N (padding with 0s to the left if necessary) and then generate all possible super bit strings of the number.
Note that, the use of inbuilt *qsort* function cannot be made to solve this problem.

Input

The first line of input is the integer N ($1 \leq N \leq 16$). The second line of input would contain the integer K ($0 \leq K < 2^N$).

Output

Print a sequence of integers (in their binary representation) denoting the list after sorting it. Print all the super bit strings sorted in ascending order separated by a newline.

---

input

```
3
2
```

output

```
010
011
110
111
```

H. Cakes and Parties

It's your best friend's birthday party and you have ordered a cake from Swiss Palace. The shop keeper has prepared the order and set it aside along with the other orders for the day in the form of a 2D array and now the owner is finding it difficult to locate your cake order as all orders have been packed using the same type of carton. All that you know is the price of the cake written on the receipt of the carton. You notice that all the prices of all the orders are *unique* and are *sorted row-wise and column-wise in ascending order*. With this information, help the shopkeeper find the your friend's cake as soon as possible. You are expected to solve this in $o(NM)$ time.

Input

The first line contains two space-separated integers N and M ($1 \leq$ N, M $\leq 10^4$) denoting the number of rows and columns of the 2D array P respectively. The following N lines contain M space-separated integers ($1 \leq P_{ij} \leq 10^9$) denoting the prices of the orders in the 2D array. The last line contains a single integer Z ($1 \leq$ Z $\leq 10^9$) denoting the price of your friend's cake.

Output

If the cake is present in the 2D array, print "YES" in the first line and in the second line print the coordinates of the cake (0-indexed). If the cake is missing, print "MISSING".

```
input
4 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
10

output
YES
2 1
```

I. Lost Pair

In this task, you are given an array A. You now need to find two elements in the given array whose sum is equal to a given value K. Note that each array element can be used only once for constructing the pair. Write a simple program to do the same.

Input

The first line of input contains a single integer N ($1 \leq N \leq 2.10^5$), denoting the number of elements in the array. The next line contains N space-separated integers ($-10^9 \leq A_i \leq 10^9$) denoting the array A. The last line contains a single integer K ($-2.10^9 \leq A_i \leq 2.10^9$) denoting the sum that has to be found.

Output

If a pair of integers exist whose sum is equal to the given value K, print "YES" in the first line followed by the pair of integers in ascending order in the next line. If multiple such pairs exist, print the pair that has *minimum* difference between its elements. If no pair exists, print "NO".

```
input
6
-3 7 6 4 8 10
14
```

```
output
YES
6 8
```

J. Sorted Rotations

After carefully sorting a large array painstakingly, you decide to stretch a bit. By the time you come back, you realize that your naughty little sister has rotated the sorted array by some unknown number of steps. You have an epiphany that it might not be extremely hard to search numbers in this rotated array. You further ease the task you decide to write a program to do the same. This problem is expected to be solved in $O(\log N)$, N being the size of the array. It is guaranteed that all the elements of the array will be unique. Assume that the array was sorted in ascending order initially.

Input

The first line of input contains a single integer N ($1 \leq N \leq 3.10^6$) denoting the number of elements in the array A. The next line contains N space-separated integers ($-10^9 \leq A_i \leq 10^9$) denoting the elements of the array. The last line of input contains a single integer Z ($-10^9 \leq Z \leq 10^9$) denoting the element that has to be found.

Output

If the element being queried is found, print "YES" in the first line, followed by the index of the element (0-indexed) in the next line. If the element is not present in the given array, print "NO".

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

output
YES
1
```

```
input
12
2 3 14 16 201 204 1000 -101 -25 -14 0 1
-102

output
NO
```
