The 43rd Annual ACM

International Collegiate Programming Contest

Asia Regional – Seoul

Nationwide Internet Competition

Problem Set

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

The memory limits for the twelve problems are all the same, 512MB.

Problem A
Black Chain
Time Limit: 0.1 Second

There is a linear chain of n black rings. The weight of every black ring is exactly 1g. We want to generate all possible weights from 1g to ng using this black chain. To do this, we need to remove some single rings from the chain. Since it is very difficult work to remove a single ring from the chain, we want to remove the minimum number of rings as possible. For example, consider the black chain with 7 rings as shown in Figure A.1. If ring 3 is removed, the chain would be separated into a single ring, a chain with rings from 1 to 2, and a chain with rings from 4 to 7 as shown in Figure A.2. Using them we can generate all weights from 1g to 7g as shown in Figure A.3.

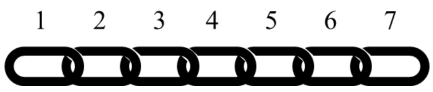


Figure A.1: A black chain with a length of 7.

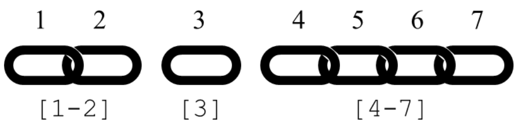


Figure A.2: A black chain separated into 3 pieces.

Weight	1g	2g	3g	4g	5g	6g	7g
Rings Configuration	[3]	[1-2]	[3] [1-2]	[4-7]	[3] [4-7]	[1-2] [4-7]	[3] [1-2] [4-7]

Figure A.3: Rings configurations for generating all weights from 1g to 7g.

Given a chain with n black rings, write a program to compute the minimum number of rings which should be removed for generating all weights from 1g to ng.

Input
Your program is to read from standard input. The input starts with a line containing an integer, n ($3 \leq n \leq 10^{18}$), where n is the number of rings in the black chain.

Output
Your program is to write to standard output. Print exactly one line which contains the minimum number of rings which should be removed for generating all weights from 1g to ng.

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

Sample Input 1	Output for the Sample Input 1
`7`	`1`

Sample Input 2	Output for the Sample Input 2
`20`	`2`

Problem A
Black Chain

제한 시간: 0.1 초

n개의 블랙 고리가 일렬로 연결된 체인이 있다. 블랙 고리 하나는 무게가 정확히 1g 이다. 이 고리들을 이용하여 1g 부터 ng 까지 가능한 모든 무게를 생성하려고 한다. 이를 위해 고리를 일부 풀어야 하는데, 고리를 푸는데 힘이 들어 최소 개의 고리만 풀기를 원한다. 예를 들어 아래의 그림 A.1 처럼 7 개의 고리로 구성된 블랙 체인이 있다고 하자. 이 체인에서 3 번 고리 하나를 풀어 내면 그림 A.2 처럼 3 번 고리 1 개와 두 개의 체인(1~2 번 고리가 연결된 체인과 4~7 번 고리가 연결된 체인)으로 분리된다. 이들을 이용하면 그림 A.3 처럼 1g 부터 7g 까지의 모든 무게를 생성할 수 있다.

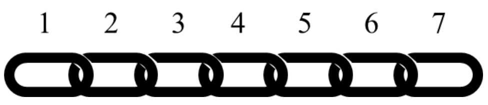


그림 A.1: 길이가 7 인 블랙 체인

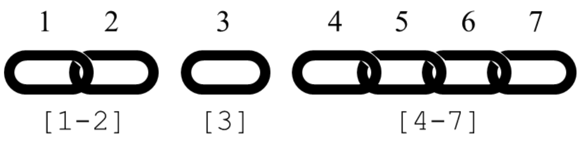


그림 A.2: 3 개로 분리된 블랙 체인

무게	1g	2g	3g	4g	5g	6g	7g
고리 구성	[3]	[1-2]	[3] [1-2]	[4-7]	[3] [4-7]	[1-2] [4-7]	[3] [1-2] [4-7]

그림 A.3: 1g 부터 7g 까지 가능한 모든 무게를 생성하는 고리 구성

n개의 고리가 연결된 체인이 주어져 있을 때, 1g 부터 ng 까지 가능한 모든 무게를 생성하기 위해 풀어야 할 고리의 최소 개수를 구하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 블랙 고리의 개수를 나타내는 양의 정수 n ($3 \leq n \leq 10^{18}$)이 주어진다.

Output

출력은 표준출력을 사용한다. 1g 부터 ng 까지 가능한 모든 무게를 생성하기 위해 풀어야 할 고리의 최소 개수를 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
7	1

Sample Input 2	Output for the Sample Input 2
20	2

The 43rd Annual ACM
International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

Problem B
Farm
Time Limit: 0.1 Second

Sangbae raises sheep and goats on his farm where each of the number of sheep and the number of goats is at least one. Sheep and goats eat same food, and every day each sheep eats exactly a grams of food and each goat eats exactly b grams of food.

Sangbae checks every day how many sheep and goats are in the farm. It was not easy to count them separately because they were moving around. He counted the total number of sheep and goats instead. Also he checked total amount of food that sheep and goats have eaten yesterday. From these values, he wants to find the number of sheep and the number of goats.

Given a, b, n (total number of sheep and goats), w grams of food that both sheep and goats have eaten yesterday, write a program to find the number of sheep and the number of goats.

Input
Your program is to read from standard input. The input consists of single line containing four integers, a, b, n, and w where $1 \leq a \leq 1{,}000$, $1 \leq b \leq 1{,}000$, $2 \leq n \leq 1{,}000$, $2 \leq w \leq 1{,}000{,}000$.

Output
Your program is to write to standard output. Print exactly one line which contains both the number of sheep and the number of goats. If there are two or more feasible solutions or there is no feasible solution, print -1.

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

Sample Input 1	Output for the Sample Input 1
`3 4 9 32`	`4 5`

Sample Input 2	Output for the Sample Input 2
`3 4 8 32`	`-1`

Sample Input 3	Output for the Sample Input 3
`100 100 2 200`	`1 1`

Problem B
Farm

제한 시간: 0.1 초

목장 주인인 상배는 양과 염소들을 같이 기르고 있다. 기르는 양과 염소는 각각 한 마리 이상이다. 양과 염소는 같은 사료를 먹고, 양 한 마리는 하루에 사료를 정확히 a 그램 먹고, 염소 한 마리는 하루에 정확히 b 그램을 먹는다고 한다.

