Problem Set

Please check that you have 12 problems that are spanned across 32 pages in total (including Korean translation and this cover page).

A. Ball Alignment	(2 pages)	
B. Bit String	(1 page)	
C. Broken Tiles	(2 pages)	
D. Coronavirus Trend	(2 pages)	
E. Cycle Game	(2+2 pages)	Korean translation available
F. Escaping	(2+2 pages)	Korean translation available
G. Hotspots	(2 pages)	
H. Negative Cycle	(2 pages)	
I. Project Teams	(1+2 pages)	Korean translation available
J. Trading System	(1 page)	
K. Road Reconstruction	(2+2 pages)	Korean translation available
L. Sliding Coins	(2+2 pages)	Korean translation available

Problem A
Ball Alignment
Time Limit: 1 Second

Balls are placed on a curved structure as shown in Figure A.1, and each ball has a number written on it. Due to the law of gravity, all balls naturally gather near the lowest spot of the structure. The permitted action to change the position of the balls is to "pick a ball, lift it up, and place it on the far left or on the far right". This action will naturally change the location of some balls due to the law of gravity.

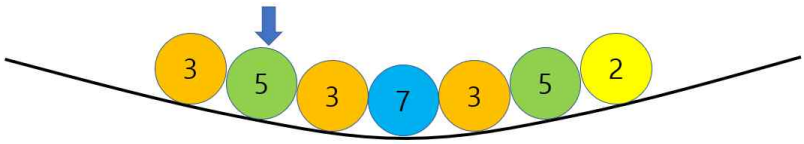


Figure A.1

Figure A.2 shows how the locations of the balls change when the marked ball 5 was picked up in Figure A.1 and placed on the far right.

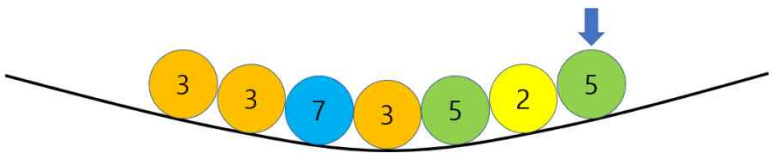


Figure A.2

When n balls are arranged on a curved structure we want to sort the numbers written on the balls in non-decreasing order by repeating the permitted actions. In Figure A.2, if ball 7 is picked up and placed on the far right and ball 2 far left, all the numbers on balls are sorted in non-decreasing order.

Given an arrangement of n balls, write a program which helps to rearrange all the numbers written on the balls in non-decreasing order by selecting minimum number of balls and relocating them through the permitted actions.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 10^5$), which denotes the number of balls. Each of the following n lines contains an integer v ($1 \leq v \leq 10^9$), which is the number written on a ball. Notice that some balls may have the same number written on them.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the minimum number of balls picked up and relocated to align all the numbers written on the n balls in non-decreasing order.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
4 21 44 39 41	1

Sample Input 2	Output for the Sample Input 2
7 11 12 25 16 11 18 25	2

Problem B
Bit String
Time Limit: 1.5 Seconds

A finite sequence of 0 and 1 is called a *bit string*. The length of a bit string w is the number of symbols contained in w. Note that the empty string is also a bit string of length zero. A substring of a bit string w is a consecutive portion of w. Note that the empty string is a substring of any bit string and any bit string is a substring of itself. For example, consider a bit string 1010 of length four. All substrings of 1010 are listed as follows: 1010, 101, 010, 10, 01, 1, 0, and the empty string. Bit strings 100 and 11 are not substrings of 1010. If a bit string P is a substring of a bit string w, then we say that w contains P as a substring or w contains P, shortly.

Consider all bit strings of length n. It is easy to see that there are exactly 2^n bit strings of length n in total. Given a nonnegative integer n and two bit strings P_1 and P_2, write a program that outputs the number of bit strings of length n that contain P_1 but do not contain P_2.

Input
Your program is to read from standard input. The input consists of three lines. The first line consists of three integers, n, k_1, and k_2 ($0 \le n \le 100{,}000$, $0 \le k_1, k_2 \le 10{,}000$), separated by a space. The second line consists of a single bit string, representing P_1 whose length is k_1. If $k_1 = 0$, then the second line is empty. The third line consists of a single bit string, representing P_2 whose length is k_2. If $k_2 = 0$, then the third line is empty. The three input numbers n, k_1, and k_2 satisfy the following conditions: If $0 < k_1 \le n$, then the product of n and k_1 does not exceed 10^7; if $0 < k_2 \le n$, then the product of n and k_2 does not exceed 10^7; if $0 < k_1 \le n$ and $0 < k_2 \le n$, then the product of n, k_1, and k_2 does not exceed 10^7.

Output
Your program is to write to standard output. Print exactly one line. The line should contain an integer r, where $0 \le r \le 1{,}000{,}000{,}006$ is the remainder when dividing by 1,000,000,007 the number of bit strings of length n that contain P_1 but do not contain P_2.

The following shows sample input and output for two test cases.

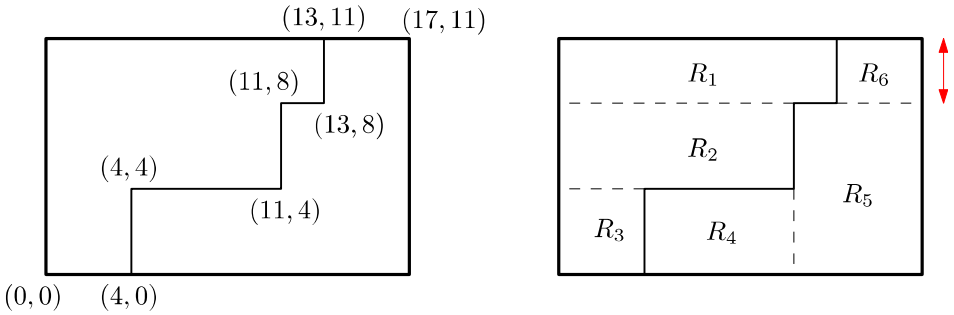
Sample Input 1	Output for the Sample Input 1
4 2 2 10 11	6

Sample Input 2	Output for the Sample Input 2
4 2 3 10 100	7

Problem C
Broken Tiles
Time Limit: 0.5 Seconds

A rectangular tile fell to the ground and broke into two pieces. Fortunately, the tile was broken along a rectilinear curve monotone to the sides of the tile as shown in the left figure below. A rectilinear curve is composed of line segments. A rectilinear curve R is monotone to a straight line or line segment L if every straight line orthogonal to L intersects R at most once.

Since the tiles must be rectangular, we decided to cut each piece further using a cutter into rectangular tiles along lines through the sides of the piece. We call such a cut *a rectangular cut* of the pieces. Every rectangular cut partitions the two pieces into rectangles. Among all possible rectangular cuts of the pieces, we want to find a rectangular cut that maximizes the minimum side-length of the resulting rectangles of the pieces.



For example, the left figure above shows an axis-aligned rectangle of side lengths 17 and 11. It is broken into two pieces along the monotone rectilinear curve connecting points (4,0), (4,4), (11,4), (11,8), (13,8), (13,11) by line segments in order. In a rectangular cut, you are allowed to cut each piece along the horizontal or vertical line or both from a corner vertex at (4,4), (11,4), (11,8) or (13,8). The right figure shows a rectangular cut. For the left piece, we choose horizontal lines at (4,4) and (11,8). For the right piece, we choose a horizontal line at (13,8) and a vertical line at (11,4). Thus, this rectangular cut partitions the two pieces into 6 rectangles $R_1, R_2, \ldots, R_6$. Since R_1 and R_6 have the minimum side-length 3 among the 6 rectangles, this rectangular cut has the minimum side-length 3. Observe that this rectangular cut maximizes the minimum side-length among all possible rectangular cuts on the broken pieces.

Given an axis-aligned rectangle and a list of n points representing the monotone rectilinear curve breaking the rectangle into two pieces, write a program to output the maximum of the minimum side-lengths among all possible rectangular cuts.

Input
Your program is to read from standard input. The input starts with a line containing two integers, a and b ($1 \le a \le 100{,}000$, $1 \le b \le 100{,}000$), where a and b are the x-coordinate and the y-coordinate of the top-right corner of the axis-aligned rectangle with bottom-left corner at (0,0). There is no input with $a = b = 1$. The next line contains one integer n ($2 \le n \le 100{,}000$), where n is the number of points of the monotone rectilinear curve breaking the rectangle into two pieces. In the following n lines, the i-th line contains two

integers, c and d ($0 \leq c \leq a$, $0 \leq d \leq b$), where c and d are the x-coordinate and the y-coordinate of the i-th point along the monotone rectilinear curve. You may assume that all the points are distinct.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the maximum of the minimum side-lengths among all possible rectangular cuts.

The following shows sample input and output for three test cases.

Sample Input 1

```
3 4
2
1 0
1 4
```

Output for the Sample Input 1

```
1
```

Sample Input 2

```
4 5
4
4 2
2 2
2 3
0 3
```

Output for the Sample Input 2

```
2
```

Sample Input 3

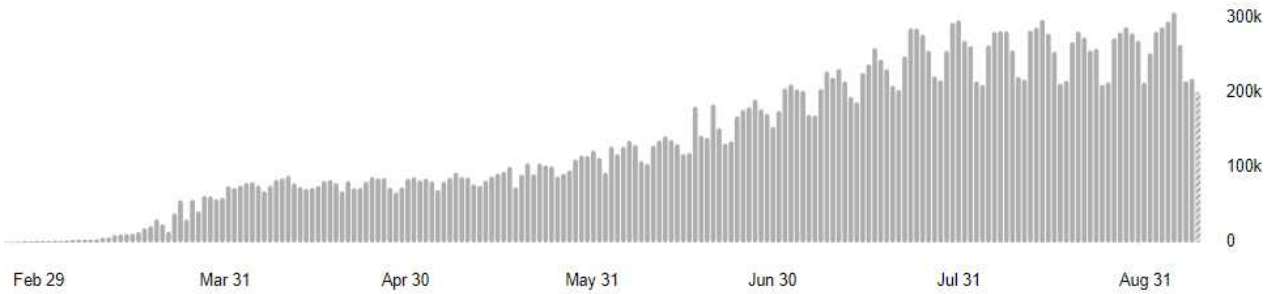
```
17 11
6
4 0
4 4
11 4
11 8
13 8
13 11
```

Output for the Sample Input 3

```
3
```

Problem D
Coronavirus Trend
Time Limit: 2 Seconds

Finally, the pandemic comes to an end and scientists start to study the cause and process of it. Specially, a group of scientists is paying attention to the world's daily record of confirmed cases. They have a sequence Q of n daily confirmed cases. It is assumed that all n positive integers in the sequence Q are different. The figure below is a graph showing the actual pandemic situation that occurred during a specific period.



They are interested in the pattern of the repetition of successive increases and successive decreases in the sequence of confirmed cases while observing the spread of the virus. A *run* in the sequence of distinct numbers is a maximal contiguous subsequence that is either increasing or decreasing. For example, the sequence (8,5,1,3,4,7,6,2) consists of three runs (8,5,1), (1,3,4,7), and (7,6,2), which have lengths 3, 4, and 3, respectively. The sequence (8,5,1,7,6,2,3,4) consists of four runs (8,5,1), (1,7), (7,6,2), and (2,3,4).

A sequence of distinct numbers whose every run has a length of at least 3 is said to be an *UD-sequence*. The sequence (8,5,1,3,4,7,6,2) is an UD-sequence, but (8,5,1,7,6,2,3,4) is not an UD-sequence because it has a run (1,7) with length 2.

Scientists believe that a run with length less than 3 is insufficient to reflect the trend of confirmed cases, so they try to find a longest UD-sequence which is a subsequence of Q. The sequence (8,5,1,3,4,7,6,2) is itself the longest UD-sequence, and the longest UD-sequence of (8,5,1,7,6,2,3,4) is (8,5,1,2,3,4) with length 6.

Write a program to find a longest UD-sequence which is a subsequence of a given sequence Q of n daily confirmed cases in order to help them.

Input
Your program is to read from standard input. The input starts with a line containing one integer n ($3 \leq n \leq 500{,}000$), where n is the number of days observed. The second line contains n positive integers, separated by a space, representing n daily confirmed cases, where they are all different and no more than 10^9.