상배는 매일 아침 양과 염소가 각각 몇 마리인지를 확인하는 작업을 한다. 양과 염소가 각각 몇 마리인지 확인할 때, 양과 염소들이 돌아 다녀서 정확하게 그 수를 구하는 것이 쉽지 않았다. 대신에 양과 염소가 전체 몇 마리인지를 확인하고, 또 양과 염소가 어제 하루 동안 소비한 전체 사료의 양만 확인해서 양과 염소가 각각 몇 마리 인지를 알려고 한다.

상배가 확인한 양과 염소 전체가 n마리이고, 어제 하룻동안 소비한 전체 사료의 양이 w그램일 때, 양과 염소가 각각 몇 마리인지를 구하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 네 정수 a, b, n, w가 한 줄에 주어진다. $1 \leq a \leq 1{,}000$, $1 \leq b \leq 1{,}000$, $2 \leq n \leq 1{,}000$, $2 \leq w \leq 1{,}000{,}000$이다.

Output

출력은 표준출력을 사용한다. 첫 번째 줄에 양의 수와 염소의 수를 각각 출력한다. 만약 가능한 해가 두 개 이상 있는 경우 혹은 가능한 해가 없을 경우, -1 을 출력한다.

다음은 세 테스트 경우에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
3 4 9 32	4 5

Sample Input 2	Output for the Sample Input 2
3 4 8 32	-1

Sample Input 2	Output for the Sample Input 2
100 100 2 200	1 1

Problem C
Lucid Strings
Time Limit: 0.5 Seconds

Consider a string on English alphabet of 26 lowercase letters. If the length of this string can be represented as the product of two positive integers, k (≥ 2) and c, the string can be decomposed into k substrings of the same length c. If the substrings are pairwise distinct, the string is called a "k-Lucid-String" (shortly "k-LS"). Here a substring is a contiguous sequence of characters within a string.

For example, for string `ababca` of length 6, there are three cases for k to consider: $k = 2, 3, 6$. For $k = 2$, the string is decomposed into two substrings, `aba` and `bca`, of length 3. Since the two substrings are pairwise distinct, the string is a 2-LS. For $k = 3$, the string is decomposed into three substrings of length 2. But `ab` appears as substring more than once, and thus the string is not a 3-LS. For $k = 6$, the string is decomposed into six substrings of length 1. But `a` and `b` appear as substring more than once, and thus the string is not a 6-LS.

Consider the problem of computing all substrings of a given input string which are k-LS's. For example, consider input string `ababca` for $k = 2$. Since each of substrings `ab`, `ba`, `ab`, `bc`, and `ca` of `ababca` can be decomposed into two pairwise distinct substrings of length 1, it is a 2-LS. Substrings `babc` and `abca` are 2-LS's because each of them can be decomposed into two pairwise distinct substrings of length 2. Since input string itself is a 2-LS, `ababca` has 8 substrings which are 2-LS's. Note that two substrings of input string are considered independently for k-LS candidates if they differ in position in input string.

Given a string S of length n and an integer k, write a program to compute the number of the substrings of S which are k-LS's.

Input
Your program is to read from standard input. The input starts with a line containing two integers, n $(3 \leq n \leq 40{,}000)$ and k $(2 \leq k \leq 40{,}000)$, where n is the length of input string and k is the number of substrings of the same length for each k-LS. You can assume that $k \leq n$. In the following line, a string of length n is given.

Output
Your program is to write to standard output. Print exactly one line which contains the number of the substrings of input string which are k-LS's.

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

Sample Input 1	Output for the Sample Input 1

`6 2`
`ababca`
	`8`

`

Sample Input 2	Output for the Sample Input 2

`6 3`
`ababca`
	`2`

The 43rd Annual ACM
International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

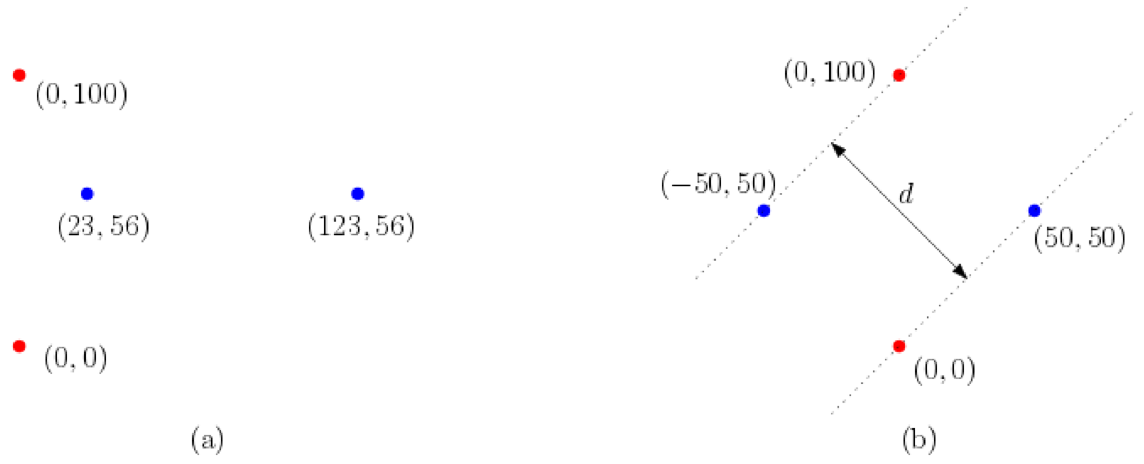
Problem D
Matching
Time Limit: 1 Second

In the geometric matching problem, two geometric objects A and B are given, and the goal is to find an optimal transformation for B such that the transformed copy of B is as close to A as possible. Usually, the distance between A and a transformed copy of B is measured by a prescribed distance function, and one wants to minimize it over all possible transformations.

Here, we consider a simple variant of the geometric matching problem. Specifically, we assume that two input objects A and B are finite sets of points in the plane and allowed transformations for B are only *translations* in the plane. A translated copy of B by a two-dimensional vector $v = (v_x, v_y)$ is defined to be

$$B + v = \{(x + v_x, y + v_y) \mid (x, y) \in B\}.$$

For any two-dimensional vector v, our *distance function* $f(v)$ measures the smallest possible perpendicular distance between two parallel lines that contain all points of A and $B + v$ in between. That is, we want to find an optimal two-dimensional vector v such that $f(v)$ is minimized.



Consider an example of $A = \{(0,0), (0,100)\}$ and $B = \{(23,56), (123,56)\}$, depicted in the above figure (a) in which the points in A are colored red while those in B are blue. Then, consider a specific vector $v = (-73, -6)$. The above figure (b) shows A and $B + v$, and two parallel lines whose perpendicular distance is d, which is exactly $d = 50\sqrt{2}$. One can verify that these two parallel lines contain all points of A and $B + v$ in between and have the smallest possible perpendicular distance. Hence, we have $f(v) = d$. Further, this is the minimum possible value of $f(v)$ over all two-dimensional vectors. Therefore, $v = (-73, -6)$ is an optimal translation vector such that $f(v)$ is minimized.

Given two sets of points in the plane, A and B, write a program that finds an optimal translation vector v for B such that $f(v)$ is minimized and outputs the value of $f(v)$.

Input

Your program is to read from standard input. The input starts with a line containing two integers, n ($1 \leq n \leq 200{,}000$) and m ($1 \leq m \leq 200{,}000$), where n is the number of points in the set A and m is the number of points in the set B. In each of the following n lines, the coordinates of each point in A are given by two integers separated by a space. Again, in each of the following m lines, the coordinates of each point in B are given by two integers separated by a space. The coordinates of all points given in the input range from -10^6 to 10^6, inclusively. Note that multiple points with the same coordinates can be given in each of A and B.

Output

Your program is to write to standard output. Print exactly one line which contains a real number z that represents the value of $f(v)$ for an optimal two-dimensional vector v such that $f(v)$ is minimized. The output z should be in the format that consists of its integer part, a decimal point, and its fractional part, and should satisfy the condition that $f(v) - 10^{-6} < z < f(v) + 10^{-6}$.

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

Sample Input 1	Output for the Sample Input 1

2 2
0 0
0 100
23 56
123 56
	70.710678

Sample Input 2	Output for the Sample Input 2

7 5
0 0
99 0
47 31
36 4
10 2
71 5
45 17
98 97
89 96
101 96
132 113
122 110
	31.000000

Problem E
Panokseon
Time Limit: 1 Second

Admiral Yi Sun-sin ordered to build the special battleship, called Panokseon, to prepare the war in the Joseon Dynasty. To test its performance, he decided to have a battleship race with Joseon navy soldiers. Admiral Yi numbered n soldiers from 1 to n, lined up in that order. He assigned the soldiers to the ships such that soldiers assigned to a ship have consecutive numbers, and the sum of their weights must not exceed the weight limit W of the ship. He did not mind the number of ships used for this assignment since he built enough number of ships. The point he cared about is the fairness of the race. He thought the fairness would be guaranteed if the soldiers' weight sums are well balanced over the ships. In other words, soldiers should be evenly assigned to the ships for their weights.

More precisely, the emptiness of a ship that one or more soldiers are assigned to is defined as the square of the difference of W and the sum of weights of the soldiers of the ship. The unfairness of an assignment is the maximum emptiness of the ships in the assignment. Admiral Yi wanted to guarantee the high fairness by such an assignment for the race whose unfairness is minimum, i.e., the fairness is maximized.

For instance, we are given a weight sequence $(10, 20, 30)$ of 3 soldiers with $W = 50$. An assignment $\{[1], [2], [3]\}$ that each soldier is assigned to an individual ship has the unfairness value of $\max\{(50 - 10)^2, (50 - 20)^2, (50 - 30)^2\} = 1600$. We can also consider two other assignments using two ships, $\{[1, 2], [3]\}$ and $\{[1], [2, 3]\}$, whose unfairness values are $\max\{(50 - 30)^2, (50 - 30)^2\} = 400$ and $\max\{(50 - 10)^2, (50 - 50)^2\} = 1600$, respectively. All three soldiers cannot be assigned to one ship because their weight sum is over $W = 50$. Thus the minimum unfairness value is 400, and the fairest assignment is that the first two soldiers are assigned to the one ship and the third soldier to the other ship alone.

Given a weight limit W of the battleship and the weights of n soldiers numbered 1 to n, write a program to find a fairest assignment for the battleship race.

Input
Your program is to read from standard input. The input starts with a line containing two integers, W ($1 \leq W \leq 10^9$) and n ($1 \leq n \leq 500{,}000$), where W is the weight limit of the battleship and n is the number of Joseon navy soldiers. The second line contains a sequence of n weights of soldiers, ordered by the soldiers' numbers from 1 to n. The weights are integers between 1 and W, inclusively. You can assume that no single soldier weighs more than W.

Output
Your program is to write to standard output. Print exactly one line which contains the minimum unfairness for the input.

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

Sample Input 1	Output for the Sample Input 1
50 3 10 20 30	400

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

Problem E
Panokseon
제한 시간: 1 초

이순신 장군은 판옥선이라 불리는 특별한 전함을 전쟁에 대비해 만들었다. 판옥선의 성능을 평가하기 위해, 조선 수병들로 하여금 경주를 하도록 할 계획이다. n 명의 수군들을 1 번부터 n번까지 차례로 번호를 매긴 후, 각 전함에 연속된 번호의 수군들의 몸무게의 합이 전함의 중량 한계 W를 넘지 않도록 배정하려고 한다. 즉, 한 전함에 배정된 수군들은 연속된 번호를 가져야 하며, 그들의 몸무게 합이 W 이하여야 한다. 충분히 많은 전함을 준비했기 때문에, 한 명 이상의 수군이 배정된 전함의 수는 중요하지 않고, 경주의 공정성, 즉 배정의 공정성이 더 중요하다. 이순신 장군은 전함 별 수군들의 몸무게 합의 차이가 작을수록 더 공정한 배정이라고 판단하였다.

보다 명확히 설명하면 다음과 같다. 한 명 이상의 수군이 배정된 전함의 여유중량 (emptiness) 값은 W와 그 전함에 배정된 수군들의 몸무게 합의 차이를 제곱한 값으로 정의한다. 배정의 불공정성 (unfairness) 값은 배정에 사용된 군함의 여유중량 값들 중에서 최대값으로 정의한다. 이순신 장군은 불공정성이 가장 작은 배정이 가장 공정한 배정이라고 생각했다.

예를 들어, 3 명의 수군의 몸무게가 차례대로 $10, 20, 30$이고, $W = 50$ 인 경우를 고려해보자. 가능한 배정 중 하나인 $\{[1], [2], [3]\}$은 하나의 전함에 수군을 한 명씩 배정하는 것이다. 이 배정의 불공정성은 $\max\{(50-10)^2, (50-20)^2, (50-30)^2\} = 1600$이다. 다른 두 가지 배정 $\{[1,2],[3]\}$과 $\{[1],[2,3]\}$ 도 가능한데, 두 배정의 불공정성은 각각 $\max\{(50-30)^2, (50-30)^2\} = 400$ 과 $\max\{(50-10)^2, (50-50)^2\} = 1600$이 된다. 그러나 3 명의 수군들의 몸무게 합이 $W = 50$보다 크기 때문에 모두 한 전함에 배정할 수는 없다. 따라서, 불공정성의 최소 값은 400이 되어 $\{[1,2],[3]\}$이 가장 공정한 배정이 된다. 이 배정은 1번, 2번 수군을 같은 전함에, 3번 수군은 다른 전함에 배정하는 것이다.