Output

Your program is to write to standard output. Print exactly one line. The line should contain an integer which represents the length of a longest UD-sequence, as a subsequence of the given input. If the input sequence has no UD-sequence as its subsequence, print `0`.

The following shows sample input and output for three test cases.

Sample Input 1	**Output for the Sample Input 1**
8 8 5 1 7 6 2 3 4	6

Sample Input 2	**Output for the Sample Input 2**
3 1 3 2	0

Sample Input 3	**Output for the Sample Input 3**
8 1 9 5 11 3 24 18 35	5

Problem E
Cycle Game
Time Limit: 1 Second

Cycle game is a game in which two players take turns drawing a line segment for given points. The first player gets to draw a line segment in the odd-numbered turns, while the other player takes even-numbered turns. At the start of the game, n points are given in the plane, and each of the points has a unique number from 0 to $n-1$. Note that no three points lie on a same line. In each turn, a player draws a line segment connecting two given points. Players cannot redraw a line segment which has already been drawn, but it is allowed to draw a line segment across existing line segments. Players take turns until there is the loser who completes a cycle first. A cycle C is a subset of the line segments drawn by the players which satisfies the following condition.

From any endpoint of a line segment in C, one can travel down and return to the end point while visiting every line segment in C exactly once.

A problem is that as players draw many segments, it is difficult to check whether a cycle is created. Therefore, sometimes players continue to play the game even though a cycle has been made and the game is actually over. To prevent this situation, we need to write a program which examines whether the game has been over.

Write a program to decide whether the game has been over and output in which turn the first cycle has been made.

Input
Your program is to read from standard input. The input starts with a line containing two integers $3 \le n \le 500{,}000$ and $3 \le m \le 1{,}000{,}000$, where n represents the number of points and m represents the number of turns in which line segments have been drawn. Every point has a unique number from 0 to $n-1$, and no three points lie on a same line. Each of the following m lines contains two numbers of points which are the end points of the line segments each player drew in the i-th turn $(1 \le i \le m)$.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number of the turn in which the cycle has been first created, i.e., the game has been over, and 0 if the game was not over for m turns.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
6 5 0 1 1 2 2 3 5 4 0 4	0

Sample Input 2	Output for the Sample Input 2
6 5 0 1 1 2 1 3 0 3 4 5	4

Problem E
사이클 게임

제한 시간: 1 초

사이클 게임은 두 명의 플레이어가 차례대로 돌아가며 진행하는 게임으로, 선 플레이어가 홀수 번째 차례를, 후 플레이어가 짝수 번째 차례를 진행한다. 게임 시작 시 0 부터 $n-1$ 까지 고유한 번호가 부여된 평면 상의 점 n 개가 주어지며, 이 중 어느 세 점도 일직선 위에 놓이지 않는다. 매 차례 마다 플레이어는 두 점을 선택해서 이를 연결하는 선분을 긋는데, 이전에 그린 선분을 다시 그을 수는 없지만 이미 그린 다른 선분과 교차하는 것은 가능하다. 게임을 진행하다가 처음으로 사이클을 완성하는 순간 게임이 종료된다. 사이클 C는 플레이어가 그린 선분들의 부분집합으로, 다음 조건을 만족한다.

C에 속한 임의의 선분의 한 끝점에서 출발하여 모든 선분을 한 번씩만 지나서 출발점으로 되돌아올 수 있다.

문제는 선분을 여러 개 그리다 보면 사이클이 완성 되었는지의 여부를 판단하기 어려워 이미 사이클이 완성되었음에도 불구하고 게임을 계속 진행하게 될 수 있다는 것이다. 이 문제를 해결하기 위해서 게임의 진행 상황이 주어지면 몇 번째 차례에서 사이클이 완성되었는지, 혹은 아직 게임이 진행 중인지를 판단하는 프로그램을 작성하려 한다.

입력으로 점의 개수 n과 m 번째 차례까지의 게임 진행 상황이 주어지면 사이클이 완성 되었는지를 판단하고, 완성되었다면 몇 번째 차례에서 처음으로 사이클이 완성된 것인지를 출력하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 입력의 첫 번째 줄에는 점의 개수를 나타내는 정수 $3 \le n \le 500{,}000$과 진행된 차례의 수를 나타내는 정수 $3 \le m \le 1{,}000{,}000$ 이 주어진다. 게임에서 사용하는 n개의 점에는 0 부터 $n-1$ 까지 고유한 번호가 부여되어 있으며, 이 중 어느 세 점도 일직선 위에 놓이지

않는다. 이어지는 m 개의 입력 줄에는 각각 i번째 차례에 해당 플레이어가 선택한 두 점의 번호가 주어진다 $(1 \le i \le m)$.

Output

출력은 표준출력을 사용한다. 입력으로 주어진 케이스에 대해, m 번째 차례까지 게임을 진행한 상황에서 이미 게임이 종료되었다면 사이클이 처음으로 만들어진 차례의 번호를 양의 정수로 출력하고, m 번의 차례를 모두 처리한 이후에도 종료되지 않았다면 `0` 을 출력한다.

다음은 두 테스트 경우에 대한 입출력 예이다.

Sample Input 1	**Output for the Sample Input 1**
6 5 0 1 1 2 2 3 5 4 0 4	0

Sample Input 2	**Output for the Sample Input 2**
6 5 0 1 1 2 1 3 0 3 4 5	4

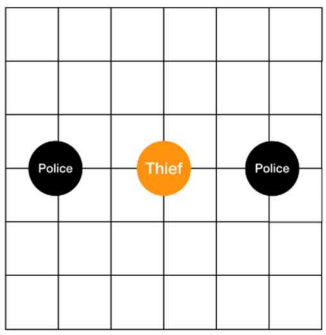
Problem F
Escaping
Time Limit: 1 second

We play a pursuit game where a thief is located at a specific grid point on an infinite grid plane. To catch this thief, N police officers are also deployed at N different grid points other than the thief's location. One thief and N police officers take turns moving alternatingly. The thief moves first. However, at police officers turn, all N police officers move at the same time. Police officers and thief can move to a point among the four adjacent (neighboring) grid points at their turn.

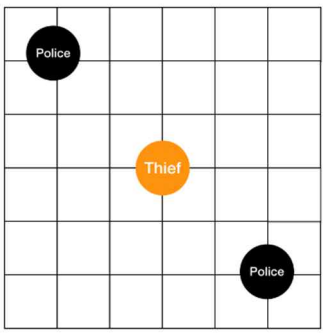
Each police officer can either stay or move, but the thief must move to one of the neighboring points at its turn. If a police officer and the thief meet at the same grid point, the thief is caught by the police officer and the game is over. In this game, we assume that all N police officers and the thief do their best. So the thief tries to escape eternally and the police officers also do their best to catch the thief as quickly as possible.

If the thief can continue moving without being caught by any police officer no matter what strategy the police officers use, then the initial game configuration is called an "escapable" case for the thief. You have to determine whether or not the given initial configuration is an "escapable" case for a thief.

For example, if the thief keeps moving in the vertical direction for the initial configuration shown in the left figure below, the police officers cannot catch the thief, so the thief can escape forever. However, for the initial configuration the right figure below, no matter how the thief moves in any direction, the thief is eventually caught by the police officers.



Escapable case



Non-escapable case

Input
Your program is to read from standard input. The input starts with a line containing one integer N, the number of police officers ($1 \leq N \leq 500{,}000$). Each of the following N lines contains two integers, x_i and y_i, where (x_i, y_i) is the initial position (grid coordinate) of each police officer. The next line contains two integers, s_x and s_y, where (s_x, s_y) is the initial position (grid coordinate) of the thief. All the values x_i, y_i, s_x, and s_y are between -10^9 and 10^9 inclusively. Note that all the thief and N police officers are located at different grid positions in the infinite grid plane.

Output

Your program is to write to standard output. Print exactly one line. The line should contain `YES` if the thief can escape forever for the initial configuration, `NO` if the initial configuration is not escapable.

The following shows sample input and output for two test cases.

Sample Input 1	**Output for the Sample Input 1**
2 1 3 5 3 3 3	YES

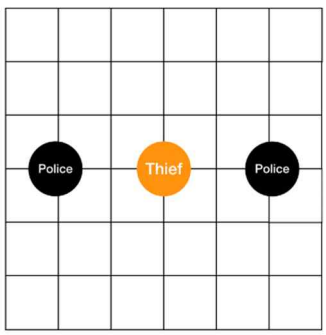
Sample Input 2	**Output for the Sample Input 2**
2 1 5 5 1 3 3	NO

Problem F
Escaping
제한 시간: 1 초

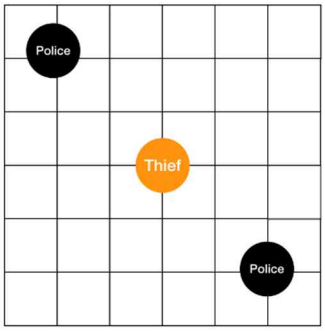
무한 정수 좌표 평면 위 한 격자 점에 있는 도둑 잡기 게임을 해 보자. 이 도둑을 잡기 위해 N명의 경찰이 도둑이 없는 N개의 다른 격자 점에 배치되어 있다. 도둑 한 명과 N명 경찰들은 서로 한번씩 번갈아 가면서 움직인다. 도둑이 먼저 시작한다. 경찰 차례일 때는 N 명의 경찰 모두가 동시에 움직인다. 경찰과 도둑은 상하좌우 인접한 네 개의 격자 점 중 하나로 이동할 수 있다. 단, 경찰은 그대로 머물러 있을 수 있지만 도둑은 자신의 차례가 오면 반드시 이웃 격자 점으로 이동해야 한다. 이때 경찰과 도둑이 같은 격자 점에서 만나면 도둑은 잡힌 것이 되고 게임은 끝이 난다. 이 게임에서 경찰과 도둑은 최선을 다한다. 즉, 도둑은 최대한 잡히지 않으려는 전략을, 경찰은 최대한 빨리 잡으려는 전략을 쓴다.

입력으로 주어진 초기 위치에서 경찰이 어떤 전략을 사용하더라도 도둑이 경찰에 의해 잡히지 않고 영원히 도망 다닐 수 있다면, 그 초기 조건은 도둑이 "**탈출 가능한**" 조건이라고 부른다. 여러분은 초기 조건, 즉 한 명의 도둑과 N 명의 경찰의 처음 위치를 보고 도둑이 탈출 가능한지 판단해야 한다.

예를 들어, 다음 아래 왼쪽 그림과 같은 초기 조건에서 도둑이 북쪽 또는 남쪽 방향으로만 움직인다면 경찰은 도둑을 영원히 잡을 수 없다. 그러나 아래 오른쪽 그림과 같은 조건이라면 도둑이 어떻게 도망을 가더라도 최선을 다하는 경찰에 의해서 결국은 잡히게 된다.



탈출 가능



탈출 불가능

Input

첫 번째 줄에는 경찰의 수 N이 주어진다. 단, $1 \leq N \leq 500{,}000$이다. 그 다음 N 개의 줄에는 각 경찰의 초기 위치의 좌표 (x_i, y_i)가 공백을 사이에 두고 주어진다. 다음 줄에는 도둑의 초기 위치의 좌표 (s_x, s_y)가 공백을 사이에 두고 주어진다. 도둑과 경찰의 좌표는 모두 다른 격자 점으로 주어지며, 정수 x_i, y_i, s_x, s_y의 범위는 $-10^9 \leq x_i,\ y_i,\ s_x,\ s_y \leq 10^9$이다.

Output

주어진 초기 조건에서 도둑이 무한히 도망갈 수 있다면, 즉, 초기 조건이 탈출 가능한 조건이라면, `YES`, 탈출 가능하지 않다면 `NO` 를 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	**Output for the Sample Input 1**
2 1 3 5 3 3 3	YES

Sample Input 2	**Output for the Sample Input 2**
2 1 5 5 1 3 3	NO

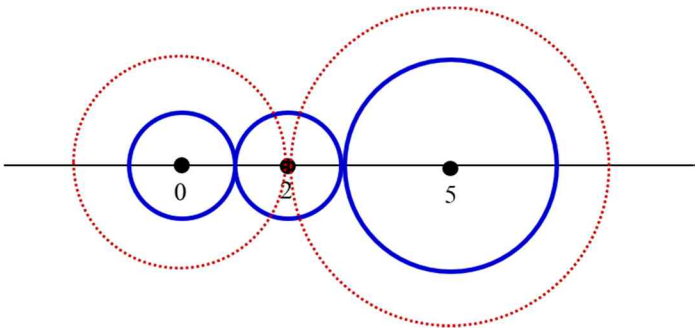
Problem G
Hotspots
Time Limit: 2.5 Seconds

A hotspot is a physical location where people can generally use Wi-Fi to access the Internet through a wireless local-area network (WLAN) with a router connected to an Internet service provider. Most people call these places "Wi-Fi hotspots". Public hotspots are typically created from wireless access points, shortly, APs. Specifically, a hotspot is a zone within distance r from where an AP is installed. That is, it is a circle with a radius r, centered at the location of AP.