여러분은 전함의 한계 중량 W와 1번부터 n번 수군까지 몸무게가 차례로 주어질 때, 가장 공정한 배정을 찾는 프로그램을 작성해야 한다.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에는 전함의 한계 중량 W ($1 \le W \le 10^9$)와 수군의 수 n ($1 \le n \le 500{,}000$)이 공백을 사이에 두고 차례로 주어진다. 두 번째 줄에는 수군의 몸무게가 1번 수군부터 n번 수군의 순서로 차례로 주어진다. 수군 몸무게는 1 이상 W 이하의 정수이다. 수군 한 명의 몸무게는 W보다 크지 않다.

Output

출력은 표준출력을 사용한다. 주어진 입력에 대한 배정의 불공정성의 최소 값을 한 줄에 출력한다.

다음은 두 테스트 경우에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
50 3 10 20 30	400

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

Problem F
Parcel
Time Limit: 4 Seconds

ICPC (International Collegiate Parcel Center) is leading the event for free international delivery of parcels among world-wide collegiate students. The requirements for free delivery are that the parcel should consist of four items and the sum of weights of them should be equal to the certain integral weight w gram.

Chansoo in Pusan National University has a lot of items to send to his friend Soowhan in Imperial College in London, and the weights of the items are all different. Since this is a limited-time event, Chansoo wants to check as fast as possible whether he has four items satisfying to the condition or not. Assume that the weights of the items are exact positive integer grams, and remember they are all distinct. In other words, Chansoo wants to select a subset B with four elements ($|B| = 4$) from a set A of n ($n \geq 4$) distinct positive integers, and wants to check whether $\sum_{b \in B} b = w$ or not.

Given a target weight w and a set A of n distinct positive integers, write a program to print `YES` if such subset B exists, and `NO`, otherwise.

Input
Your program is to read from standard input. The input starts with a line containing two positive integers w and n separated by a space, where w ($10 \leq w \leq 799{,}994$) is the target weight and n ($4 \leq n \leq 5{,}000$), the number of elements of A. In the following line, n positive integers, $a_i \in A$ ($1 \leq i \leq n$), are given separated by a space. Each element a_i is in the range of 1 and 200,000 inclusively ($1 \leq a_i \leq 200{,}000$).

Output
Your program is to write to standard output. Print exactly one line which contains `YES` or `NO` according to the requirements above.

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

Sample Input 1	Output for the Sample Input 1
10 6 5 10 7 3 2 1	NO

Sample Input 2	Output for the Sample Input 2
21 7 10 1 4 6 2 8 5	YES

The 43rd Annual ACM
International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

Problem F
Parcel
제한 시간: 4 초

국제대학소포센터(ICPC: International Collegiate Parcel Center)는 전세계 대학생들을 대상으로 소포 무료 배송 이벤트를 진행하고 있다. 무료 배송 조건은 보낼 소포가 물품 4개로 구성되어야 하며 이들 물품의 무게 합이 정확히 정해진 정수 무게 w 그램이어야 한다는 것이다.

부산대학교에 있는 찬수는 영국 왕립대학에 있는 수환에게 보낼 물품이 매우 많이 있는데, 각 물품의 무게(모두 정수 그램)는 모두 다르다. 이 이벤트는 한시적으로 진행되는 이벤트이기 때문에 찬수는 자신이 보낼 물품 중에서 이 조건을 만족하는 물품 4개가 있는지 가능하면 빨리 알아내고 싶다. 다시 말해서 서로 다른 $n(n \geq 4)$개의 정수로 이루어진 집합 A에서 4개의 원소만 꺼내어 만든 부분집합 $B(|B| = 4)$가 $\sum_{b \in B} b = w$ 조건을 만족하는지 판단하려고 한다.

주어진 w와 A에 대해서, 위 조건을 만족하는 부분집합 B가 존재하면 YES를, 아니면 NO를 출력하는 프로그램을 작성하시오.

Input
입력은 표준입력을 사용한다. 입력의 첫 줄에는 무게 $w(10 \leq w \leq 799{,}994)$와 A의 원소 개수 $n(4 \leq n \leq 5{,}000)$이 공백으로 분리되어 주어진다. 다음 줄에는 A의 원소인 n개의 정수 $a_i \in A(1 \leq i \leq n)$가 공백으로 분리되어 주어진다. 각 원소 a_i는 1이상 200,000이하이다$(1 \leq a_i \leq 200{,}000)$.

Output
출력은 표준출력을 사용한다. 문제의 조건에 따라 YES나 NO를 한 줄에 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
10 6 5 10 7 3 2 1	NO

Sample Input 2	Output for the Sample Input 2
21 7 10 1 4 6 2 8 5	YES

Problem G
Passport Control
Time Limit: 0.2 Seconds

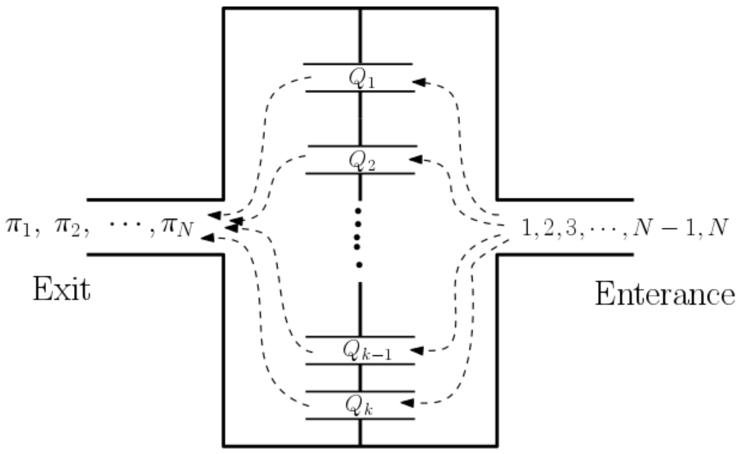


Figure G.1: N passengers should be controlled at one of the passport control offices $\{Q_k\}$.