In a city, there is a long straight road. The APs are already installed along the road. The city officials need to set the radii of hotspots. Then, for any two different APs, the hotspots created from them should not overlap, but at their boundaries, they can meet. As a special case, if the radius of hotspot is zero and another hotspot contains it inside, then the two hotspots overlap and it should not happen. But even if the radius of hotspot is zero, the hotspot can touch a boundary of another hotspot.

The city officials are trying to set the radii of hotspots such that their coverage areas are as large as possible. Thus, they shall maximize the sum of areas of hotspots, simply, the sum of squares of radii of hotspots. To achieve the goal, some radii of hotspots may be set to zero.

The road is considered as a line on the plane, and the locations of APs installed on the road are points on the line. Given the n points on the line, write a program to determine the radii of hotspots, maximizing the sum of squares of radii of hotspots, where the hotspots should not overlap.



For example, there are three APs located at 0, 2, and 5, respectively, in the above figure. As a candidate, the blue and red hotspots are given. The radii of the blue hotspots are 1, 1, and 2, from left to right. Then the sum of squares of radii is 6. But for the red hotspots, their radii are 2, 0, and 3, from left to right. Thus, the sum of squares of radii is 13, which is the maximum.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($2 \leq n \leq 3{,}000$), where n is the number of APs. The second line contains n distinct integers, separated by a space, in increasing order to represent the locations of APs, where the integers are between 0 and 10^9.

Output

Your program is to write to standard output. Print exactly one line. The line should contain a nonnegative integer that is the maximum sum of squares of radii of hotspots.

The following shows sample input and output for three test cases.

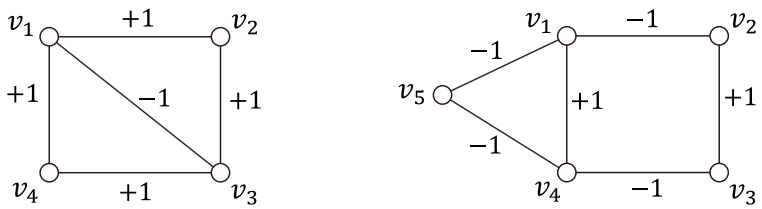
Sample Input 1	Output for the Sample Input 1
3 0 2 5	13

Sample Input 2	Output for the Sample Input 2
4 0 1 3 6	10

Sample Input 3	Output for the Sample Input 3
5 5 7 12 13 15	9

Problem H
Negative Cycle
Time Limit: 1 Second

Let G be a simple undirected graph having no self-loops or multiple edges. For two vertices x and y of G, a *path* from x to y in G is a sequence $\langle v_1, v_2, \ldots, v_l \rangle$ of distinct vertices of G such that $v_1 = x$, $v_l = y$, and v_i is adjacent to v_{i+1} for all $i \in \{1, \ldots, l-1\}$. If $l \geq 3$ and v_l is adjacent to v_1, the path is called a *cycle*. In a graph in which a weight of $+1$ or -1 is assigned to each edge, the weight product of the edges in a cycle will be $+1$ or -1. If the weight product is -1, the cycle is called a *negative cycle*; it is called a *positive cycle* otherwise. Refer to the graphs shown below for illustrative examples. The left graph contains negative cycles $\langle v_1, v_2, v_3 \rangle$ and $\langle v_1, v_3, v_4 \rangle$ and also positive cycle $\langle v_1, v_2, v_3, v_4 \rangle$, whereas the right graph contains no negative cycle, but positive cycles $\langle v_1, v_2, v_3, v_4 \rangle$, $\langle v_1, v_4, v_5 \rangle$, and $\langle v_1, v_2, v_3, v_4, v_5 \rangle$.



Write a computer program that determines the existence of a negative cycle in the graph given as an input and report an arbitrary negative cycle, if any.

Input
Your program is to read from standard input. The first line contains two positive integers n and m, respectively indicating the numbers of vertices and edges of a simple undirected graph, in which we assume $n \leq 20{,}000$ and $m \leq 200{,}000$. The vertices are indexed from 1 to n. In the following m lines, each line contains three integers x, y, and $w \in \{+1, -1\}$, which represent an edge (x, y) of weight w. The integers given in a single line are always separated by a single space.

Output
Your program is to write to standard output. The first line must contain an integer indicating whether the given graph has a negative cycle. If yes, the integer must be `1`; otherwise `-1`. When and only when the first line is `1`, it must be followed by the description of an arbitrary negative cycle of the input graph. A cycle is described by a single line containing an integer l, representing its length, followed by l lines containing, one by one, the vertices encountered when we traverse the cycle starting from an arbitrary vertex. Note that the vertices of the cycle must be distinct.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
4 5 1 2 +1 2 3 +1 3 4 +1 4 1 +1 3 1 -1	1 3 1 2 3

Sample Input 2	Output for the Sample Input 2
5 6 1 4 +1 4 5 -1 5 1 -1 1 2 -1 2 3 +1 3 4 -1	-1

Sample Input 3	Output for the Sample Input 3
6 4 1 2 -1 2 3 +1 4 5 +1 5 6 -1	-1

Problem I
Project Teams
Time Limit: 0.5 Seconds

Suchan is teaching a coding project class and wants to make project teams as fair as possible. A project team consists of two students, but the coding abilities of students are all different. Every student should be a member of a single team. To promote fairness, he wants to make project teams where the sum of the coding ability of team members is as even as possible. Write a program to help Suchan to make project teams given the coding abilities of students.

To make the problem simple, the number of students is even, say $2n$, where n is a positive integer. The coding ability of a student s_i is represented by a positive integer $w(s_i)$. The goal of your program is to make n teams, $G_1, G_2, \ldots,$ and G_n, to maximize $S_m = \min\{\sum_{s \in G_i} w(s) \mid 1 \le i \le n\}$ and print S_m.

For example, if the coding abilities of the students are given by $\{1, 7, 5, 8\}$, you can make two teams, each with $(8, 1)$ and $(7, 5)$, and your program should print 9.

Input
Your program is to read from standard input. The input starts with a line containing a positive integer n ($1 \le n \le 5{,}000$), where n is the number of teams to make. The next line contains $2n$ positive integers separated by a space, each of which represents the coding ability $w(s_i)$ of a student s_i ($1 \le w(s_i) \le 100{,}000$). Note that the coding abilities of students are all different, that is, $w(s_i) \ne w(s_j)$ if $i \ne j$.

Output
Your program is to write to standard output. Print exactly one line containing S_m.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
2 1 7 5 8	9

Sample Input 2	Output for the Sample Input 2
3 1 7 3 5 9 2	8

Problem I
Project Teams
제한 시간: 0.5 초

코딩 프로젝트 수업을 가르치는 수찬이는 프로젝트 팀을 가능하면 공정하게 구성하려고 한다. 프로젝트 팀 하나는 두 명의 학생으로 구성되는데, 각 학생들의 코딩 역량은 모두 다르다. 각 학생은 한 팀의 팀원이어야 한다. 공정성을 높이기 위해 수찬이는 팀원 코딩 역량의 합을 최대한 일정하게 유지하려고 한다. 학생들이 코딩 역량이 주어졌을 때 수찬이가 팀을 구성하는데 도움이 되는 프로그램을 작성하라.

문제를 간단하게 하기 위해, 학생 수는 $2n$이라 가정하자 (n은 양의 정수). 각 학생 s_i의 코딩 역량은 양의 정수 $w(s_i)$로 나타내면 한 i번째 팀 G_i의 코딩 역량은 $w(G_i) = \sum_{s \in G_i} w(s)$로 나타낼 수 있다. 작성할 프로그램 목적은 $S_m = \min\{w(G_i) \mid 1 \le i \le n\}$이 최대화되도록 n개의 팀 $G_1, G_2, \ldots, G_n$을 구성하고 이때 S_m을 출력하는 것이다.

예를 들어, 학생들의 코딩 역량이 $\{1, 7, 5, 8\}$로 주어졌다면 $(8, 1)$, $(7, 5)$로 2개의 조를 짤 수 있으며 프로그램은 9를 출력해야 한다.

Input

입력은 표준입력을 사용한다. 입력의 첫 번째 행에는 팀 수를 나타내는 양의 정수 $n (1 \le n \le 5{,}000)$이 주어진다. 그 다음 행에 학생 s_i의 코딩 역량 $w(s_i)$를 나타내는 $2n$개의 양의 정수가 공백으로 분리되어 주어진다 $(1 \le w(s_i) \le 100{,}000)$. 학생들의 코딩 역량은 모두 다르다. 즉, $i \neq j$이면 $w(s_i) \neq w(s_j)$이다.

Output

출력은 표준출력을 사용한다. 표준출력 한 행에 S_m을 출력한다.

다음은 두 테스트 경우에 대한 입출력 예이다.

Sample Input 1	**Output for the Sample Input 1**
2 1 7 5 8	9

Sample Input 2	**Output for the Sample Input 2**
3 1 7 3 5 9 2	8

Problem J
Trading System
Time Limit: 1 Second

SY Company wants to improve its stock trading system. For this, the company decides to utilize the information on the fluctuation of the stock prices. The fluctuation value is the difference in stock prices for two consecutive days. The company collects n recent fluctuation values for some stock. It turns out that the stock volatility is greatly affected by the largest sum of the contiguous fluctuation values. Finding such contiguous fluctuation values whose sum is the maximum is known as the *largest sum contiguous subarray problem* in computer science, where input values are stored in an array. It is natural that utilizing the $k(\geq 1)$ largest contiguous sums rather than the largest one will help improve the trading system.

Write a program to find the k largest sums of contiguous fluctuation values for the given n fluctuation values and a positive integer k.

Input
Your program is to read from standard input. The input starts with a line containing two integers, n and k, where $1 \leq n \leq 250{,}000$ and $1 \leq k \leq min(10{,}000, n(n+1)/2)$. The next line contains n integers representing n fluctuation values. All fluctuation values are between -10^9 and 10^9 inclusively.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the k largest sums of contiguous fluctuation values in non-increasing order. Note that any contiguous sum is the sum of one or more consecutive fluctuation values.

The following shows sample input and output for two test cases.

Sample Input 1

```
5 3
1 -2 -3 5 4
```

Output for the Sample Input 1

```
9 6 5
```

Sample Input 2

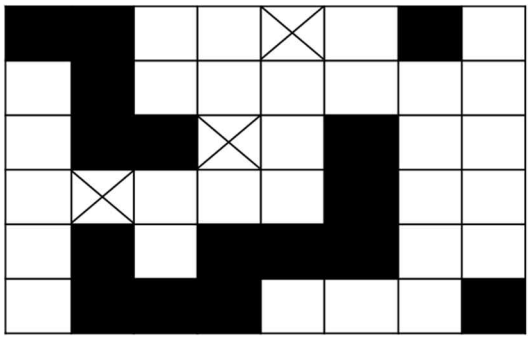
```
6 10
3 8 -3 2 5 2
```

Output for the Sample Input 2

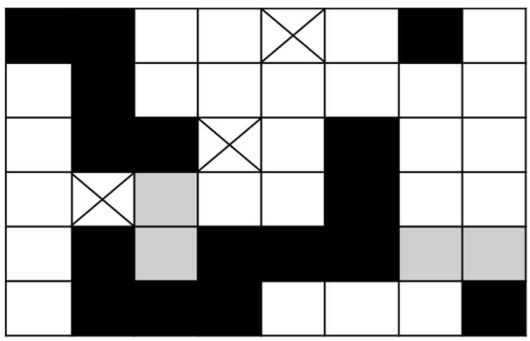
```
17 15 14 12 11 10 9 8 8 7
```

Problem K
Road Reconstruction
Time Limit: 1 Second

ICP(International Computational Plan) City is faced with natural disaster, flooding. A lot of its transportation roads are destructed because of the flooding. The city is planning to reconstruct its roads. To simplify the problem, the city is represented by a grid structure like the following figure. We call it as *city grid*. A unit grid cell with black color means a unit road. A unit grid cell with white color means an area where a unit road is destructed or where a unit road has not existed previously. A unit grid cell with X denotes an area where a road cannot be constructed.