All N passengers wait in one immigration queue for the passport control in the order of $[1, 2, 3, \dots, N-1, N]$ as shown in Figure G.1. Each passenger is required to be checked at one of k passport control queues. After completing the passport control, the passenger exits from the airport through the exit gate (Exit as shown in Figure G.1). The initial order of the entrance queue, $[1, 2, 3, \dots, N-1, N]$ can be changed to a shuffled order, $[\pi_1, \pi_2, \dots\ \pi_{N-1}, \pi_N]$ at the exit gate. You must determine if the exiting order $[\pi_1, \pi_2, \dots\ \pi_{N-1}, \pi_N]$ is possible using k parallel passport control queues. Let us explain with an example. In the case of $N = 3$ and $k = 2$, the exit orders $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$ are all possible, but $[3, 2, 1]$ is not.

Given an exit order $[\pi_1,\ \dots, \pi_N]$ as an input, write a program to print `YES` if the exit order is possible, or `NO` if not. Note that passengers are not allowed to change their order in passport control queues and each control queue is long enough to keep all N passengers.

Input
Your program is to read from standard input. The standard input consists of two lines. The first line gives two integers, N and k, where N is the number of passengers and k is the number of passport control queues. Note that $2 \le k \le N \le 100$. The second line gives $[\pi_1, \dots, \pi_N]$, an exit order of the passengers from the airport.

Output
Your program is to write to standard output. Print the string `YES` if the exit order is possible or `NO` otherwise.

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

Sample Input 1	Output for the Sample Input 1

`3 2`
`3 2 1`
	`NO`

Sample Input 2	Output for the Sample Input 2

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

The 43rd Annual ACM
International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

Problem G
Passport Control
제한 시간: 0.2 초

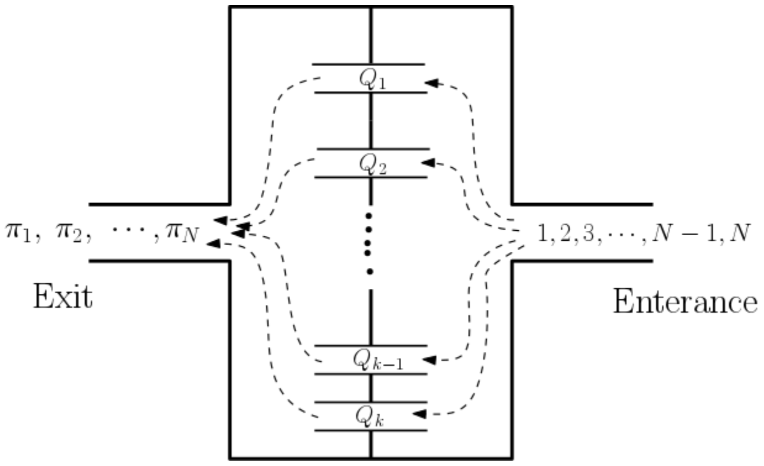


그림 G.1: N명의 입국 승객은 k개의 여권 심사 창구 $\{Q_k\}$ 중 하나를 반드시 거쳐야 한다.

N명의 입국 승객이 여권 심사를 위하여 그림 G.1 과 같이 입국 대기 줄에서 $[1, 2, \ldots, N-1, N]$ 순서로 기다리고 있다. 입국 승객은 준비된 k개의 여권 심사 창구 중 하나를 통과한 뒤 공항을 빠져나갈 수 있다. 입국할 때의 줄 선 승객의 순서를 $[1, 2, \ldots, N-1, N]$이라고 할 때 k개의 여권 심사 창구를 통과하여 입국장을 빠져나가는 순서 $[\pi_1, \pi_2, \ldots \pi_{N-1}, \pi_N]$는 처음과 달라질 수 있다. k개 여권 심사 창구가 준비되어 있을 때, 이 입국장을 빠져나가는 순서가 가능한 순서인가를 계산해야 한다.

예를 들어 설명해보자. 만일 $N = 3, k = 2$라고 할 때 입국장을 빠져나가는 순서 중 $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$는 가능하지만 $[3, 2, 1]$은 불가능하다. 여러분은 입국장을 빠져나가는 순서 $[\pi_1, \ldots, \pi_N]$가 입력으로 주어질 때, 이 순서가 가능하면 YES 를, 불가능하면 NO 를 출력해야 한다. 단, 특정 큐에 줄을 선 상황에서는 승객들의 순서를 임의로 바꿀 수는 없다. 그리고 각 여권 심사 창구에 준비된 큐는 N명 승객이 모두 들어올 정도로 충분히 크다고 가정한다.

Input
입력은 표준입력을 사용한다. 첫 번째 줄에는 두 개의 정수 N 과 k 가 주어진다. N은 입국 승객의 수이며 k는 여권 심사 창구의 수이다. 단, $2 \le k \le N \le 100$ 이다. 그리고 두 번째 줄에는 승객이 입국장을 빠져 나가는 순서 $[\pi_1, \ldots, \pi_N]$가 주어진다.

Output

출력은 표준출력을 사용한다. 입력으로 주어진 순서 $[\pi_1, \dots, \pi_N]$대로 입국장을 빠져나가는 것이 가능하면 `YES` 를 출력하고, 아니면 `NO` 를 출력해야 한다.

다음은 두 테스트 경우에 대한 입출력 예이다.

Sample Input 1

```
3 2
3 2 1
```

Output for the Sample Input 1

```
NO
```

Sample Input 2

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

Output for the Sample Input 2

```
YES
```

The 43rd Annual ACM
International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

Problem H
Path Embedding
Time Limit: 1 Second

Given a guest graph G and a host graph H with the same number of vertices, the graph embedding problem is to find a one-to-one correspondence σ from the vertex set $V(G)$ of G to the vertex set $V(H)$ of H, along with a mapping from each edge in the edge set $E(G)$ of G to a path in H. Many applications can be modeled as graph embedding. In particular, graph embedding has long been used to model the problem of arranging a parallel algorithm in a parallel architecture.

The quality of an embedding can be measured by certain cost criteria. Among others, the dilation is the maximum of the lengths of paths mapped by all edges of G. If the host graph H is a tree in which any two vertices are joined by a unique path, an edge (u, v) of G is necessarily mapped to the unique path in H joining $\sigma(u)$ and $\sigma(v)$. So, the dilation of an embedding σ of graph G into tree H can be simply represented as $\max_{(u,v)\epsilon E(G)} d_H(\sigma(u), \sigma(v))$, where $d_H(\sigma(u), \sigma(v))$ denotes the distance between $\sigma(u)$ and $\sigma(v)$ in H. The dilation of the embedding shown in Figure H.1 below, for example, is three.

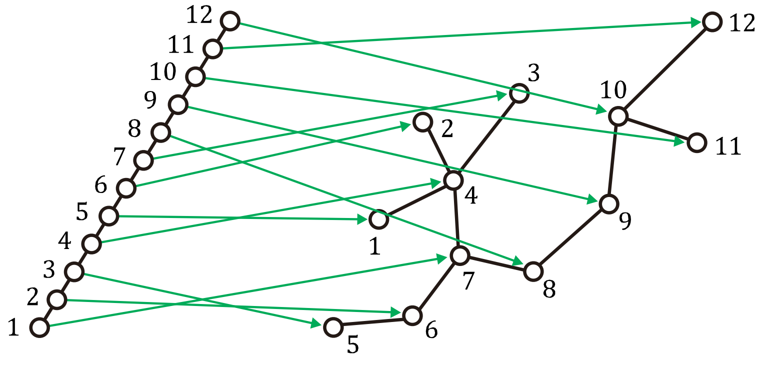


Figure H.1: An embedding σ of a path graph into a tree both with 12 vertices, where σ can be written in two-line notation $\begin{pmatrix} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12 \\ 7\ 6\ 5\ 4\ 1\ 2\ 3\ 8\ 9\ 11\ 12\ 10 \end{pmatrix}$, meaning $\sigma(1) = 7$, $\sigma(2) = 6$, $\sigma(3) = 5$, …, and $\sigma(12) = 10$.

We are concerned with the problem of embedding of a path graph into a tree, where a path graph is a tree that has at most two leaves. Given an embedding of a path graph into a tree, your job is to write an efficient program that finds the dilation of the embedding.

Input

Your program is to read from standard input. The first line contains an integer, n, representing the number of vertices of the host graph H which forms a tree, where $2 \leq n \leq 100{,}000$. It is followed by $n - 1$ lines, each contains two positive integers u and v that represent an edge between vertex u and vertex v of H. It is assumed that the vertices are indexed from 1 to n. The last line contains an ordering $\sigma(1)$, $\sigma(2)$, …, $\sigma(n)$ of the vertices of H, which represents the embedding of a path graph G into H, where $V(G) = \{1, 2, \dots, n\}$ and $E(G) = \{(u, v) : v = u + 1\}$.

Output
Your program is to write to standard output. Print exactly one line which contains an integer. The integer should be the dilation of the given embedding if the dilation is three or less; the integer should be `99` otherwise.

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

Sample Input 1
```
12
4 1
4 2
4 3
4 7
5 6
6 7
7 8
8 9
9 10
11 10
12 10
7 6 5 4 1 2 3 8 9 11 12 10
```

Output for the Sample Input 1
```
3
```

Sample Input 2
```
4
1 2
4 3
2 3
4 2 3 1
```

Output for the Sample Input 2
```
2
```

Sample Input 3
```
7
1 2
4 3
2 3
5 7
6 5
4 5
7 6 1 2 3 4 5
```

Output for the Sample Input 3
```
99
```

The 43rd Annual ACM

International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

Problem I
Registration
Time Limit: 1 Second

Print out your team's DOMJudge account ID and password.

Input
No input is given for this problem.

Output
Your program is to write to standard output. Print exactly two lines. The first line should contain your DOMJudge account ID, and the second line should contain your DOMJudge account password.

The following shows sample input and output, where the account ID is `team123` and the password is `passw0rd`. Notice that no input is given.

Sample Input	Output for the Sample Input
	team123 passw0rd

The 43rd Annual ACM
International Collegiate Programming Contest
Asia Regional – Seoul
Nationwide Internet Competition

Problem I
Registration
Time Limit: 1 Second

자신이 속해 있는 팀의 DOMJudge ID 와 패스워드를 그대로 출력하는 프로그램을 작성하시오.

Input

이 문제는 입력이 없다.

Output

표준출력(standard output)으로 출력해야 한다. 첫 줄에 DOMJudge ID, 둘째 줄에 패스워드를 출력한다.

다음은 ID 가 `team123` 번, 패스워드가 `passw0rd` 인 경우의 입출력 예제이다. 참고로 입력이 없는 것에 주의한다.

Sample Input	Output for the Sample Input
	`team123` `passw0rd`

Problem J
Sliding Blocks
Time Limit: 1 Second

When rectangular blocks, which are all parallel to the xy-axis, are given one by one in the plane, each block moves along the specified direction from the specified starting location. The movement of a block consists of two phases. For the first phase, each block moves from the specified location along the given direction which is always left-downward. While moving if it hits either the x-axis, the y-axis, or another block it changes its direction and enters the second phase. More specifically, if the bottom part of the moving block hits the x-axis or another block its moving direction changes leftward and it enters the second phase. Similarly, if the left part of the moving block hits the y-axis or another block its moving direction changes downward and it enters the second phase.

The second phase movement consists of repeating ‘move-left’ and ‘move-down’ (or ‘move-down’ and ‘move-left’) as long as the block can move. Once the block enters the second phase, it keeps moving in the same direction as farther as possible. If it is blocked either by the xy-axis or by another block it changes its moving direction and continues moving.

A block’s location is denoted by the (x, y) coordinate of its lower-left corner point. Figure J.1 shows an example how a block moves. The size of each block is shown in the figure. Suppose seven blocks (A to G, all are colored green) are already placed and block H has the turn to move. The dotted rectangle indicates the starting and the intermediate locations of block H (orange block at final position). The red lines indicate the moving path. As Figure J.1 shows it first moves left-downward (in the first phase) until it hits block C. Then the direction changes leftward, which means it enters the second phase of movement, and continues moving as farther as possible until it hits block F. Then it continues moving in the second phase by changing directions until it can no longer move.

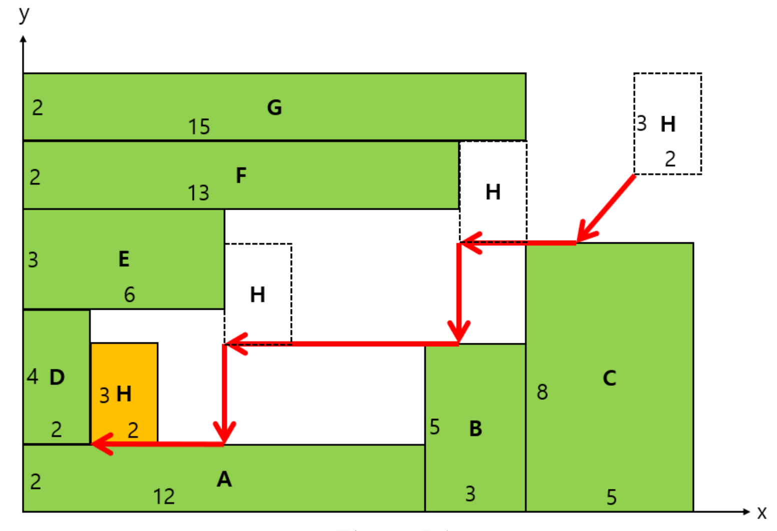


Figure J.1

How to treat some special cases will be explained with an example as shown in Figure J.2. While a block, say G, is moving left-downward in the first phase, if the lower left corner point of it hits another block, say block D, at the upper right corner point of D, G may have two choices of changing direction either downward or leftward. In such case, moving downward has higher priority than moving leftward. If G, in such case, is unable to move downward due to some other blocks, then its direction changes leftward. While block G is moving left-downward its upper-left corner point (or its lower-right corner point) may hit another block's corner points as shown in Figure J.2 (Note that block F and block C hit block G at their corner points during the first phase movement.) In such case, block G continues moving in the same direction since it has not been blocked by the corner points.