We assume that cars in the city can go horizontally or vertically from a black colored cell to an adjacent black colored cell. For two grid cells u and v in the city grid, if a car can go from u to v and the both cells are black colored, then u and v are called *connected*. ICP city is planning to reconstruct its transportation roads by connecting grid cells from the upper and leftmost grid to the lower and rightmost one with minimum construction cost. We assume that it costs 1 or 2 for constructing a unit road. For the example above, the gray cells in the following figure show how to construct roads that connect the upper and leftmost grid cell with the lower and rightmost one by the minimum construction cost 4. We assume that it costs 1 for constructing a unit road on the white cells in the figure. Note that it costs 1 or 2 for constructing a unit road in general case.



Given a city grid, write a program to compute the minimum construction cost that connects roads from the upper and leftmost grid cell to the lower and rightmost one.

Input

Your program is to read from standard input. The input starts with a line containing two integers, m and n ($1 \le m \le 1{,}000$, $1 \le n \le 1{,}000$), where m and n denote the numbers of rows and columns of the grid that represent the city, respectively. In the following m lines, the i-th line contains n integers that represent states of unit grid cells in the i-th row of the city grid. Each state is represented by using four numbers `0`, `1`, `2`, and `-1`. The number `0` indicates that the current unit grid cell is a unit road, that is, an undamaged road. The numbers `1` and `2` indicate that the current unit grid cell is empty and a unit road is constructible with the costs `1` and `2`, respectively. The number `-1` indicates that the current unit grid cell cannot be used for constructing a road.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum construction cost that connects roads from the upper and leftmost unit grid cell to the lower and rightmost unit one. When it is impossible to connect roads from the upper and leftmost unit grid cell to the lower and rightmost one, the program should print out `-1` on the line.

The following shows sample input and output for four test cases. The third case corresponds to the example above.

Sample Input 1

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

Output for the Sample Input 1

```
-1
```

Sample Input 2

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

Output for the Sample Input 2

```
-1
```

Sample Input 3

```
6 8
0 0 1 1 -1 1 0 1
1 0 1 1 1 1 1 1
1 0 0 -1 1 0 1 1
1 -1 1 1 1 0 1 1
1 0 1 0 0 0 1 1
1 0 0 0 1 1 1 0
```

Output for the Sample Input 3

```
4
```

Sample Input 4

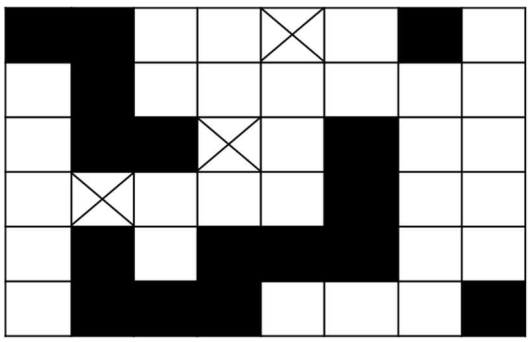
```
6 8
2 0 1 1 -1 1 0 1
1 0 1 1 1 1 1 1
1 0 0 -1 1 0 1 1
1 -1 2 1 1 0 1 1
1 0 2 0 0 0 -1 1
1 0 0 0 1 2 2 0
```

Output for the Sample Input 4

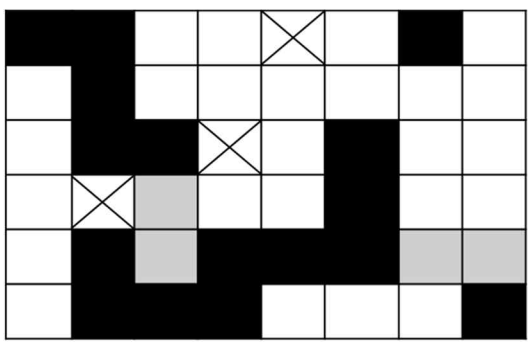
```
8
```

Problem K
Road Reconstruction
제한 시간: 1 초

홍수의 발생으로 인해 도시의 도로들이 유실되어 많은 ICP(International Computational Plan) 시민들이 불편을 겪고 있다. 도시는 아래와 같은 격자 형태로 표현이 된다고 가정하자. 검정색 단위 격자가 '단위도로'를 의미하며 흰색 단위 격자는 도로가 없었거나 유실된 상태를 의미한다. X로 표시된 단위 격자는 도로를 건설할 수 없는 곳을 의미한다.



도시의 차들은 단위 도로 상에서 가로나 세로 방향으로만 움직인다고 가정했을 때 도시의 기능을 회복시키기 위해 맨 왼쪽 위 단위 격자에서 맨 오른쪽 아래 단위 격자로 도시의 차들이 가는 경로를 만들기 위해 필요한 최소한의 도로 건설 비용을 계산하고자 한다. 단위 도로 하나를 건설하기 위해 1 또는 2 의 비용이 소요된다고 가정하자. 위 그림에서는 흰색 단위 격자에 단위 도로 하나를 건설하기 위해서는 1 의 비용이 든다고 가정한다.



위와 같이 회색으로 표시된 단위 도로들을 4 의 비용으로 건설하면 목적을 달성할 수 있다.

도시가 위와 같이 격자 형태로 주어졌을 때 맨 왼쪽 위 단위 격자에서 맨 오른쪽 아래 단위 격자로 가는 경로를 만들기 위해 필요한 최소 도로 건설 비용을 구하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 도시를 표현하는 격자의 행과 열의 크기를 각각 나타내는 두 개의 양의 정수 m, n $(1 \leq m, n \leq 1{,}000)$이 주어진다. 다음 m개의 각 줄에 격자의 각 열의 정보를 나타내는 n개의 숫자가 주어진다. 각 열의 정보는 정수 `0`,`1`,`2`,`-1` 로 나타내며 `0` 은 단위도로가 이미 존재하는 것을, 즉, 도로에 유실이 없는 상태, `1` 은 단위 도로가 없고 `1` 의 비용으로 도로를 건설할 수 있는 단위 격자, `2` 는 단위 도로가 없고 `2` 의 비용으로 도로를 건설할 수 있는 단위 격자를 의미한다. `-1` 은 X로 표시된 단위 격자로 그 위치에 단위 도로를 건설할 수 없는 상태를 의미한다.