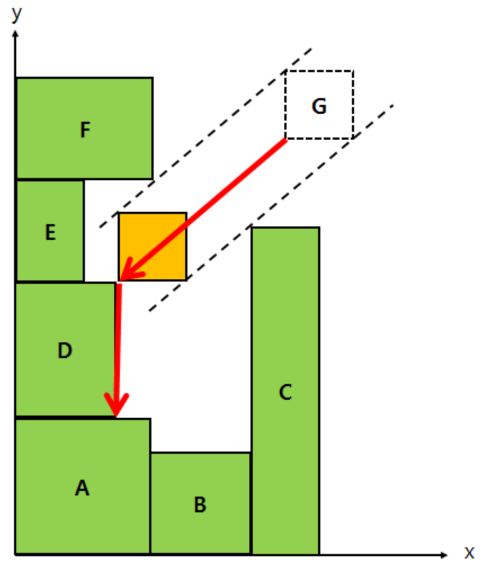


Figure J.2

The movement rules can be summarized as follows:

1. Each block begins move at the given location along the given direction. The direction is always left-downward.
2. A moving block changes its direction when it is blocked either by the x-axis, by the y-axis, or by another block.
3. The moving block keeps moving by changing directions until it can no longer move.

For each block, following information is given as shown in <Fig. 3>.

1. The width and height $w,\ h$ of the block.
2. The starting location (x, y) at which the block begins move.
3. $(\Delta x, \Delta y)$ to indicate the direction of the first movement (see the red line in Figure J.3). Each block begins move from location (x, y) toward $(x - \Delta\text{x}, y - \Delta\text{y})$.

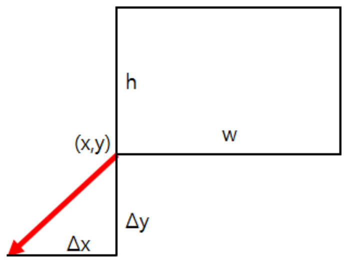


Figure J.3

Given information on n rectangular blocks, write a program which finds the location of the last block after moving every block one by one according to the movement rules.

You may assume that the $i^{th}(2 \leq i \leq n)$ block at its the starting location is overlapped with none of the previous blocks (i.e., blocks of indices between 1 and $(i-1)$) and that some block may not move at the very starting location since it is blocked by other blocks.

Input

Your program is to read from standard input. The input starts with a line containing an integer, n $(1 \leq n \leq 1{,}000)$, where n is the number of blocks. Each of the following n line contains information on each block. Each block information consists of 6 integers. The first two integers w, h $(1 \leq w, h \leq 100{,}000)$ indicate the size of it. Next two integers (x, y) $(0 \leq x, y \leq 10^8)$ indicate the starting location of it. Next two integers $(\Delta x, \Delta y)$ $(1 \leq \Delta x, \Delta y \leq 10^5)$ represent the direction for the first movement as explained above.

Output

Your program is to write to standard output. Print exactly one line which contains two integers x and y, where (x, y) is the location of the last block after moving every block one by one according to the movement rules.

The following shows sample input and output for two test cases. (Note that the first sample corresponds to Figure J.1 and the second to Figure J.2)

Sample Input 1

```
8
12 2 100 100 1 1
3 5 100 1 1 1
5 8 100 1 1 1
2 4 0 100 1 1
6 3 0 100 1 1
13 2 1 100 100 1
15 2 1 100 100 1
2 3 18 10 1 1
```

Output for the Sample Input 1

```
2 2
```

Sample Input 2

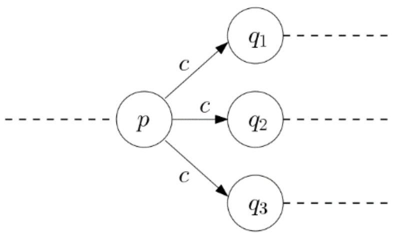
```
7
4 4 1000 1 1 1
3 3 1000 1 1 1
2 10 1000 1 1 1
3 4 1 100 1 1
2 3 0 100 1 1
4 3 0 100 1 1
2 2 10 15 1 1
```

Output for the Sample Input 2

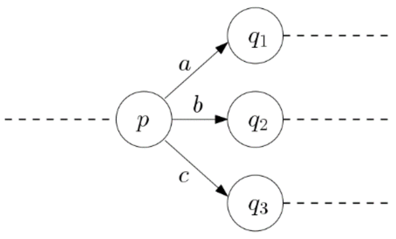
```
3 4
```

Problem K
Suffix-Freeness
Time Limit: 0.3 Seconds

A deterministic finite automaton (DFA) is a labeled directed graph. That is, instead of having a directed edge in the form (p, q), edges are given as a triple (p, q, c), where p and q are nodes in the graph and c is a label character from an alphabet. At each node, there is at most one outgoing edge per character. In other words, there is no subset of edges in the form $(p, q_1, c), (p, q_2, c), (p, q_3, c), \ldots$ of size greater than 1. On the other hand, a node can have a several outgoing edges all of which have different labels. In a DFA, one node is designated as the starting node q_0 and a subset F of nodes are designated as final nodes.



DFA (X)

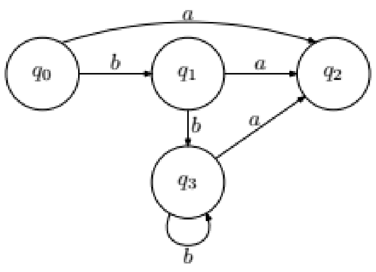


DFA (O)

A DFA is specified by a tuple(N, K, M, q_0, F), where N is a set of states, K is an input alphabet, M is a set of transitions, q_0 is the start state and F is a set of accepting states. Given a DFA D and a string $w = w_1 w_2 \ldots w_n$, we say that D accepts w if there is a series of edges $(q_0, q_1, w_1), (q_1, q_2, w_2), \ldots, (q_{n-1}, q_n, w_n)$ spelling out w and q_n is a final node in F. Note that nodes and edges can be visited multiple times. In this sense, a DFA can store a (possibly infinite) set of strings, namely the set of all strings accepted by the DFA. DFAs are useful in several practical applications. For instance, in software verification or pattern matching, often a target object is represented as a DFA for efficient processing.