Output

출력은 표준출력을 사용한다. 도시의 맨 왼쪽 위 단위 격자에서 맨 오른쪽 아래 단위 격자로 가는 경로를 만들기 위해 필요한 최소한의 도로 건설 비용을 한 줄에 출력한다. 해당 경로를 건설할 수 없을 때는 `-1` 을 출력한다.

다음은 네 개의 테스트 경우에 대한 입출력 예이다. 세 번째 테스트경우는 위 예제에 대한 것이다.

Sample Input 1

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

Output for the Sample Input 1

```
-1
```

Sample Input 2

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

Output for the Sample Input 2

```
-1
```

Sample Input 3

```
6 8
0 0 1 1 -1 1 0 1
1 0 1 1 1 1 1 1
1 0 0 -1 1 0 1 1
1 -1 1 1 1 0 1 1
1 0 1 0 0 0 1 1
1 0 0 0 1 1 1 0
```

Output for the Sample Input 3

```
4
```

Sample Input 4

```
6 8
2 0 1 1 -1 1 0 1
1 0 1 1 1 1 1 1
1 0 0 -1 1 0 1 1
1 -1 2 1 1 0 1 1
1 0 2 0 0 0 -1 1
1 0 0 0 1 2 2 0
```

Output for the Sample Input 4

```
8
```

Problem L
Sliding Coins
Time Limit: 0.5 Seconds

Suppose pennies and nickels are arranged on the row. Now we define the 'two-finger sliding' on these coins as follows:

Step 1. Choose any two arbitrary coins on the row.
Step 2. Move them to any position on the row, while keeping their relative order. Note that 'any position' contains their original positions (that means one can move only one, or none of coins for the two-finger sliding if their relative order is preserved).

The relative order of unchosen coins is also preserved after the two-finger sliding. For example, suppose the initial arrangement of coins is `oxox**ox**xxoo`, where `o` and `x` represent pennies and nickels respectively. Then the following example shows four possible arrangements after one chooses two coins marked in bold and performs exactly one two-finger sliding (note that there are other possible arrangements).

oxoxoxxxoo
ox**o**oxxx**x**oo
oxox**o**xxo**x**o
oxox**ox**xxoo

Given two arrangements S and T of n coins and the positions of two chosen coins for the two-finger sliding, write a program to decide whether one can change the arrangement from S to T by exactly one two-finger sliding for the specified two coins.

Input
Your program is to read from standard input. The input starts with a line containing an integer n, ($3 \leq n \leq 10{,}000$), where n is the number of coins on the row. In the following two lines, two arrangements S and T of n coins are given where each arrangement is a string of size n from the lowercase alphabet `o` and `x`. The next line contains two nonnegative integers i and j which represent the positions of chosen coins for the two-finger sliding. The coin positions are indexed from 0 to $n-1$ from left to right, and i is smaller than j ($0 \leq i < j \leq n-1$).

Output
Your program is to write to standard output. Print `YES` if one can change the arrangement from S to T by exactly one two-finger sliding for the specified two coins, and print `NO` otherwise.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
9 oxoxoxxxo oxoxoxxox 4 5	YES

Sample Input 2	Output for the Sample Input 2
10 oxoxxoxxxo ooxoxxxxox 3 4	NO

Problem L
동전 옮기기
제한 시간: 0.5 초

100 원짜리 동전과 10 원짜리 동전이 임의의 순서로 한 선 위에 나열되어 있다고 하자. 이제 여기서 '**두 손가락 이동**' 을 아래와 같이 정의하자.

단계 1: 임의의 두 동전을 선택한다.
단계 2: 단계 1 에서 선택한 두 동전을 둘의 순서를 유지한 채 임의의 위치로 이동한다. (두 동전 모두 제자리에 있거나 두 동전의 순서를 유지한다면 하나만 이동해도 된다.)

'두 손가락 이동' 후에도 다른 동전들 간의 순서는 그대로 유지된다. 예를 들어 100 원을 ○, 10 원을 x 라 했을 때, 초기에 동전이 ○x○x**ox**xx○○ 와 같이 나열되어 있다 하자. 이제 이들 중 굵게 표시된 두 동전을 선택하여 두 손가락 이동을 한번 한 경우, 나올 수 있는 여러 결과들 중에서 네 가지 결과만 아래에 표시했다 (아래 예시에 없는 다른 결과들 또한 나올 수 있음에 유의하자).

ox○x○x**x**x○○
○x**o**○x**x**x**x**○○
○x○x**o**x**x**○**x**○
○x○x**ox**x**x**○○

n 개의 동전이 나열되어 있는 두 상태 S, T와 함께 두 손가락 이동을 위해 선택할 두 동전의 위치가 주어졌을 때, 한번의 두 손가락 이동을 통해 S에서 T로의 변환이 가능한지 결정하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 나열된 동전 개수를 나타낸 양의 정수 n ($3 \le n \le 10{,}000$)이 주어진다. 다음 두 줄에 n 개의 동전이 나열된 상태인 S 와 T 가 각각 주어지며, 이 때 S와 T 는 ○ 와 x 로 이루어진 길이가 n 인 문자열로 표현된다 (○ 와 x 는 모두 소문자이다). 동전의 위치는 왼쪽에서 오른쪽으로 0부터 $n-1$까지 차례대로 번호를 매겨 구분한다. 마지막 줄에는 두 손가락 이동을 위해 선택할 두 동전의 위치 i와 j가 주어지며, i는 j 보다 작다 ($0 \le i < j \le n-1$).

Output

출력은 표준출력을 사용한다. 한번의 두 손가락 이동을 통해 S 에서 T로의 변환이 가능한 경우 `YES` 를, 그렇지 않은 경우 `NO` 을 출력한다.

다음은 두 테스트 경우에 대한 입출력 예이다.

Sample Input 1	**Output for the Sample Input 1**
9 oxoxoxxxo oxoxoxxox 4 5	YES

Sample Input 2	**Output for the Sample Input 2**
10 oxoxxoxxxo ooxoxxxxox 3 4	NO