Given two strings x and y, we say that y is a suffix of x if there is another string u such that $x = uy$ (the concatenation of u and y in order). For example, for a string *icpc2018*, all strings *icpc2018*, *cpc2018*, *pc2018*, *c2018*, *2018*, *018*, *18*, *8*, λ are suffixes of *icpc2018*, where λ denotes the empty string. A set of strings is suffix-free if there is no pair of distinct strings x and y in the set such that y is a suffix of x. For instance, {*2018*, *18*, *icpc*} is not suffix-free since *18* is a suffix of *2018*. Similarly {λ, *icpc2018*, *icpc*} is not suffix-free since λ is a suffix of *icpc2018* and *icpc*. In fact, λ is a suffix of all nonempty strings and, thus, any set containing λ and any other nonempty string is not suffix-free. Also, by definition, empty set is always suffix-free.

Given a DFA D, the language $L(D)$ of D is the set of strings accepted by D. Then we say that D is suffix-free if there are no two distinct strings x and y in $L(D)$ such that y is not a suffix of x. Suffix-freeness plays an important role in several applications including efficient pattern matching. If no string is accepted by D, then $L(D)$ is empty and, therefore, is suffix-free.



Your task is to determine whether or not a given DFA is suffix-free. In the right figure, the arrows correspond to labeled edges (transitions) in M. For example, there is an edge

(q_0, q_2, a) and (q_3, q_3, b). Assume that q_2 is the only final node in the DFA; namely, $F = \{q_2\}$. The string a is accepted as there is a path (q_0, q_2, a). Furthermore, $bbba$ is also accepted by (q_0, q_1, b), (q_1, q_3, b), (q_3, q_3, b), (q_3, q_2, a). Since a is a suffix of $bbba$, this DFA is not suffix-free. Note that these are not the only examples. For all inputs, you can assume that all nodes in the DFA are reachable from the starting node.

Input

Your program is to read from standard input. The input starts with a line containing four integers n, m, k, f, where n $(1 \le n \le 2{,}000)$ is the number of nodes, k $(1 \le k \le 26)$ is the number of characters in the alphabet, m $(1 \le m \le kn)$ is the number of edges and f $(1 \le f \le 2{,}000)$ is the number of final nodes. The nodes in the graph are labeled from 0 to $n - 1$, where 0 is the start node. The alphabet consists of the first k lowercase English letters. The following line contains f integers, separated by spaces, corresponding to the final node labels. The following m lines cointain two integers p and q and a character c, all separated by spaces, per line, which corresponds to the labeled edge (p, q, c).

Output

Your program is to write to standard output. Print exactly one line which contains 1 (if D is suffix-free) or 0 (otherwise).
The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
4 6 2 1 2 0 2 a 0 1 b 1 2 a 1 3 b 3 2 a 3 3 b	0

Sample Input 2	Output for the Sample Input 2
4 7 3 1 3 0 1 a 0 2 b 0 3 c 1 1 b 1 3 a 2 2 c 2 3 b	1

Problem L
Three Robots
Time Limit: 2 Seconds

An undirected weighted graph G is given and G is connected, that is, arbitrary two vertices in G must be connected by a path. Three robots are exploring G through edges. Here the weight of each edge means the time to be spent when a robot passes through it. The speeds of all the robots are identical and at least two robots may be passed through a same edge at a time. At some instant of the explorations of the robots, all of them should get together at a vertex to share their information. It is called a *rendezvous*.

Initially, three robots lie on specified vertices. Of course, at least two robots may be located on an identical vertex. Also all the three robots start to move simultaneously. We wish to know the minimum time needed to fulfill the first rendezvous.

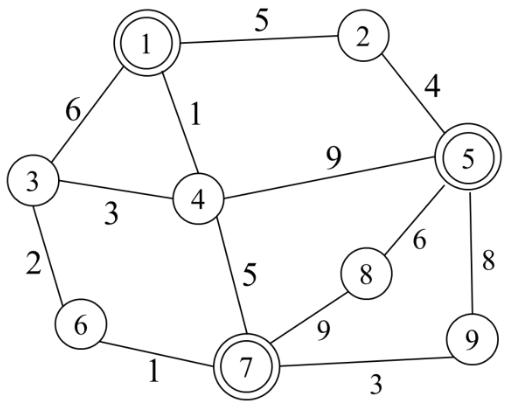


Figure L.1

For example, initially, three robots are located on the vertices 1, 5, and 7 in Figure L.1. A movement of the robot on the vertex 1 into the vertex 9 requires at least 9 time units. Also it takes at least 8 and 3 time units, respectively that the robots on the vertices 5 and 7 travel into the vertex 9. So the rendezvous at the vertex 9 requires at least 9 time units and it is the minimum time needed for the first rendezvous. Of course, the first rendezvous happens either at the vertex 1 or 4 and it also requires the minimum time of 9.

Given a weighted and connected graph G and the initial locations of three robots, write a program to find the minimum time needed to fulfill the first rendezvous.

Input
Your program is to read from standard input. The input starts with a line containing two integers, N and M ($1 \leq N \leq 20{,}000$, $N-1 \leq M \leq 100{,}000$), where N and M are the numbers of vertices and edges of G, respectively. The vertices of G are represented by $1, 2, \ldots, N$. In each of the following M lines, three integers a, b, and t ($1 \leq a \neq b \leq N$, $1 \leq t \leq 10{,}000$) are given, where an edge connects the two vertices a and b, and its weight is t. The last $(M+2)$-th line contains three integers u, v, and w, which are the initial locations of

three robots ($1 \leq u, v, w \leq N$). Of course, it is possible that at least two of the three robots are initially located on an identical vertex.

Output

Your program is to write to standard output. Print exactly one line which contains the minimum time needed to fulfill the first rendezvous.

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

Sample Input 1

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

Output for the Sample Input 1

```
4
```

Sample Input 2

```
9 13
1 2 5
3 1 6
1 4 1
2 5 4
3 4 3
5 4 9
6 3 2
4 7 5
8 5 6
7 8 9
5 9 8
7 6 1
7 9 3
1 5 7
```

Output for the Sample Input 2

```
9
```